**EDER MATEUS NUNES GONÇALVES**

# UMA ABORDAGEM PARA ESPECIFICAÇÃO DE CONHECIMENTO PARA SISTEMAS MULTIAGENTES COGNITIVOS

**FLORIANÓPOLIS**
**2006**

**UNIVERSIDADE FEDERAL DE SANTA CATARINA**

**PROGRAMA DE PÓS-GRADUAÇÃO**
**EM ENGENHARIA ELÉTRICA**

# UMA ABORDAGEM PARA ESPECIFICAÇÃO DE CONHECIMENTO PARA SISTEMAS MULTIAGENTES COGNITIVOS

Tese submetida à
Universidade Federal de Santa Catarina
como parte dos requisitos para a
obtenção do grau de Doutor em Engenharia Elétrica.

**EDER MATEUS NUNES GONÇALVES**

Florianópolis, Agosto de 2006.

# Uma Abordagem para Especificação de Conhecimento para Sistemas Multiagentes Cognitivos

Eder Mateus Nunes Gonçalves

'Esta Tese foi julgada adequada para a obtenção do título de Doutor em Engenharia Elétrica, Área de Concentração em *Controle, Automação e Informática Industrial*, e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Santa Catarina.'

Prof. Guilherme Bittencourt, Dr.
Orientador

Prof. Nelson Sadowski, Dr
Coordenador do Programa de Pós-Graduação em Engenharia Elétrica

Banca Examinadora:

Prof. Guilherme Bittencourt, Dr.

Profa. Maria del Rosário Girardi, Dra.

Prof. Augusto Loureiro da Costa, Dr.

Profa. Silvia Silva da Costa Botelho,, Dra.

Prof. Ricardo José Rabelo, Dr.

*E o caos concebe sua obra prima!*

*William Shakespeare*

# AGRADECIMENTOS

Olhando para trás, até o dia em que decidi pela empreitada de uma tese de doutorado, e realizando o balanço final deste projeto, chego a conclusão de que se ela foi finalizada, foi muito mais pelo suporte emocional que recebi, do que pelos meus próprios méritos. A persistência e a paciência, minhas principais virtudes neste período, sustentaram-se neste pilar.

Neste aspecto, minha família tem grande responsabilidade. E quando digo família, entenda-se como algo que extrapola os laços consangüíneos, mas que é determinado por valores de amizade, lealdade e confiança.

À meus pais, e ao meu irmão, que nas horas de pânico fizeram o que deles eu sempre esperei, e até o que não esperei, me colocando em pé quando imaginei que jamais poderia me reerguer. Muitas vezes, um gesto ou um olhar é muito mais expressivo do que mil palavras.

À uma mulher que em apenas uma pessoa consegue ser amiga, namorada, esposa, e as vezes até mãe, e que assim se torna a síntese do meu amor. Obrigado Suzane, se eu cheguei até o fim, tu também tem uma grande parcela de culpa.

Dedico esta vitória também a dois irmãos que "adotei" durante a vida, e que ajudaram a moldar o principal valor que tenho na vida: amizade. Ao Sam e ao Maurício, um obrigado por estarem comigo sempre.

Agradeço também aqueles fizeram parte da minha vida nesse período em que vivi em Florianópolis: Jerusa Marchi, Paulo Mafra, Luciano Rotava, Rafael Obelheiro, Márcio Hangai, Carlos Brandão, Fred Freitas, Sérgio Melo, . . . , dentre tantos outros que também fizeram parte desta história. Podem ter a certeza de que sempre nos reencontraremos.

E se estes foram os que me apoiaram até o fim, devo citar também aqueles que determinam o norte da minha vida. Dedico este trabalho àqueles que me fizeram compreender a grandeza de ser um professor: Guilherme Bittencourt e Augusto Loureiro. Agradeço a ambos pelo privilégio e pela oportunidade de trabalhar consigo. Rapidamente se transformaram um referencial profissional, mas hoje se tornaram exemplos de homens que desejo seguir. Tendo vocês como referência, sei que no fim, tudo termina bem.

Resumo da Tese apresentada à UFSC como parte dos requisitos necessários para obtenção do grau de Doutor em Engenharia Elétrica.

# UMA ABORDAGEM PARA ESPECIFICAÇÃO DE CONHECIMENTO PARA SISTEMAS MULTIAGENTES COGNITIVOS

**Eder Mateus Nunes Gonçalves**

Agosto/2006

Orientador: Guilherme Bittencourt, Dr.
Área de Concentração: Controle, Automação e Informática Industrial
Palavras-chave: Sistemas Multiagentes, Aquisição de Conhecimento, Redes de Petri, Futebol de Robôs
Número de Páginas: xxii + 190

A *Aquisição de Conhecimento* em *Sistemas Multiagentes Cognitivos*, além de estar sujeita às dificuldades de um processo convencional de desenvolvimento segundo uma abordagem baseada em conhecimento, deve ainda supor a imersão do agente em uma sociedade, cujo modelo de mundo deve considerar as mudanças impostas pelos outros membros do sistema. Não obstante, a aquisição de conhecimento é, ainda hoje, viabilizada sem uma metodologia ou sistemática padrão, sujeita às especificidades de cada domínio. Neste trabalho, propõe-se uma abordagem para aquisição de conhecimento voltada a sistemas multiagentes cognitivos, baseado em um modelo de cognição genérico, utilizando *Redes de Petri* como *linguagem de especificação*. As principais características desta abordagem são a estruturação do conhecimento de um domínio em diferentes níveis de abstração, a utilização de uma linguagem única desde a especificação social até a especificação inidividual no nível de ações no ambiente, e a capacidade de adaptação a qualquer formalismo de representação de conhecimento. Os principais méritos do modelo foram o deslocamento do foco do projeto para o nível de conhecimento do sistema, permitindo que engenheiro de conhecimento e especialista interajam independente dos aspectos de implementação. Além disso, o nível de especificação social permite uma maior flexibilidade de ações por parte dos agentes, devido ao modelo de coordenação adotado, garantindo maior robustez ao sistema. Este modelo é empregado na Expert-Coop++, um arcabouço para o desenvolvimento deste tipo de sistema e foi implementado no desenvolvimento de uma equipe de futebol de robôs, o UFSC-Team, segundo as diretrizes da Robocup.

Abstract of Dissertation (Thesis) presented to UFSC as a partial fulfillment of the requirements for the degree of Master (Doctor) in Electrical Engineering.

# AN APPROACH TO KNOWLEDGE SPECIFICATION TO COGNITIVE MULTI-AGENT SYSTEMS

**Eder Mateus Nunes Gonçalves**

August/2006

Advisor: Guilherme Bittencourt, Dr.
Area of Concentration: Control, Automation and Industrial Informatics
Key words: Multi-Agent Systems, Knowledge Acquisition, Petri Nets, Soccer Robotic
Number of Pages: xxii + 190

The *Knowledge Acquisition* in *Cognitive Multi-Agent Systems*, beyond to be subject to the difficulties of a conventional process of development of an approach based in knowledge, it must yet consider the changes made by the other members of the system. Nevertheless, the knowledge acquisition is, still today, considered without a standard methodology or systematic, subject to the specificity of each domain. In this work, it is proposed a knowledge acquisition approach to cognitive multi-agent systems based in a cognition generic model, using *Petri Nets* as *specification language*. The main features of this approach is the knowledge structuring of a domain in different abstraction levels, the use of a only language since the social specification until the individual specification in the action level in the environment, and the capacity to adapt to any formalism of knowledge representation. The main advantages of this model was the shift of the project focus to the knowledge level of the system. In this way, the knowledge engineer and the domain expert can interact independent of the implementation aspects. Besides that, the social specification level permits a better flexibility of actions, because the coordination model used. This model is used in the Expert-Coop++, a shell to develop systems of this kind and it was implemented in the development of a team of robotic soccer, the UFSC-Team, according to Robocup guidelines.

# Sumário

# Lista de Figuras

# Lista de Tabelas

# Capítulo 1

# Introdução

A *Inteligência Artificial Distribuída* (IAD) é um subdomínio da *Inteligência Artificial* (IA) que combina métodos e técnicas da IA e de *Sistemas Distribuídos* (Durfee e Rosenschein, 1994). A IAD está especialmente interessada na resolução de problemas *complexos distribuídos*, como a recomposição de uma rede elétrica após uma queda do sistema, o monitoramento e controle do ambiente natural de um parque florestal, o gerenciamento de um sistema de educação a distância, ou o controle de uma equipe de robôs que jogam futebol. Problemas deste tipo compartilham as seguintes características:

- eles são fisicamente e/ou conceitualmente distribuídos, no sentido de que seu estado global é composto pela agregação de estados locais parcialmente independentes,
- as tarefas envolvidas na resolução destes problemas referem-se a diferentes níveis de abstração, variando desde protocolos de coordenação global até procedimentos de percepção/ação local, que usam sensores/atuadores para perceber/agir no mundo.

Uma técnica possível para resolver esta classe de problemas são os *Sistemas Multiagentes Cognitivos*, onde cada agente possui metas e conhecimento locais, associados a um dos estados locais do problema, e também compartilha metas globais com outros agentes, determinando como os estados locais são agregados em um estado global (Garcia e Sichman, 2003). Devido aos diferentes níveis de abstração do problema, o comportamento de cada agente no sistema multiagente pode ser descrito por uma estratégia hierárquica baseada em um ou mais *Sistemas Baseado em Conhecimento* (SBC) (Rezende et al., 2003).

A forma como elicitar esta estratégia e como representar as bases de conhecimento associadas aos SBC é um problema de *aquisição de conhecimento* (Garcia et al., 2003).

O sucesso de um SBC depende diretamente da quantidade e qualidade do conhecimento codificado em sua base de conhecimento (Feigenbaum, 1977). Nesse sentido, o processo de aquisição de conhecimento tem importância fundamental no desenvolvimento de qualquer SBC. Por um lado, o projetista do sistema, o engenheiro de conhecimento, deve representar o conhecimento adquirido usando um determinado formalismo de representação de conhecimento, em geral, uma tarefa bastante complexa. Por outro lado, a pessoa que detém o conhecimento do domínio em pauta, o especialista, normalmente possui formação e perspectivas específicas a respeito do problema, que podem levar a sérios problemas de comunicação com o engenheiro de conhecimento e, conseqüentemente, comprometer todo o processo de aquisição.

No caso de um sistema multiagente cognitivo composto por vários SBC's, onde cada um possui apenas um conhecimento parcial do estado do mundo e um conjunto limitado de possíveis ações, problemas de comunicação entre agentes aumentam ainda mais a complexidade do processo de aquisição de conhecimento. Além disso, as metodologias clássicas de desenvolvimento de SBC's possuem apenas aplicação parcial a esta classe de problemas, onde deve-se construir várias bases de conhecimento, cada qual associada com um nível de abstração do problema.

Neste contexto, as principais motivações deste trabalho são:

- a necessidade de tratar a complexidade crescente no desenvolvimento de SBC's, especialmente nos casos em que estes devem estar inseridos em um sistema multiagente;
- a descontinuidade entre o nível social, definido com o uso de plataformas para Sistemas Multiagentes, e individual, definido com ferramentas "clássicas" de aquisição de conhecimento. Em outras palavras, refere-se a mecanismos que estabeleçam uma relação formal entre a estratégia social empregada e as respectivas ações no ambiente.
- e a necessidade de metodologias e/ou abordagens com fundamentação matemática.

## 1.1 Problema

Sendo assim, pode-se delimitar o problema genérico de aquisição de conhecimento em sistemas multiagentes cognitivos nos seguintes termos. Tem-se um sistema constituído por dois níveis fundamentais: o nível social, onde define-se a estratégia de ação coletiva do sistema; e o nível individual,

que corresponde ao conhecimento de cada agente, constituído pela instanciação do conhecimento social segundo as ações disponíveis e o papel do agente dentro do ambiente.

O nível social corresponde à especificação dos possíveis cenários, ou planos, tratados pelo sistema. Forma o chamado conhecimento social, que deve ser compartilhado por todos os agentes a fim de garantir a correta coordenação de ações no sentido dos objetivos do grupo. Dado o seu grau de abstração, não contém um nível de implementação em si, apenas projeções nas bases sociais dos diversos agentes, que servem para estabelecer um plano conjunto de ações.

O nível individual corresponde à especificação e implementação dos SBC's de cada agente. Cada agente pode ser constituído por um ou mais SBC's, dependendo da complexidade cognitiva e arquitetura subjacente. Não obstante, cada SBC contém características próprias, como os formalismos de representação de conhecimento e métodos de resolução de problemas utilizados. O ideal é que o conhecimento individual seja concebido no nível de conhecimento do sistema, para então ser determinado o respectivo nível de implementação. Além disso, é importante que a linguagem de especificação utilizada permita uma estruturação do conhecimento do agente em múltiplos níveis de abstração, segunda sua complexidade cognitiva, e uma estruturação genérica dos formalismos de representação de conhecimento

## 1.2 Objetivos

O objetivo desta tese é estabelecer uma abordagem de aquisição de conhecimento para Sistemas Multiagentes Cognitivos. Esta abordagem deve estar adequada às diretrizes básicas de qualquer processo de aquisição de conhecimento no nível de desenvolvimento individual dos agentes, considerando, contudo, as características inerentes à inserção destes agentes em ambientes sociais.

Um processo padrão de aquisição de conhecimento prevê a existência de três etapas fundamentais: elicitação, análise e interpretação do conhecimento (Kidd, 1987). Um modelo amplamente difundido (Buchanan et al., 1983) permite o refinamento destas etapas, em cinco outras: identificação, conceitualização, formalização, implementação e teste. A abordagem aqui proposta, em cada nível de concepção do sistema, deve cobrir principalmente as etapas de análise e interpretação, cujo refinamento abrange parte da conceitualização, a formalização, e parte da implementação de um SBC. A proposta não considera a etapa de elicitação e suas extensões dentro do processo.

Dada a ausência de ferramentas sustentadas pelo rigor de bases formais lógico-matemáticas, sugere-se uma abordagem baseada na utilização de Redes de Petri (Cardoso e Valette, 1997) como

linguagem de especificação do sistema. Esta linguagem permite a especificação do sistema em nível de conhecimento, por intermédio de cenários elaborados em conjunto pelo especialista mais o engenheiro de conhecimento, durante a etapa de formalização. Estes cenários constituem o conjunto de situações manipuladas pelo sistema, cuja complexidade pode ser medida pelo número de cenários tratáveis e as relações de precedência e causalidade entre eles. Uma vez simuladas e verificadas, as Redes de Petri são utilizadas para a construção automática das bases de conhecimento do sistema.

As Redes de Petri foram escolhidas como linguagem de especificação na medida em que seu modelo gráfico pode ser utilizado como linguagem diagramática que auxilia na interação entre especialista e engenheiro de conhecimento, seu modelo matemático serve como base para os processos de verificação de anomalias nas bases de conhecimento, e sua composição hierárquica permite a composição do sistema em múltiplos níveis de abstração, dando modularidade a abordagem.

Desta forma, a abordagem pode ser compreendida a partir dos seguintes requisitos:

- a utilização de uma única abordagem de especificação para os diferentes níveis de abstração do sistema;
- modelo para especificação social segundo o paradigma de orientação à agentes, onde é possível a representação explícita de conceitos sociais;
- modelo para especificação social que permite maior flexibilização e exploração do potencial de atuação dos agentes, por intermédio de um mecanismo de coordenação social;
- modelo de especificação individual segundo uma arquitetura genérica de cognição (Barkley, 1997) em uma abordagem baseada em conhecimento;
- aproveitamento dos resultados teóricos e ferramentas já desenvolvidos em torno da utilização de Redes de Petri no nível de especificação individual;
- linguagem de especificação apta à estruturação de diversos formalismos de representação de conhecimento, como lógica, quadros e redes semânticas;
- transformação automática do conhecimento estruturado em Redes de Petri em bases de conhecimento, independente da arquitetura subjacente.

## 1.3 Metodologia

No que se refere a metodologia empregada nesta tese, ela divide-se em dois contextos. O primeiro considera a aplicação da abordagem de aquisição de conhecimento para Sistemas Multiagentes Cognitivos no desenvolvimento de uma equipe de futebol de robôs, dentro de um cenário delimitado do ambiente. No segundo, considera-se a comparação da abordagem proposta com outras existentes na literatura. Na medida em que não existem abordagens e/ou metodologias com relação direta àquela proposta, algumas restrições a esta análise devem ser impostas com o objetivo de viabilizá-la.

Quanto ao desenvolvimento de uma equipe de futebol de robôs, esta idéia está contida dentro do desenvolvimento do UFSC-Team (da Costa e Bittencourt, 1999b), um projeto de pesquisa que busca soluções para os problemas propostos pela Robocup (Federation, 2003). A Robocup é uma iniciativa mundial de pesquisadores em Inteligência Artificial (IA) que visa o estabelecimento de uma plataforma de pesquisa em torno de um problema padrão: o futebol de robôs. Entretanto, o objetivo aqui não é a construção completa da equipe, mas apenas desenvolver um sistema considerando um número limitado de agentes/jogadores para a execução de uma jogada específica no ambiente.

O UFSC-Team estabelece uma arquitetura de agente cognitivo e social sobre a qual são desenvolvidos clientes/jogadores para disputas realizadas com o uso de um simulador, o Soccerserver (Chen et al., 2001). O Soccerserver é um simulador de partidas de futebol, desenvolvido dentro do âmbito da Robocup, envolvendo agentes computacionais. Esta arquitetura de agente prevê a divisão da complexidade do problema em níveis de decisão, de forma a atribuir a cada nível, diferentes aspectos do conhecimento necessário à resolução do problema.

Para a implementação deste agente cognitivo foi utilizada a Expert-Coop++ (da Costa et al., 2003), uma biblioteca orientada a objetos destinada ao desenvolvimento de sistemas multiagentes sob restrições de tempo real do tipo melhor esforço. Esta biblioteca, assim como um arcabouço para o desenvolvimento de sistemas especialistas, provê uma série de funcionalidades específicas a este tipo de solução, como métodos de resolução de problemas e formalismo de representação de conhecimento. Entretanto, a Expert-Coop++ não prevê uma metodologia e/ou abordagem que auxilie de forma sistemática e estruturada a etapa de aquisição de conhecimento do agente, em seus diversos níveis.

Segundo a abordagem de construção por intermédio da Expert-Coop++, o conhecimento do agente é provido pela codificação das regras de cada nível de decisão: reativo, instintivo e cognitivo.

O nível reativo tem uma natureza diferenciada dos demais níveis, na medida em que é constituído por um conjunto de controladores nebulosos que caracterizam os diferentes comportamentos

do agente no ambiente. Os controladores prevêem a definição de suas regras pela manipulação de uma série de conjuntos nebulosos definidos a partir das características específicas do domínio. Entretanto, segundo as especificações da arquitetura do agente, este nível está imerso em um contexto evolutivo e, portanto, deve ser aperfeiçoado segundo os princípios da seleção natural. Por este motivo, o conhecimento do nível reativo está fora do escopo da proposta desta tese.

O conhecimento dos níveis instintivo e cognitivo deve estabelecer, respectivamente, a caracterização dos estados do ambiente, e o planejamento de ações em um contexto individual e multiagente. Desta forma, uma abordagem de aquisição de conhecimento deve ser estabelecida de forma a orientar a codificação destas bases de regras, levando em consideração as singularidades da arquitetura do agente.

Por outro lado, para viabilizar a comparação da abordagem de aquisição de conhecimento proposta com outras abordagens, estabelece-se um conjunto de restrições sobre a literatura no sentido de estabelecer um referencial teórico compatível com os objetivos do trabalho. Por isso, esta análise comparativa pode ser divida em três tipos de referências:

- o primeiro tipo se refere a metodologias e/ou abordagens de desenvolvimento de Sistemas Multiagentes utilizando ferramentas como AUML, ZEUS e GAIA. Nesse caso, a abordagem é comparada quanto a especificação do nível social do sistema.
- o segundo tipo se refere a metodologias de desenvolvimento de SBC's utilizando ferramentas como EMYCIN/TEIRESIAS, ONCOCIN/OPAL, e KADS/CommonKADS. Estas metodologias estabelecem um referencial teórico para o desenvolvimento do nível individual do sistema.
- o terceiro tipo se refere a modelos que utilizam Redes de Petri para a especificação e implementação de Sistemas Multiagentes Cognitivos, seja no nível social e/ou individual.

## 1.4 Organização do Trabalho

Sendo assim, a tese é estruturada da seguinte maneira.

O capítulo 2 apresenta uma revisão da literatura sobre o tema desta tese: Engenharia de Conhecimento em Sistemas Multiagentes Cognitivos. Ele é dividido em duas partes: a primeira relativa aos princípios básicos sobre Engenharia do Conhecimento englobando conceitos sobre SBC's, Aquisição de Conhecimento e metodologias e/ou abordagens de desenvolvimento para este tipo de sistema. A

segunda parte aborda os conceitos relativos ao desenvolvimento de Sistemas Multiagentes e as metodologias existentes na literatura.

O capítulo 3 apresenta as Redes de Petri como parte da solução da metodologia de aquisição de conhecimento. Sua descrição dá-se apenas no plano fundamental da matéria, sem entrar em aspectos relativos ao estado da arte. Esta abordagem justifica-se pela necessidade de validação do modelo proposto. O capítulo encerra com a revisão dos principais trabalhos que utilizam Redes de Petri em algum nível de desenvolvimento de Sistemas Multiagentes Cognitivos.

O capítulo 4 descreve a abordagem proposta de aquisição de conhecimento para Sistemas Multiagentes Cognitivos utilizando Redes de Petri. Para isso, descreve-se um modelo de cognição genérico que considera a concepção do conhecimento em dois níveis fundamentais, o social e o individual. A seguir, descreve-se um modelo de Rede de Petri de Alto-Nível, como linguagem de especificação, capaz de relacionar e integrar os aspectos social e individual do conhecimento, estruturado em múltiplos níveis hierárquicos de abstração. O capítulo é encerrado com uma discussão a respeito da relação entre a abordagem proposta com aquelas existentes na literatura.

O capítulo 5 descreve os resultados relativos à implementação da abordagem proposta no desenvolvimento de uma equipe de futebol de robôs. O capítulo inicia com uma breve descrição do problema de construção de equipes de futebol de robôs no contexto da Robocup, e a arquitetura e metodologia de desenvolvimento de agente subjacente. A seguir, o capítulo apresenta o processo de construção das bases de regras relativas ao problema estabelecido, finalizando com as observações e conclusões a respeito da aplicação da metodologia.

Finalmente, o capítulo 6 descreve as conclusões deste trabalho e suas perspectivas futuras.

# Capítulo 2

# Inteligência Artificial Distribuída e Engenharia de Conhecimento

## 2.1 Introdução

Segundo o enunciado do problema apresentado na Introdução[1], o trabalho desenvolve-se em torno de dois aspectos: primeiro, o nível social, que se constitui na especificação dos mecanismos que implementam um Sistema Multiagente (SMA), e o segundo, o nível individual, onde se especifica e implementa os agentes, segundo uma abordagem baseada em conhecimento.

Nesse sentido, este capítulo busca, inicialmente, fazer uma revisão de literatura sobre Inteligência Artificial Distribuída (IAD) e Engenharia do Conhecimento. Além disso, busca-se determinar o estado da arte quanto às metodologias e/ou abordagens para a construção de SMA, e também para o desenvolvimento de Sistema Baseados em Conhecimento (SBC).

O capítulo é, então, assim organizado. A seção 2.2 descreve os conceitos relativos a IAD. A seção 2.3 faz o relato das principais metodologias para o desenvolvimento de SMA, além de uma análise crítica. A seção 2.4 apresenta os principais aspectos da Engenharia do Conhecimento, abordagem utilizada para o desenvolvimento dos agentes dentro da abordagem proposta neste trabalho. A seção 2.5 apresenta uma análise análoga à seção 2.3, entretanto, neste caso, no que se refere às metodologias de desenvolvimento de SBC's. Finalmente, a seção 2.6 apresenta as principais conclusões deste capítulo.

[1] c.f. item 1.1

## 2.2 Inteligência Artificial Distribuída

Numa análise histórica, afirma-se que a Inteligência Artificial evoluiu de uma metáfora *psicológica* à uma metáfora *sociológica*(Garcia e Sichman, 2003).

No início, sob a metáfora psicológica, a IA era concebida segundo modelos de inteligência baseados no *comportamento individual*, ou seja, modelos computacionais que descrevem como os seres humanos pensam, tomam decisões, planejam, percebem, se comunicam, etc. A partir disso, surgiram métodos de resolução de problemas, como a busca heurística em espaço de estados, planejamento de ações, aprendizagem de máquina, percepção, entre outros.

Sob essa visão, os sistemas desenvolvidos apresentavam em comum características de uma *concepção centralizada*, *não-reutilizável* e *não abertos ao exterior*.

No final da década de 70, a Inteligência Artificial assume a metáfora sociológica, inspirada em áreas como a lingüística, sociologia, economia, filosofia e biologia, surgindo a IA Distribuída. O objetivo é a compreensão e desenvolvimento de modelos e técnicas para a resolução da classe de problemas cuja distribuição, física ou funcional, é inerente. A inteligência passa a ser descrita a partir do *comportamento social*, ou, como descreve Garcia e Sichman (2003):

> "... como estabelecer modelos, arquiteturas e implementações para que um conjunto de entidades inteligentes possam executar ações de modo coordenado no seio de uma sociedade para que ao final se obtenha um comportamento global coerente."

### 2.2.1 Agente

O poder de um sistema construído sob a IAD está em suas entidades individuais, os *agentes*, e suas relações.

A definição mais citada de agente, dada a ausência de uma aceita universalmente, é apresentada por Ferber e Gasser (1991):

> "Um agente é uma entidade real, ou virtual, imersa em um dado ambiente onde ela pode executar 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".

Assim, graças ao seu alto grau de abstração, a noção de agente permite que se descreva um sistema em termos de ambiente, ações, dados perceptivos e objetivos, evitando considerações a respeito da forma como ele será implementado. Em outras palavras, antes de especificar como implementar um agente, deve-se garantir que ele possua características como (Hübner et al., 2004):

- **Percepção**: o agente é capaz de perceber mudanças no ambiente;
- **Ação**: o agente provoca mudanças no ambiente por meio de ações que devem necessariamente ser executadas no sentido de seus objetivos;
- **Comunicação**: a comunicação torna-se uma ação imprescindível para um agente uma vez que ele necessita coordenar suas ações com os demais agentes da sociedade;
- **Representação**: o agente precisa de uma representação interna daquilo que ele acredita ser verdade no mundo, considerando inclusive o conhecimento sobre os demais agentes da sociedade;
- **Motivação**: o agente necessita de uma representação interna dos seus objetivos dentro do ambiente, ou seja, dos estados que ele almeja alcançar.
- **Deliberação**: dada a motivação e a representação interna do ambiente, o agente deve ser apto a deliberar sobre as ações que levaram o ambiente do estado atual ao estado desejado.

Contudo, estas características são apresentadas por agentes em SMA cognitivos.

Além destes, existe ainda os SMA reativos, formados por agentes baseados em modelos de organização biológica ou etológica, como por exemplo as colméias, as sociedades de formigas e cupins. Nestes modelos, a força do sistema emerge da coordenação entre os agentes, que são extremamente simples. Agentes reativos tem um modelo de funcionamento do tipo estímulo-resposta, sem qualquer tipo de representação interna do ambiente, memória de suas ações, e planejamento de ações futuras. Normalmente, sociedades de agentes reativos são compostas por milhares de membros.

Por outro lado, os SMA cognitivos possuem normalmente poucos agentes em sua comunidade, uma vez que cada agente é um sistema complexo capaz de apresentar o conjunto de características definidas acima.

### 2.2.2 Subáreas da Inteligência Artificial Distribuída

Segundo Durfee e Rosenschein (1994), a IAD pode ser dividida em duas abordagens distintas: a Resolução Distribuída de Problemas (RDP) (do inglês *Distributed Problem Solving*), e os Sistemas Multiagentes (SMA).

Em uma abordagem sob RDP o sistema visa a solução um problema, e para isso ele é desenvolvido. O problema é decomposto em sub-problemas, e os agentes são desenvolvidos para a resolução destes sub-problemas, e a coordenação entre os membros da sociedade leva a composição da solução final. Via de regra, esta abordagem inviabiliza a reutilização destes agentes.

Em uma abordagem sob SMA, os agentes são considerados autônomos, no sentido de que possuem uma existência própria e objetivos próprios, independente do problema a ser resolvido. Desta forma, uma abordagem em SMA visa a concepção de mecanismos que garantam a interação e a organização de agentes, independente do problema a ser resolvido.

Apesar da distinção entre as abordagens propostas por Durfee e Rosenschein (1994), a literatura sobre IAD trata, com raras exceções, ambas como SMA.

### 2.2.3 Arquiteturas de Sistemas Multiagentes

Quanto à arquitetura, um SMA pode ser classificado em três tipos principais, não mutuamente exclusivos (Scalabrin et al., 1996):

- **SMA Federados**: caracteriza-se um SMA Federado pela presença de agentes especiais, denominados *facilitadores*, que se diferenciam dos demais agentes por possuírem conhecimentos específicos sobre os demais agentes da sociedade. Seu papel é o de orientar os demais agentes da sociedade quando estes necessitam interagir para alcançar seus objetivos. O facilitador dispõe de informações sobre a capacidade dos agentes e de seus endereços na sociedade. Sob esta arquitetura, tem-se, de um lado, a vantagem de um gerenciamento eficiente da comunicação entre agentes. Por outro lado, a centralização de algumas tarefas no facilitador torna o sistema dependente de seu bom funcionamento.

- **SMA Democráticos**: nesta arquitetura, todos os agentes estão no mesmo nível hierárquico. A comunicação é vista como uma ação assíncrona, e obedece às regras de alguma linguagem de comunicação, como o *KQML* (*Knowledge Query Manipulation Language*) (Finin et al., 1995). A vantagem desta arquitetura é a modularidade e flexibilidade obtida, porém tem-se a necessidade de um conhecimento específico a respeito das habilidades de cada agente.

- **SMA Abertos**: em um SMA aberto a composição na comunidade não é fixa, o que permite a entrada e saída dos agentes de forma dinâmica. A vantagem obtida neste tipo de sistema é a sua robustez, porém sob o custo de uma alta taxa de trocas de mensagens em um complexo protocolo de comunicação.

### 2.2.4 Aspectos Inerentes à Interação entre Agentes

Na medida em que um agente não age sozinho, o estado atual do ambiente não resulta apenas de suas ações, mas também das ações dos demais agentes. Por este motivo, alguns aspectos são inerentes à construção de um SMA. Entre estes aspectos estão a *comunicação*, a *coordenação* e a *organização* entre agentes.

#### Comunicação entre Agentes Cognitivos

Independente do tipo de SMA adotado, os agentes necessitam se comunicar para que possam alcançar os seus objetivos e os objetivos do sistema. Assim, independente do modelo de coordenação e organização adotado, a comunicação tem um papel central.

Segundo de Freitas e Bittencourt (2002), a comunicação direta entre agentes pode ser implementada segundo dois modelos: o modelo Cliente-Servidor, e o modelo *peer-to-peer*.

O modelo Cliente-Servidor, o menos utilizado, opera por meio de chamadas a procedimentos remotos. Internamente, o agente efetua uma comunicação do tipo pedido-resposta com os parâmetros do procedimento solicitado.

O modelo *peer-to-peer* baseia-se na *Teoria dos Atos de Fala* (Austin, 1962). Segundo esta teoria, a linguagem humana é uma forma de ação no ambiente em que o agente está inserido. Esta ação é classificada segundo a *performativa* da sentença, ou seja, a conseqüência que ela gera no mundo. Estas performativas classificam-se em (Bordini et al., 2001): *assertivas* (informar), *diretivas* (pedir ou consultar), *comissivas* (prometer ou comprometer-se), *proibitivas*, *declarativas* (causar eventos para o próprio comunicador) e *expressivas* (emoções).

Para uma comunicação em nível de conhecimento, são necessários dois componentes básicos. O primeiro é a intenção *pragmática* da mensagem, definida pelo ato de fala, e que deve fazer parte do protocolo de comunicação. O segundo componente é o conteúdo semântico da mensagem, cuja compreensão por parte dos agentes é possível pela definição de um formalismo de representação de conhecimento.

A intenção pragmática de uma mensagem, ou seja, o ato de fala empregado na comunicação deve fazer parte do *protocolo de comunicação* do SMA. Nesse sentido, um protocolo normatiza as múltiplas interações comunicativas entre os agentes por meio de seqüências padrões de troca de mensagens. Entre os principais protocolos estão a Rede Contratual (do inglês *Contract Net*) (Smith, 1980)

e o Protocolo de Negociação por Atos de Fala (SANP) (do inglês *Speech Act Negotiation Protocol*) (Chang e Woo, 1991).

Além de um protocolo, a interação no nível de conhecimento dos agentes depende também de uma linguagem de comunicação. Uma linguagem consiste em um conjunto de primitivas conhecido pelos agentes e um conjunto de regras de conversação. As primitivas são os atos de fala, enquanto as regras regulamentam as atitudes adotadas pelos agentes durante a comunicação. Entre as linguagens mais utilizadas está a KQML (Finin et al., 1995), que propõe um formato para as mensagens a serem trocadas e um protocolo de manipulação das mensagens compartilhadas.

**Coordenação em SMA**

A *coordenação* é uma atividade inerente a qualquer SMA, e deve ser definida, entre outros motivos, para: decidir a ordem de execução das ações, qual agente irá realizar qual ação, como os agentes irão trocar as informações sobre o resultado da execução das mesmas, como terão acesso a recursos escassos, como irão eventualmente alterar a prioridade de suas ações em função da ação dos outros.

Segundo Malone e Crowston (1994), coordenação pode ser definida como o ato de gerenciar dependências entre atividades. Estas dependências podem variar de acordo com o volume de atividades executados em um mesmo ambiente.

Uma cooperação eficaz entre os agentes em um SMA incrementa a qualidade das soluções geradas pela comunidade, bem como aumenta o desempenho da atuação dos agentes na resolução de tarefas.

Segundo Ferber (1999), existem quatro mecanismos de coordenação de ações em SMA. São eles:

- **Sincronização**: a coordenação de ações dá-se por seqüências de ações que devem ser sincronizadas em algum momento da execução. Sincronizar ações define uma forma de encadeá-las para que sejam realizadas em um mesmo instante de tempo.
- **Planejamento**: implementado em três fases: (i) determinação do conjunto de ações que devem ser realizados no sentido do objetivo global, por meio de um conjunto de planos; (ii) sincronização dos planos; (iii) escolha de um dos planos para execução. Os planos podem ser monitorados durante sua execução para verificação do estado do ambiente e os objetivos do sistema.

- **Reatividade**: consiste na reação do agente às modificações do ambiente e na adaptação de suas ações às ações dos outros agentes.

- **Regulamentação**: a coordenação é baseada em leis ou convenções sociais. Busca-se utilizar regras de comportamento para eliminar possíveis conflitos. Um exemplo são regras de prioridade para processos em execução.

Em Frozza e Alvares (2001) citam-se algumas aplicações que utilizam os mecanismos de coordenação acima. É importante ressaltar que um mesmo SMA pode utilizar mais de um mecanismo, de acordo com o problema a ser tratado.

### Organização em SMA

O propósito principal de uma *organização* em um SMA é fazer com que a finalidade do sistema seja facilmente alcançada.

Mesmo que pareça fácil, diferenciar um sistema organizado de um não organizado, definir o que é uma organização, como ela se constitui, quais as suas estruturas e mecanismos, não é uma tarefa fácil. Segundo Dignum e Dignum (2001), a organização de um SMA é um conjunto de restrições definidas a um conjunto de agentes de forma que eles ajam segundo uma finalidade comum. Estas restrições têm por objetivo controlar a autonomia dos agentes buscando produzir um comportamento global direcionado a uma finalidade, e podem ser de ordem estrutural, na forma de papéis, ou funcional, na forma de planos.

Segundo Conte e Castelfranchi (1992), existem duas classes de organizações, classificadas segundo as interações sociais empreendidas pelo sistema:

- **Modelos Estáticos ou Descendentes**: O sistema é concebido a partir da pressuposição de um problema, e os agentes são projetados para resolvê-lo. As restrições da organização servem para orientar os agentes no sentido dos objetivos do sistema.

- **Modelos Dinâmicos ou Ascendentes**: Neste tipo de organização, não existe um problema global a ser resolvido. Os agentes interagem socialmente e se organizam dinamicamente, para atingir seus próprios objetivos.

Os modelos dinâmicos podem ser decompostos ainda em duas subclasses:

- **Modelos Baseados na Utilidade**: baseado na Teoria de Jogos, afirma que os agentes devem coordenar suas ações para que obtenham um comportamento global coerente. Entretanto, a existência de múltiplos agentes em um ambiente limita a autonomia e o poder da ação de cada agente.

- **Modelos Baseados na Complementaridade**: neste modelo a organização do sistema é baseada na complementaridade de capacidades entre os agentes, em relação às ações e aos recursos disponíveis no ambiente. Este tipo de modelo de organização aumenta a autonomia e poder dos agentes, uma vez que mesmo que um agente não possa alcançar seus objetivos sozinho, os demais podem auxiliá-lo.

Uma ontologia de organização descreve seus elementos formalmente segundo suas prioridades, relações, restrições, e comportamentos. Assim, os elementos de uma organização são (Garcia e Sichman, 2003):

- **Organização**: Uma organização é formada por divisões e subdivisões, um conjunto de agentes alocados para estas divisões, um conjunto de papéis que estes agentes assumem e um conjunto de metas.

- **Papel**: são protótipos de funções que são atribuídas aos agentes na organização. Cada papel possui uma série de propriedades, entre elas: o seu conjunto de metas, processos que permitem alcançar as metas, permissões, habilidades necessárias, restrições na execução do processo, recursos necessários.

- **Agente**: membro de uma das divisões da organização, que pode assumir um ou mais papéis e que pode se comunicar com os demais agentes segundo as restrições da organização.

A literatura sobre organização em SMA apresenta uma série de modelos que variam segundo a classe de organização a ser implementada (estática ou dinâmica), e a política de restrições adotadas (estrutural ou funcional) (Hübner e Sichman, 2003).

No modelo AALAADIN (Ferber e Gutknecht, 1998), a organização é definida por um conjunto de grupos segundo uma determinada estrutura. A estrutura de um grupo é descrita pelos papéis que ele deve cumprir e pelos agentes membros. Assim, os papéis são representações abstratas das funções que devem ser exercidas pelos agentes. O AALADIN não faz qualquer restrição quanto a arquitetura interna dos agentes. Desta forma, um agente é visto apenas como uma entidade ativa e comunicativa que assume papéis nos grupos onde é membro.

O modelo TÆMS (Decker, 1996) é um modelo organizacional funcional no qual a função central é a de *tarefa*. Assim, o objetivo do modelo é descrever a estrutura de tarefa de modo a viabilizar a análise e a simulação da organização. Para isso, as tarefas são abordadas segundo três pontos de vistas diferentes. A visão *objetiva* considera a tarefa em sua estrutura completa, aquele que resolve um problema em um determinado período de tempo. A visão *subjetiva* é a tarefa vista pelo ângulo dos agentes, ou seja, a parte da execução que lhes cabe, segundo as restrições de organização do sistema. A visão *generativa* é aquela que contém as informações para a geração das visões objetivas e subjetivas de cada tarefa para a resolução dos problemas de um dado domínio. Embora a representação do TÆMS não tenha o objetivo de gerar a coordenação entre as tarefas, este tipo de informação pode ser gerado a partir do modelo, mais especificamente, por uma estrutura específica da visão generativa.

O modelo organizacional $\mathcal{M}$ oise$^+$ (Hübner et al., 2002) apresenta uma visão centrada na organização, ou seja, a organização existe objetivamente e possui uma descrição própria, e considera três formas de representar as restrições de uma organização: a estrutura (papéis), o funcionamento (planos globais) e as normas da organização. Este último aspecto estabelece as relações entre a estrutura e as funções, definindo as responsabilidades dos papéis nos planos globais. Assim, o problema de encontrar uma boa organização pode ser formulado em termos de uma busca no espaço de comportamentos. Este espaço de comportamentos mapeia as observações do ambiente em ações.

## 2.3 Metodologias para desenvolvimento de Sistemas Multiagentes

Nesta seção o objetivo é apresentar as principais metodologias para o desenvolvimento de SMA. Esta análise serve de referencial teórico para o nível social da abordagem proposta no capítulo 4.

Uma metodologia de desenvolvimento de SMA é constituída por uma série recomendada de etapas e procedimentos, que compõem os métodos que devem ser seguidos durante o processo de concepção do sistema. Esta metodologia deve capturar a flexibilidade, autonomia e o poder que a abstração de agentes incorpora ao projeto e desenvolvimento de sistemas complexos. Além disso, deve também auxiliar o projetista em tomadas de decisão sobre aspectos relativos à análise, projeto e implementação (Dastani et al., 2004).

O poder deste tipo de sistema e a facilidade obtida com o desenvolvimento de metodologias específicas para estes cenários, extrapolou os limites da IA. Atualmente, mais do que uma solução para Engenharia do Conhecimento, a abstração destas metodologias para SMA tornaram-se paradigmas para a Engenharia de Software, denominada Programação Orientada a Agentes (Nwana, 1995).

Assim, do ponto de vista da construção de programas, as metodologias que suportam o paradigma de agentes tornam-se atrativas, dentre outros motivos, devido ao conceito de um agente como um sistema autônomo, capaz de interagir com outros sistemas para satisfazer seus objetivos. Ou seja, um agente é visto como um sistema computacional genérico, que cumpre as metas e os papéis necessários ao sistema (de Brito et al., 2001).

A seguir são apresentadas algumas destas metodologias.

### 2.3.1 Gaia

Gaia (Wooldridge et al., 2000) é uma metodologia para análise e projeto orientado a agentes cujo objetivo é maximizar parâmetros globais de um sistema. Para um SMA construído pelo Gaia, agentes são sistemas computacionais genéricos, que fazem uso de recursos do ambiente. Desta forma, agentes podem ser vistos como processos de um sistema UNIX.

Quanto às restrições, Gaia pode ser aplicada a SMA heterogêneos, no sentido de que os agentes podem ser implementados usando diferentes linguagens de programação, arquiteturas e técnicas. A estrutura da organização do sistema é estática, já que as relações inter-agentes não mudam durante a execução. Além disso, as habilidades dos agentes, bem como os serviços providos por eles, também não se modificam.

Mesmo que a implementação do sistema não seja direta, Gaia busca uma especificação suficientemente detalhada para que este problema seja contornado.

A metodologia é estruturada em torno de uma série de modelos que estabelecem a relação entre três etapas distintas: especificação de requisitos, análise e projeto. Esta estrutura pode ser visualizada na figura 2.1.

As entidades de uma especificação em Gaia são de dois tipos: *abstratas* e *concretas*. Entidades abstratas são aquelas utilizadas durante a análise para a conceitualização do sistema, mas que não possuem uma implementação direta. É o caso de conceitos como papéis, permissões, responsabilidades, protocolos, entre outros. Por outro lado, entidades concretas são usadas durante a fase de projeto, e terão uma representação direta no sistema em execução. São conceitos como tipos de agentes, serviços e relações.

Na fase de análise, o objetivo é especificar a estrutura do sistema, independente de aspectos relativos a sua implementação. Esta estrutura captura a organização do sistema, ou seja, o conjunto

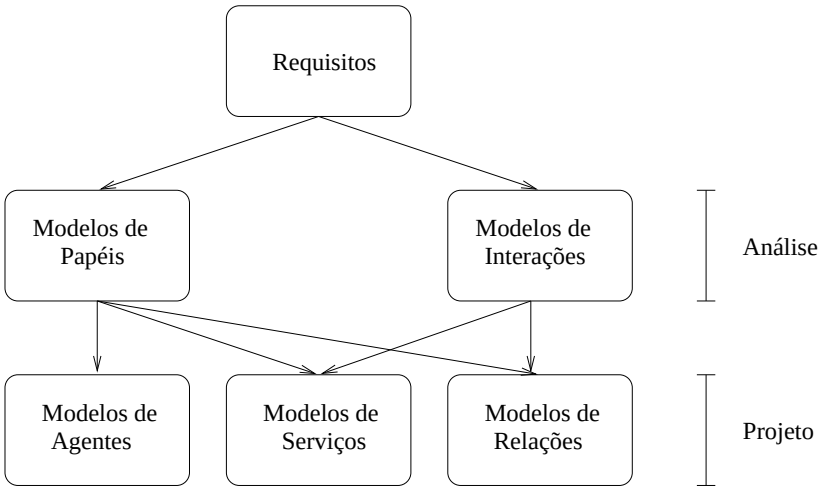


**Figura 2.1:** *Estrutura do Gaia*

de papéis que devem ser executados. Assim, o conceito mais abstrato é o de *sistema*, que tem relação semântica com a idéia de sociedade.

Ainda estruturando um sistema computacional de forma hierárquica, em um próximo nível está o conceito de *papel*. Uma papel é definido por quatro atributos: *responsabilidades*, *permissões*, *atividades* e *protocolos*. As responsabilidades determinam as funcionalidades de um papel, que em conjunto com as permissões, definem os recursos que podem ser utilizados para implementá-las. As atividades são as ações que o agente pode executar sem o auxílio de outros agentes, enquanto os protocolos definem as relações entre os diferentes papéis.

Assim, a fase de análise é formada pela especificação de dois modelos: o *modelo de papéis*, conforme a estrutura acima, e o *modelo de interações*. Este último, define a forma como os papéis se relacionam, ou seja, especifica o protocolo entre os papéis.

A fase de projeto envolve a construção de três modelos. O *modelo de agente* identifica os tipos de agentes que formarão o sistema. O *modelo de serviço* identifica os principais serviços que serão realizados pelo papel do agente, enquanto o *modelo de relação* especifica a comunicação entre os diferentes tipos de agentes.

A grande limitação de Gaia está na sua incapacidade de transformar os modelos abstratos derivados de sua especificação em modelos de baixo nível suficientes para uma implementação automática do sistema. Além disso, prevê apenas a especificação de sistemas centralizados na organização, o que limita a sua capacidade de reutilização. Gaia não conta também com estruturas gráficas capazes de permitir a visualização do sistema global e os protocolos envolvidos, o que facilitaria um processo de verificação de consistência.

### 2.3.2 TROPOS

Tropos é uma metodologia de engenharia de software, em nível de conhecimento, para a programação orientada a agentes (Bresciani et al., 2001), fundamentada em duas características chaves: a noção de agente e suas características cognitivas são usadas em todas as fases do desenvolvimento, e o papel crucial dado à fase de análise de requisitos que precede a especificação.

Para o Tropos, mais do que desenvolver um SMA, desenvolve-se um programa, em cinco fases.

Na etapa de *requisitos primários* procura-se compreender o problema por intermédio de um cenário organizacional. A saída deste estágio é um modelo de organização que inclui os atores principais e suas respectivas dependências. Esses atores são caracterizados por possuírem metas, que de forma isolada, são incapazes de alcançar. Esta metas são alcançáveis devido ao conhecimento do sistema e de suas dependências. Os atores possuem uma certa relação com os papéis de Gaia.

A descrição do sistema a ser construído dentro de seu ambiente operacional é feita na etapa de *requisitos secundários*, juntamente com as funções e habilidades necessárias. Esta descrição modela o sistema por um conjunto de atores, que possuem um certo número de dependências sociais com outros atores do ambiente.

A etapa de *projeto de arquitetura* define a arquitetura global do sistema por meio de subsistemas interconectados por dados e fluxos de controle. Subsistemas são representados por atores enquanto interconexões de dados e controle correspondem as suas dependências. Além disso especifica-se as capacidades dos atores e dos tipos de agentes, finalizando com a especificação dos agentes do sistema.

Na etapa de *projeto detalhado*, cada agente do sistema é definido em termos de seus eventos internos e externos, planos, crenças e protocolos de comunicação.

Finalmente, na fase de *implementação*, o sistema é construído na linguagem de agente JACK JACK (2005), uma plataforma de programação de agentes baseada na arquitetura BDI, de acordo com a fase de projeto detalhado.

Uma vantagem da metodologia Tropos é permitir a implementação automática dos agentes que compõem o sistema. Isto só é possível uma vez que o Tropos faz restrições quanto a arquitetura destes agentes, que deixam de ser caixas pretas, como no caso do Gaia, vistos apenas pelas relações entrada/saída. Assim, permite-se implementar os agentes segundo a arquitetura BDI. Por outro lado, ao restringir a implementação sob esta arquitetura, o SMA está restrito a problemas para os quais soluções do tipo BDI sejam aplicáveis. Não obstante, as etapas da metodologia não têm uma relação

direta com os conceitos de uma organização SMA, como papéis, metas, ações, que só são considerados já na etapa de especificação dos agentes. Assim como Gaia, Tropos não possui qualquer tipo de verificação em qualquer das etapas de desenvolvimento.

### 2.3.3 ZEUS

ZEUS (Nwana et al., 1999) é uma uma ferramenta de engenharia de software que utiliza a abordagem multiagente de cooperação em nível de conhecimento e cujo objetivo é a construção de sistemas com características de interoperabilidade, escalabilidade, e reconfigurabilidade. A filosofia de ZEUS é facilitar o rápido desenvolvimento de aplicações de agentes colaborativos por meio de uma biblioteca de componentes de agentes, e um ambiente de suporte. O resultado final é a geração de parte do código fonte dos agentes em Java.

ZEUS utiliza uma arquitetura de SMA Federado, cujo facilitador possui informações a respeito do endereço e do conjunto de ações de cada agente do sistema. Segundo ZEUS, uma metodologia deve abordar aspectos como serviços de informação - cujo papel é desempenhado pelo facilitador - comunicação, ontologias, coordenação, e integração de softwares.

A ferramenta ZEUS é formada por um conjunto de componentes, escritos em Java, classificados dentro de três grupos funcionais: uma biblioteca de componentes de agentes, uma ferramenta para a construção dos agentes e uma suite de utilitários relativos aos agente facilitador.

A biblioteca de componentes de agentes é uma coleção de classes que formam os blocos para a construção de agentes. Segundo a filosofia de ZEUS, esta biblioteca tem o objetivo de implementar o nível de agente de tal modo que ele seja independente da aplicação. Os assuntos abordados por esta biblioteca incluem comunicação, ontologia e coordenação social. A comunicação é implementada em KQML, segundo um sistema de passagem de mensagens baseado em sockets assíncronos. A ontologia é descrita por um editor específico, de acordo com uma representação de conhecimento em quadros, para a representação dos conceitos do domínio. A coordenação é provida por um sistema de escalonamento e planejamento para aplicações orientadas a tarefas, e um motor de coordenação que controla o comportamento social de um agente, ou seja, quando ele interage com outros agentes e de que forma.

Segundo a ferramenta para a construção de agentes, em seu mais alto nível de abstração, um agente ZEUS é composto por três camadas: uma *camada de definição*, uma *camada de organização* e uma *camada de coordenação*. Na camada de definição, o agente é visto como uma entidade autônoma, segundo suas competências, modelo racional, recursos, crenças e preferências, entre outras

propriedades. Na camada de organização, o agente é definido segundo suas relações com os outros agentes, tratando questões como quais agentes ele conhece, quais as habilidades destes agentes conhecidos, etc. Na camada de coordenação o agente é visto como uma entidade social, em termos de suas técnicas de coordenação e negociação, abordando assuntos como protocolos e comunicação.

Desta forma, a ferramenta espera resolver assuntos relativos à análise do domínio, a definição dos agentes, das tarefas, a organização e coordenação.

A suíte de utilitários é voltada para as questões envolvendo o desenvolvimento do agente facilitador. Este agente é construído de forma a centralizar o conhecimento da arquitetura do SMA.

Ao fazer a escolha por uma arquitetura SMA federada, um sistema concebido por meio do ZEUS está sujeito ao bom desempenho de seu facilitador. Não obstante, ZEUS faz muitas restrições quanto aos mecanismos de seu sistema, como o tipo de linguagem, os mecanismos de coordenação, a ontologia, o que não resulta necessariamente na total automação do processo de implementação dos agentes, já que somente parte do seu código é gerada. Assim, o ZEUS não pode ser classificado como uma metodologia para SMA, mas sim como uma ferramenta específica para a construção um tipo restrito de sistema.

### 2.3.4 AUML

AUML (Odell et al., 2000) não deve ser vista como uma metodologia completa para o desenvolvimento de SMA, como Gaia, ZEUS, ou Tropos. AUML limita-se a especificação das diferentes relações existentes dentro do sistema. Dentre estas relações citam-se os protocolos de comunicação, o fluxo de informação interna do agente, ou ainda diagramas de colaboração. Deste modo, AUML não deve ser utilizado isoladamente, mas sim, juntamente com outra metodologia.

AUML é uma extensão de UML (*Unifield Modeling Language*) utilizado para a especificação de sistemas sob orientação a objeto. Desta forma, para AUML, o paradigma de orientação a agentes é uma extensão da orientação a objetos, uma vez que este último é incapaz de capturar toda a semântica de um agente. Esta hipótese é sustentada na literatura, uma vez que existem um grande número de metodologias que utilizam as técnicas de orientação a objeto, estendendo ou adaptando às características específicas do paradigma de agentes. Gaia e ZEUS também utilizam esta abordagem.

Para AUML, um protocolo como um todo é uma entidade, formado por sub-entidades. Os diagramas de seqüências descrevem as transações inter-agentes para implementar o protocolo. Finalmente, as atividades intra-agentes completam a especificação do protocolo. Assim como em Gaia, o

objetivo é especificar o sistema até o nível mais baixo possível, onde ele possa estar suficientemente adequado para o desenvolvimento ou geração de código.

A descrição do parágrafo acima define as etapas de especificação do sistema por AUML. No nível 1, define-se a representação do protocolo geral. No nível 2, definem-se as interações entre os agentes, enquanto no nível 3, especifica-se a representação interna do processamento dos agentes. Assim, vê-se que a representação em AUML é voltada para sistemas centrados na organização, ou seja, o sistema é desenvolvido para um problema específico a ser resolvido.

O fato de AUML partir de uma adequação do paradigma de agentes à orientação a objetos torna-a uma técnica não trivial quando deve tratar de conceitos básicos de SMA como papéis, responsabilidades e organização. Eles são concebidos, porém de forma não explícita.

Assim como Gaia, AUML é limitado quanto à especificação interna dos agentes, restringindo-se à especificação do conjunto de eventos que devem ser considerados. Conseqüentemente, o resultado final fica longe da implementação do sistema.

### 2.3.5 MaSE

MaSE (*Multiagent Systems Engineering*) (Wood e DeLoach, 2001) é uma metodologia criada para o desenvolvimento de sistemas que buscam a coordenação do comportamento local de agentes, para prover um comportamento apropriado para o SMA como um todo. O MaSE é independente de arquitetura SMA, linguagem de programação, e protocolos e linguagens de comunicação. Uma descrição do progresso de desenvolvimento do MaSE é apresentada na figura 2.2.

Numa primeira observação pode-se concluir que o processo de análise dá-se pela especificação das questões sociais do sistema, enquanto que o projeto trata da especificação das entidades que constituem o sistema.

A primeira fase é destinada a capturar as meta do sistema, tomando uma especificação inicial e a transformando em um conjunto estruturado de metas. Para MaSE, uma meta sempre é definida no contexto do sistema. Construções de baixo nível podem até ser responsáveis por algumas metas, mas estas sempre estarão no nível do sistema.

Na segunda fase, são aplicadas construções denominadas Casos de Uso, para a construção das conversações entre agentes, que permitem a distribuição do problema. Casos de Uso são descrições narrativas de uma seqüência de eventos que definem o comportamento desejado do sistema. Nesta fase, capturam-se Casos de Uso dos requisitos iniciais do sistema, reestruturando-os como Diagramas

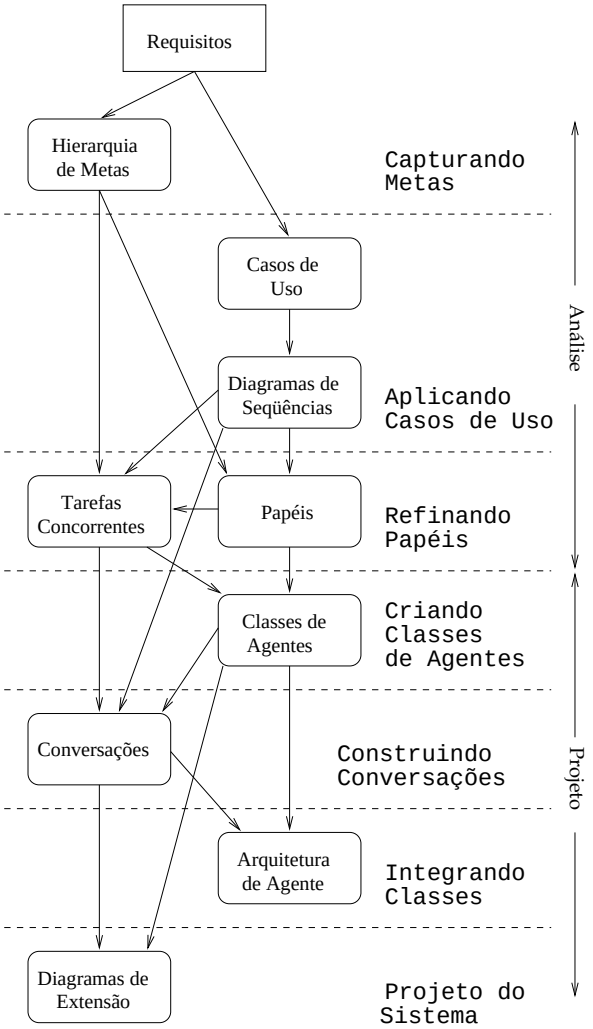


**Figura 2.2:** *O MaSE*

de Seqüências. Um Diagrama de Seqüência é usado para determinar o conjunto mínimo de mensagens que devem ser passadas entre os papéis.

Na última fase da análise, transformam-se as metas estruturadas da primeira fase em papéis. Para o MaSE, papéis são os blocos básicos para a definição das classes de agentes, englobando as metas do sistema durante a fase de projeto. Associado aos papéis, estão as tarefas que devem ser executadas para realização da meta.

A primeira fase da etapa de projeto é voltada para a identificação das classes de agentes que implementaram as metas do sistema. O resultado desta fase é um Diagrama de Classes de Agentes que descreve as classes especificadas e as interações entre elas.

As conversações entre as classes devem suportar e ser consistentes com todos os diagramas de seqüência derivados das fases anteriores.

Na fase de integração das classes, aspectos internos dos agentes são tratados. Para isso, deve-se escolher a arquitetura que melhor se adapta às necessidades do problema, o que MaSE deixa em aberto.

Na fase final de projeto, MaSE instancia as classes de agentes em agentes reais. Para isso, ele utiliza um Diagrama de Extensão que mostra o número, os tipos, e as localizações dos agentes dentro do sistema.

Quanto ao resultado obtido com MaSE, ele é muito similar ao obtido com Gaia, na medida em que seu resultado final são os requisitos para a construção do código que gerará os agentes do sistema. Seu ponto forte está na utilização de mecanismos gráficos para todas as etapas do processo. Entretanto, cada fase possui seu próprio mecanismo. O MaSE ainda dispõe de ferramentas para a verificação dos protocolos empregados para as conversações.

### 2.3.6 Prometheus

Prometheus (Padgham e Winikoff, 2002) é uma metodologia para o desenvolvimento de agentes inteligentes voltada para usuários que não possuem conhecimento específico na área de agentes.

A metodologia consiste em três fases principais: *especificação do sistema*, onde se identifica as funcionalidades básicas do sistema, *projeto da arquitetura* que usa a o resultado da fase anterior para determinar que agentes o sistema conterá e como eles interagirão, e *projeto detalhado* que trata da especificação interna do agente.

O resultado final deste processo é a implementação dos agentes na linguagem de programação escolhida, baseado em uma arquitetura BDI. Deste modo, Prometheus sacrifica a generalidade da arquitetura final pela possibilidade de implementação automática dos agentes do sistema.

Na fase de especificação do sistema, as funcionalidades do SMA são definidas segundo seu conjunto de percepções e ações. Estas funcionalidades podem ser vistas como os papéis necessários à resolução dos problemas do ambiente. O ideal é que estas funcionalidades sejam refinadas o suficiente para tratarem com aspectos simples, como uma única meta. A definição de uma funcionalidade dá-se por um descritor, cujos atributos são *nome*, uma *descrição* em linguagem natural, uma lista de *ações*, uma lista das *percepções relevantes*, e uma breve descrição das *interações* com outras funcionalidades. A dinâmica entre as funcionalidades é descrita por casos de uso (cenários), onde cada instância é descrita por um *identificador*, uma *descrição* em linguagem natural, um campo de *contexto* que indica quando este cenário ocorre, e uma lista de *variações* para os cenários.

Na fase de projeto da arquitetura, funcionalidades são atribuídas a grupos de agentes, segundo critérios de coerência e acomplamento. Isso envolve, portanto, análises de prós e contras referentes a agrupamentos de determinadas funcionalidades. Numa etapa seguinte, os agentes são definidos por um descritor, similar ao descritor de funcionalidades, onde definem-se *nome*, a *descrição* em linguagem natural, as *funcionalidades* incluídas, *dados* utilizados, e as possíveis *interações* com outros agentes. A consistência da descrição pode ser feita por dois tipos de diagramas: um para a descrição do sistema global, e outro específico para as interações entre os agentes.

A fase de projeto detalhado volta-se para o desenvolvimento da estrutura interna dos agentes, definindo como se executarão suas tarefas no sentido das metas do sistema. Estas especificação interna é constituída pelos planos definidos pelo usuário, que serão implementados pelas especificações das crenças, desejos e intenções dos agentes (arquitetura BDI), implementadas em alguma linguagem de agente. O foco da etapa está na definição da capacidade dos agentes, seus eventos internos, planos e estruturas de dados.

O ponto forte da metodologia é a capacidade de gerar, ao final do processo, os agentes em nível de implementação. Entretanto, para que isso seja possível, o Prometheus restringe-se a uma especificação interna segundo a arquitetura BDI, nem sempre a melhor solução para alguns problemas. Além disso, a metodologia faz uso de múltiplas ferramentas para checagem, sem que exista uma relação formal explícita entre elas. Este tipo de solução exige que o usuário seja capaz de manipular diferentes ferramentas, sem que isso leve a uma verificação do sistema global, mas apenas dos aspectos específicos para os quais as ferramentas foram projetadas.

### 2.3.7 Síntese Comparativa entre as Metodologias para Sistemas Multiagentes

De forma geral, todas as metodologias derivam de extensões da orientação a objetos, de forma a compreender questões como autonomia, metas e protocolos de comunicação. Esta adaptação é necessária uma vez que o conceito de objeto é insuficiente para abranger a complexidade das principais propriedades de agentes. O exemplo mais evidente desta abordagem é AUML (Odell et al., 2000).

Para AUML, conceitos como metas, planos e papéis devem ser expressos de forma indireta por intermédio dos diferentes níveis de relações que ela pode expressar, sejam protocolos ou diagramas de seqüências.

Por este motivo, seria interessante que uma metodologia e/ou abordagem orientada a agentes, desenvolvida para esta paradigma, fosse capaz de representar conceitos relativos a SMA de forma direta, e que pudesse fazer parte da análise de requisitos em processos de verificação nas diferentes etapas do projeto.

Outra característica comum às metodologias apresentadas é a descrição das interações entre os agentes por meio de formalismos baseados em diagramas de seqüências. Estes diagramas devem abordar, necessariamente, aspectos relativos à protocolos e linguagens de comunicação. O resultado desta abordagem é a construção de vários diagramas, tornando o modelo de interação muito vezes incompreensível para o projetista, de acordo com o aumento do número de agentes. Um exemplo deste tipo de abordagem pode ser vista no MaSE, em que são necessárias duas etapas para descrição das interações entre os agentes: uma análise de Diagramas de Seqüências e projeto da Conversações.[2]

Uma forma interessante de simplificar o processo de especificação seria tratar as interações entre os agentes como um problema de coordenação. Neste tipo de solução, as interações são especificadas por mecanismos de coordenação, como planejamento por exemplo, e todas as questões envolvendo comunicação (linguagens, protocolos, etc.) derivariam de forma automática desta especificação. Assim, tudo o que a especificação precisa estabelecer é um planejamento global, coordenando as ações dos agentes, segundo seus papéis e aptidões.

Uma característica muito presente nas metodologias são estruturas gráficas para auxiliar na especificação das múltiplas etapas do projeto. Estas ferramentas permitem verificações da consistência quanto aos requisitos da etapa em desenvolvimento, seja de maneira formal ou informal. Exemplos delas podem ser vistas em Prometheus, MaSE e ZEUS.

Mais do que ferramentas gráficas, seria interessante a estruturação do sistema global em torno

[2] c.f. figura 2.2

de um formalismo único, poderoso o suficiente, para auxiliar na especificação, verificação e implementação do sistema.

Uma questão importante, presente em todas as metodologias, é a decisão quanto à generalização ou restrição da arquitetura interna dos agentes que compõem a sociedade em desenvolvimento. Uma vez que se opte pela generalização da arquitetura, o que Lind (2001) define como *noção fraca de agente*, o resultado final de sua aplicação é um conjunto de requisitos que devem ser atendidos para a implementação do agentes. Estes, por sua vez, podem ser implementados por qualquer arquitetura ou linguagem de programação, de acordo com a afinidade e habilidades do projetista do sistema. Este é o caso de Gaia e de AUML. Por outro lado, ao se fazer restrições quanto a arquitetura interna do agente e/ou linguagens de programação a serem utilizadas, fica-se mais próximo do nível de implementação do sistema. Esta visão, Lind (2001) define como uma *noção forte de agente*.

Neste caso, a barreira a ser vencida é a unicidade de soluções que restringem a estrutura interna dos agentes à arquitetura BDI e as diversas linguagens que a implementam, como JACK e Jason (Hübner et al., 2004). Uma alternativa seriam restrições a arquiteturas mais gerais, como aquelas baseada em conhecimento. Assim, a metodologia e/ou abordagem seria capaz de especificar o agente no nível de conhecimento para sua implementação, deixando aspectos relativos a sua implementação, a ferramentas específicas.

Sendo assim, um passo adiante quanto a metodologias para desenvolvimento de SMA seria uma concepção segundo o paradigma de orientação a agentes, onde conceitos específicos a esta abordagem sejam representados de forma direta e explícita. Para isso, é importante a utilização de um formalismo único e poderoso, capaz de expressar os múltiplos níveis de concepção, social e individual, do sistema. Este formalismo deve ainda especificar as interações em um nível de coordenação que permite deduzir formalmente a comunicação necessária para sua implementação, independente de linguagens e protocolos. No nível de estruturação interna do agente, para que se possa chegar o mais próximo possível do seu nível de implementação final, propor-se-ia restrições de arquiteturas baseadas em conhecimento, como uma alternativa a arquitetura BDI, menos geral, e por vezes, mais complexa.

Dado estes requisitos, a abordagem proposta está de acordo com Dastani et al. (2004), quando afirma que:

> "...Além de diretrizes para as fases de análise e projeto, uma metodologia deve também prover diretrizes para a fase de implementação, e explicar como estes conceitos podem ser mapeados em instruções de uma linguagem de programação disponível."

## 2.4 Engenharia de Conhecimento

Ao considerar uma abordagem baseada em conhecimento para a estruturação interna do agente, faz-se necessário uma revisão bibliográfica a respeito desta área. Nesse sentido, esta seção tem o objetivo de apresentar a área da IA relativa a este assunto, a Engenharia do Conhecimento.

### 2.4.1 Fundamentos

*Engenharia do Conhecimento* é o processo de aquisição e representação de conhecimento humano para avaliação e incorporação em um sistema computacional. Uma vez incorporado ao sistema, este conhecimento pode ser manipulado por processos simbólicos dentro da base de conhecimento para a resolução de problemas e outras aplicações do conhecimento humano. Este processo é empreendido por um *engenheiro de conhecimento*, que atua em conjunto com um *especialista* humano. Seu caráter multidisciplinar encontra fundamentos na ciência da computação, psicologia cognitiva, psicologia comportamental, entre outras matérias (Tuthill, 1989, cap. 1).

O engenheiro de conhecimento é aquele que investiga o domínio, determina o conjunto relevante de conceitos, e estabelece uma representação formal dos objetos e relações do domínio (Russel e Norvig, 1995, cap. 8). Para isso o engenheiro de conhecimento realiza uma série de entrevistas com um especialista do domínio, dentro de um processo denominado *aquisição de conhecimento*.

Um especialista, mais do que "saber", deve atuar efetivamente no domínio. Dado os tipos de conhecimento presentes em uma base de conhecimento, um especialista deve prover o "saber" do conhecimento *declarativo* e o "fazer" do conhecimento *procedural*. Segundo esta classificação, o conhecimento pode ser enquadrado em:

- *Conhecimento Declarativo*: é um representação descritiva do conhecimento. Trata-se de declarações factuais sobre objetos, ou seja, descreve "o que é" um determinado objeto.
- *Conhecimento Procedural*: é a habilidade cognitiva que permite-nos saber como fazer alguma coisa, ou seja, é a compreensão de um contexto pela descrição "como funciona".
- *Conhecimento Heurístico*: é aquele adquirido pela experiência no domínio, e tem um caráter altamente individual. Regras heurísticas, que formam este tipo de conhecimento, são vistas como estratégias para a resolução de problemas.
- *Conhecimento de Senso Comum*: é formado por um conjunto de conhecimentos gerais sobre o mundo que é compartilhado entre as pessoas. Pode ser expresso na forma declarativa ou

procedural. É aquilo que permite-nos um julgamento sobre o "certo e o errado". Dado a sua heterogeneidade, não é utilizado em SBC.

Uma boa base de conhecimento deve estar apta a representar qualquer um dos tipos acima, exceto o conhecimento de senso-comum. A natureza do problema, contudo, determina o tipo preponderante, e com ele, as fontes de informação que devem ser consultadas pelo engenheiro de conhecimento.

Independente do tipo, o conhecimento pode ainda ser estruturado em unidades básicas, que constituem o suporte para o desenvolvimento de uma base de conhecimento (Rezende et al., 2003). São elas:

- *Fatos*: relações arbitrárias entre objetos, símbolos, eventos, etc.
- *Conceitos*: resultam de idéias abstratas, de natureza hierárquica.
- *Regras*: conjuntos de operações e passos que orientam a ação, desenvolvidas a partir da análise de fatos e conceitos, correspondendo a aplicação do conhecimento.
- *Metaregras*: responsável pela criação e aplicação de novas regras e situações novas, correspondendo à geração do conhecimento novo.

### 2.4.2 Sistemas Baseado em Conhecimento

Sistemas Baseados em Conhecimento são definidos pela sua função e suas propriedades: são sistemas computacionais que apresentam uma clara distinção entre o conhecimento necessário para a execução de uma tarefa e os mecanismos de decisão envolvidos na mesma. Aplicam-se, sobretudo, a problemas em que uma grande quantidade de conhecimento especializado é necessária.

De modo geral, os SBC's são indicados para as seguintes classes de tarefas (Rezende et al., 2003, pag. 21):

- **Interpretação**: consiste na análise de um conjunto de dados onde o objetivo é uma determinação simplificada do seu significado.
- **Classificação**: consiste no processo de determinação de falhas em um sistema, dado um conjunto de sintomas. É aplicado principalmente em sistemas de diagnósticos médicos.

- **Monitoramento**: consiste no processo de observação contínua do comportamento de um sistema a fim de realizar ações quando alguma situação específica acontece.

- **Planejamento**: consiste na especificação de seqüências de ações que levam a realização das metas do sistema.

- **Projeto**: consiste no desenvolvimento das especificações de um objeto, seguindo os requisitos exigidos. Exemplos de aplicação incluem sistemas de consultoria para plataformas de hardware, projeto de *layout* de circuitos elétricos, entre outros.

### Componentes de um SBC

A necessidade de uma separação explícita entre os mecanismos de um SBC, indica dois componentes principais: a *base de conhecimento* e o *mecanismo de inferência*. Independente da aplicação à qual se destina, a presença destes componentes é condição necessária, porém não suficiente. Na sua formação mais básica, um SBC tem o aspecto descrito na figura 2.3.

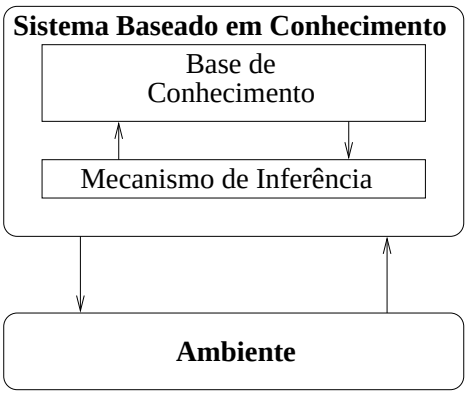


**Figura 2.3:** *Componentes básicos de um SBC*

A base de conhecimento é o componente central de um SBC. Informalmente, uma base de conhecimento é um conjunto de representações de fatos sobre o mundo. Cada representação individual é denominada *sentença*. Estas sentenças devem ser expressas de acordo com as regras de uma linguagem de representação de conhecimento. Eventualmente é possível a utilização de uma combinação de diversas técnicas de representação de conhecimento. SBC destes tipos são denominado sistemas híbridos com relação à representação de conhecimento (Russel e Norvig, 1995).

O mecanismo de inferência detém o controle de execução do sistema, viabilizando a manipulação das sentenças da base de conhecimento, derivando novas sentenças ou ações no ambiente. Nesse sen-

tido, processa a linguagem de representação de conhecimento, gerando e percorrendo o espaço de busca sempre que necessário. Dentre os métodos de raciocínio de um mecanismo de inferência estão:

- *Encadeamento Progressivo*: (do inglês "forward chaining") controle para estratégia de busca que emula o raciocínio dedutivo. Neste caso, o processo é viabilizado por um encadeamento de resoluções que tem origem no conjunto de fatos providos pelo usuário ou ambiente e tem fim na solução esperada. Este método é também conhecido como raciocínio dirigido por dados. Tem aplicação em problemas de planejamento e projeto.
- *Encadeamento Regressivo*: (do inglês "backward chaining") controle de estratégia de busca que emula o raciocínio abdutivo. Em cada processo, todas as soluções são consideradas inicialmente como verdadeiras. Porém, cada solução é decomposta até que se encontre aquela que melhor se adapte aos fatos de entrada. Por este motivo, é também denominado raciocínio dirigido por metas. É aplicável em problemas de diagnóstico, controle e monitoramento.

Em um Sistema Especialista, um tipo específico de SBC, a base de conhecimento é refinada em dois módulos: a *memória de trabalho* e a *base de regras*.

A memória de trabalho tem a atribuição de armazenar as condições iniciais de um ciclo de raciocínio, conclusões e decisões intermediárias, além de soluções finais. Desta forma, representa literalmente a "memória" do sistema.

A base de regras contém as condições que representam "perguntas" à representação de conhecimento da memória de trabalho e podem, portanto, ser de diferentes tipos. Em geral, envolvem instanciação de variáveis e eventualmente algum tipo de inferência.

A nova formulação de um SBC, contendo base de regras e memória de trabalho, é descrita pela figura 2.4.

**Formalismos para Representação de Conhecimento**

Uma Representação de Conhecimento (RC) é uma combinação de estruturas de dados e procedimentos de inferência que, se projetados de maneira adequada, levam o sistema a um comportamento inteligente (Barr e Feigenbaum, 1981). De fato , tais procedimentos de inferência podem ser compreendidos como uma interpretação das estruturas de dados, que levam em consideração o domínio representado. Em outras palavras, um comportamento é dito inteligente se suas estruturas de dados possuem uma semântica em relação ao contexto do problema.

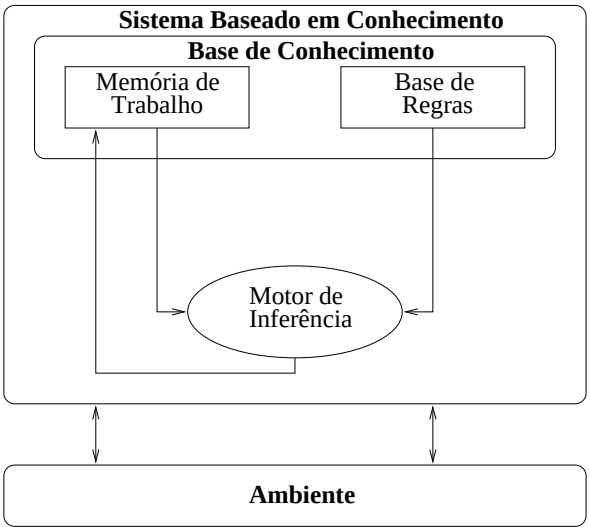


**Figura 2.4:** *Sistema Especialista*

Uma RC é o padrão estabelecido no nível lógico do sistema que permite a descrição do nível de conhecimento. Desta forma, observa-se uma clara distinção entre o conhecimento e a estrutura que o representa. Portanto, uma representação deve ser avaliada não pelo conhecimento que ela descreve, mas como ela descreve.

A escolha dos métodos de RC tem fundamental importância no desempenho do SBC. Esta escolha é feita na tentativa de dar a uma base de conhecimento características como expressividade, concisão, ausência de ambiguidade, eficiência, clareza e correção. A seguir, apresentam-se os principais formalismos de representação de conhecimento encontrados na literatura.

**Lógica** Dado o seu rigor formal, a lógica é uma alternativa interessante, não só como mecanismo de RC, mas como um formalismo de inferência completo. Este rigor - ausente na linguagens ditas naturais, como o português, passíveis de regras gramaticais imprecisas - reflete-se na existência de uma sintaxe bem definida, viabilizando mecanismos de correção em que símbolos são avaliados com relação às regras de construção de expressões.

A sintaxe lógica permite inferências dedutivas, quando novas expressões são derivadas daquelas já existentes. A formulação lógica mais simples é o cálculo proposicional, que opera a partir de um conjunto de constantes e operadores lógicos. Contudo, esta simplicidade pode acarretar limitações incompatíveis com as restrições do problema, especialmente quando depara-se com um número grande de constantes. Neste caso, a primeira alternativa é a lógica de primeira ordem (Bittencourt, 1998, cap. 3), onde as proposições agora são formadas por predicados, argumentos, quantificadores e variáveis.

Existem ainda outros tipos de lógicas, como lógica não-monotônica, modal, temporal e descritiva, aplicáveis em contextos especiais.

A lógica pode ser utilizada de forma explícita, quando, por exemplo, SBC's são construídos sobre uma linguagem como Prolog, ou ainda em representações cuja existência de predicados e proposições lógicas são menos evidentes, como em listas do tipo:

(⟨ objeto ⟩, ⟨ atributo ⟩, ⟨ valor ⟩, ⟨ coeficiente de certeza ⟩)

utilizadas como formalismo de RC no sistema MYCIN (Bruchanan e Shortliffe, 1984, cap. 23). Neste caso, mesmo que o formalismo não seja diretamente expresso por lógica, seu comportamento é descrito por uma especificação lógica.

Uma representação como a lógica, pelo menos em tese, seria suficientemente expressiva para representar qualquer tipo de conhecimento. No entanto, há problemas relativos à eficiência da linguagem que são relacionados à dificuldade de expressão e tratamento de conhecimento incerto e impreciso.

**Redes Semânticas** As redes semânticas podem ser vistas como uma primeira tentativa de RC expressa graficamente (Quillian, 1968). Trata-se de um grafo direcionado onde nós representam os objetos (indivíduos, coisas, conceitos, estados) e arcos representam as relações entre objetos. Essas relações descrevem decomposições bilaterais entre objetos, que podem ser de dois tipos: (i) `classe-de` (do inglês "*is-a*") descrevendo relações entre objetos dentro de uma hierarquia taxonomica; e (ii) `faz-parte` (do inglês "*parto-of*") quando um objeto é componente de outro, não havendo nenhum tipo de herança.

Uma das características mais evidentes deste formalismo é a *herança de propriedades*, que deriva da transitividade de objetos mais gerais. Assim, um mecanismo de inferência pode, convenientemente, derivar certas propriedades para objetos mais específicos.

A figura 2.5 representa os atributos de um agente dentro do Soccerserver.[3]

**Regras de Produção** As regras de produção estão no âmago daquele que é considerado o ancestral mais antigo dos SBC's dentro da IA: os *Sistemas de Produção de Post* (Post, 1943). Esses sistemas baseavam-se na idéia de que o conhecimento pode ser codificado numa série de expressões do tipo `se` *condições* `então` *conclusões*. No princípio, as condições e conclusões resumiam-se a simples seqüências de caracteres. Hoje, entretanto, são utilizados outros formalismos de RC.

[3] c.f. item 5.2.2

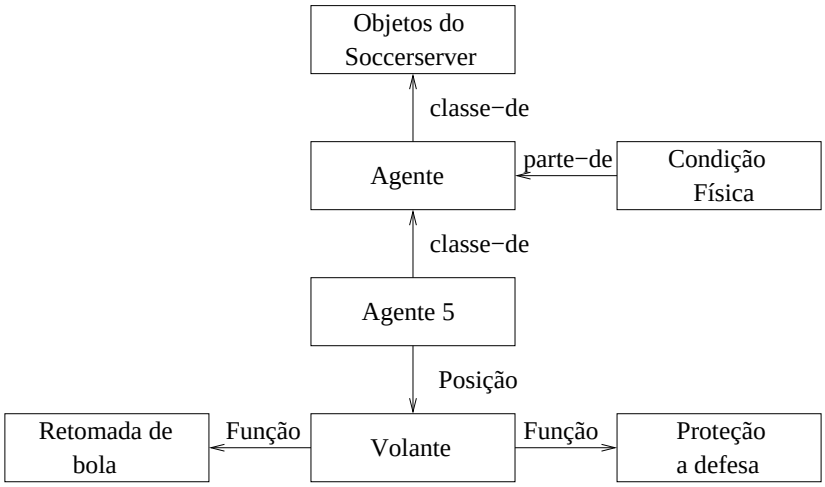


**Figura 2.5:** *Rede Semântica*

Em um SBC que manipula regras de produção, o motor de inferência avalia as regras segundo o conteúdo de sua base de fatos, gerando novos fatos. No contexto de um sistema especialista, uma regra tem uma estrutura do tipo:

SE ⟨condições⟩ ENTÃO ⟨conclusões⟩ FAÇA ⟨ações⟩

onde SE contém a lista de condições a serem atendidas, ENTÃO contém a lista de conclusões e FAÇA contém as ações que devem ser executadas.

**Quadros** A RC em quadros (do inglês *"frames"*) foi proposta originalmente em Minsky (1975), cuja idéia fundamental era a representação de objetos a partir de sua estrutura interna, permitindo o conceito de heranças de propriedades, existentes em redes semânticas.

Um quadro é identificado pelo objeto que ele define, e consiste, basicamente, na descrição de um conjunto de atributos. Pode ser visto como uma estrutura de dados, porém mais poderosa e expressiva, que possibilita a modelagem de objetos do mundo real. Cada atributo é identificado por um nome, e pode também , por sua vez, possuir uma série de atributos, neste caso, denominado *facetas*. As facetas definem o tipo, domínio, valores iniciais, e outras características de um atributo. O valor de um atributo pode ainda ser representado por um ponteiro para outro quadro.

As relações de herança implicam uma estrutura hierárquica de quadros. Essa relações ligam um quadro-pai ao seus respectivos filhos. Um quadro-pai pode ser compreendido como uma generalização do seu filho, assim como um quadro-filho pode ser uma especialização de seu pai.

A correta estruturação em quadros dos objetos de um ambiente evita a duplicação inconveniente

de informação, simplifica a representação, pela simples abstração de dados relacionados hierarquicamente. Isto gera um sistema de fácil leitura e manutenção.

**Sistemas Híbridos** As dificuldades e desvantagens de cada formalismo de RC apresentado até aqui podem ser minimizadas a partir de um formalismo híbrido. A ineficiência da lógica, por exemplo, além da sua dificuldade de interpretação pode ser contrabalanceada pela sua utilização em conjunto com uma representação em quadros, que por sua vez, pode ter atenuada sua dificuldade de representar fatos negativos, disjunções e asserções quantificadas.

As dificuldades agora referem-se aos aspectos relativos à integração destes formalismos, que deve ser viabilizada de forma a não prejudicar a funcionalidade do sistema. Neste caso, a falha mais grave seria a perda da consistência semântica do conhecimento do sistema. Para que isto não aconteça, cada formalismo deve estar relacionado a um aspecto específico do conhecimento do sistema. A seguir deve ser estabelecida uma semântica unificada, de forma a estabelecer uma relação formal entre as expressões dos diferentes, agora denominados, *subformalismos*.

### 2.4.3 Princípios de Aquisição de Conhecimento

Em sua definição mais básica, *aquisição de conhecimento* (AC) é formada pelos processos de construção de uma base de conhecimento. Buchanan et al. (1983) define AC como a transformação de conhecimento especializado com potencial para a resolução de problemas a partir de alguma fonte, transferindo-o a uma representação computacional do problema. O foco desta definição está no objetivo do processo, e não nos mecanismos que o implementam.

Já Kidd (1987) define AC a partir de etapas fundamentais do processo. De acordo com o autor, AC envolve, em caráter indispensável, a elicitação, a análise e a interpretação do conhecimento sobre um domínio, além de sua posterior transformação em uma representação de máquina. Tem-se, assim, um quadro que inclui os objetivos do processo, dado por Buchanan et al. (1983), e seus métodos fundamentais, dado por Kidd (1987).

Segundo Studer e Benjamins (1998), a aquisição de conhecimento é compreendida sob duas abordagens de desenvolvimento: aquisição de conhecimento como um *processo de transferência* e aquisição de conhecimento como um *processo de modelagem*.

Quando visto como um processo de transferência, parte-se do pressuposto de que o conhecimento sobre o domínio já existe e a tarefa do engenheiro de conhecimento é simplesmente transferi-lo

para uma base de conhecimento. Normalmente este conhecimento é representado por meio de regras de produção, como no sistema MYCIN (Shortliffe, 1976). Entretanto, regras de produção como método de representação mostram-se ineficazes para alguns aspectos do conhecimento. Atualmente, este problema pode ser atenuado por intermédio da sua inclusão em uma representação híbrida[4]. A grande fonte de problemas em abordagens por transferência reside na existência de conhecimento tácito sobre o domínio, o que gera problemas de comunicação entre engenheiro de conhecimento e especialista.

Em uma abordagem por processos de modelagem o objetivo é a construção de um modelo computacional dos métodos de resolução de problemas do especialista. A proximidade deste modelo com o especialista permite sua participação direta no desenvolvimento do SBC, minimizando problemas de comunicação, existentes na abordagem anterior. Não se trata de um modelo cognitivo do especialista, mas sim, um modelo que apresenta os mesmos resultados obtidos pela atuação deste. Como conseqüência, o conhecimento perde aquele aspecto estático adquirido na primeira abordagem. Busca-se um modelo aproximado do conhecimento, cujo objetivo final, mesmo que inalcançável, é um modelo mais próximo do conhecimento total. A modelagem dá-se por um processo cíclico, cuja aproximação do conhecimento total é obtida por sucessivos refinamentos.

### Etapas do Processo de Aquisição de Conhecimento

De maneira geral, um processo de aquisição de conhecimento é constituído por cinco fases principais (Buchanan et al., 1983):

1. **Identificação**: seguindo o modelo genérico de AC, esta fase, junto com a conceitualização do domínio, consitui-se na etapa de elicitação de conhecimento, onde ocorre, independente da metodologia, o maior período de interação entre engenheiro de conhecimento e especialista. Especificamente na fase de identificação, o objetivo é uma primeira aproximação do engenheiro de conhecimento com o domínio e, portanto, apresenta um caráter informal. Assim, cabe ao engenheiro de conhecimento identificar as classes de problemas que o sistema deverá resolver, o tipo de dados manipulados, e os critérios utilizados pelas soluções. Define-se também, a partir da complexidade da tarefa, a viabilidade técnica, e até mesmo econômica do SBC.

   Os problemas desta etapa derivam da dificuldade de comunicação entre engenheiro de conhecimento e especialista. O primeiro destes problemas é relativo à verbalização do conhecimento tácito. Normalmente, um especialista sabe resolver um problema, mas quando indagado sobre

[4] c.f. item 2.4.2

os motivos de sua decisão, ou não sabe explicar ou não apresenta uma resposta completa. Neste caso, o conhecimento está representado implicitamente, e cabe ao engenheiro de conhecimento apresentar meios para que o especialista explicite-o. Não obstante, outra dificuldade decorrente da comunicação entre os agentes do processo deve-se à diferença entre suas formações. Para que a aquisição seja efetiva, ambos devem convergir para um entendimento comum, o que pode ser feito pelo estabelecimento de uma representação comum à ambos.

2. **Conceitualização**: nesta etapa, o engenheiro de conhecimento deve formular os conceitos básicos e suas relações, identificados na fase anterior. Esta estruturação do domínio passa pela descrição dos objetos em termos de relações como causa-efeito, espaço-tempo, parte-todo.

   A etapa pode ser entendida como um processo de abstração do domínio, cuja finalidade é encontrar uma simplificação tratável computacionalmente. Tem-se, assim, uma aproximação da representação adequada ao domínio, que pode eventualmente já ser definida, desde que seja apta a representar os conceitos e relações estabelecidas.

3. **Formalização**: a etapa de formalização tem como objetivo principal estabelecer a base computacional sobre a qual será codificado o conhecimento identificado e conceitualizado nas etapas anteriores. Em outras palavras, busca-se estabelecer a semântica do conhecimento a partir da representação escolhida.

   A natureza do domínio é agora especificada por intermédio do seu espaço de busca, e pela forma como a busca é realizada. Isso implica a definição formal dos métodos de representação de conhecimento (lógica, quadros, regras de produção, etc...) e dos métodos de resolução de problemas.

4. **Implementação**: Uma vez estabelecido os aspectos formais do sistema, deve-se selecionar uma linguagem suficientemente expressiva para representar o conhecimento requerido na formalização. Esta é uma decisão que exige mais uma vez a habilidade do engenheiro de conhecimento, neste caso, para identificar a linguagem mais expressiva para o sistema. Nos primórdios destes sistemas, as linguagens mais utilizadas para SBC's eram o LISP e o PROLOG (Bittencourt, 1998, cap. 3) e mais recentemente utiliza-se C++ ou Java. A escolha da linguagem define o controle e o fluxo de informação do SBC. Ao final desta fase, obtém-se um protótipo do sistema, sujeito a avaliações do especialista e usuários.

5. **Testes**: O SBC deve ser testado segundo os requisitos de seu projeto. Entretanto não existem mecanismos formais que garantam a validade dos sistemas. Esta etapa é normalmente realizada por intermédio de exaustivos casos de testes, sujeitos a avaliação do especialista. Mesmo que não seja a melhor saída, esta continua sendo a alternativa mais utilizada para a validação de

sistemas. O ideal seria garantir a validade do SBC por meio de especificações e provas formais de correção.

Uma característica peculiar a sistemas de conhecimento em geral é a sua tolerância a respostas erradas, desde que em pequena escala. A dificuldade neste caso é determinar o limite desta tolerância, que claramente é dependente do domínio. O nível de performance do sistema deve ser equiparado ao nível de performance do especialista.

### Linguagens para Aquisição de Conhecimento

Para cada etapa do modelo de Buchanan et al. (1983) tem-se a utilização de uma ou mais linguagens, dentre as seguintes:

- **Linguagem Natural**: utilizada principalmente do período de elicitação de conhecimento, intermediando a comunicação entre engenheiro de conhecimento e especialista. É responsável muitas vezes pela inserção de erros no sistema devido a problemas de interpretação.
- **Linguagem Diagramática**: utilizada para representar conhecimento por processos de reconhecimento de padrões subsimbólicos e pelas relações entre objetos do mundo (fluxo de operações, dependências, causalidades, etc.). Além disso, permitem ainda a verificação de inconsistências nos diversos níveis do sistema.
- **Linguagens Semiformais**: combinam elementos formais e informais, de acordo com a etapa de desenvolvimento, permitindo uma melhor interação entre engenheiro de conhecimento e especialista.
- **Linguagens Formais**: utilizada quando da necessidade de representar o conhecimento segundo o formalismo do sistema.
- **Linguagens de Programação**: a linguagem utilizada para a implementação do sistema.

Entretanto, ao invés de se trabalhar com múltiplas linguagens entre as diferentes etapas da aquisição de conhecimento, uma alternativa é trabalhar apenas no nível de conhecimento, abstraindo-se detalhes de implementação. Neste caso é utilizado uma *linguagem de especificação* (Studer e Benjamins, 1998). Esta linguagem permite a descrição do conhecimento necessário ao sistema bem como dos processos de raciocínio que usa para as tarefas do sistema. Além de abstrair-se do nível de implementação, esta linguagem habilita uma especificação precisa e detalhada de um SBC além do escopo da especificação da linguagem natural.

As principais características de uma linguagem de especificação são:

- modelo conceitual forte para estruturas formais de conhecimento;
- prover meios para especificação do raciocínio de um SBC;
- adaptação aos diferentes formalismos de representação de conhecimento.

Para a formalização do nível de conhecimento, o engenheiro de conhecimento utiliza as linguagens que possibilitam a implementação do SBC no ambiente computacional. Neste nível as principais características de uma linguagem devem ser o rigor formal, permitindo minorar problemas de semântica e eliminação de erros sintáticos.

## 2.5 Metodologias para desenvolvimento de SBC's

Uma amostragem das metodologias para desenvolvimento de SBC's converge para dois tipos de ferramentas. O primeiro tipo refere-se a ferramentas baseadas na abstração de sistemas de regras, mais antigas, conhecidas como arcabouços (do inglês *shell*) de sistemas especialistas. O segundo tipo, mais atual, referem-se as ferramentas que decompõe o domínio em múltiplos modelos, deixando implícita a separação entre conhecimento e seus mecanismos de manipulação.

No itens a seguir, descrevem-se algumas dessas metodologias.

### 2.5.1 EMYCIN/TEIRESIAS

Uma primeira tentativa de simplificar o projeto de SBC's a partir de componentes reutilizáveis foi proposta com o EMYCIN (Bruchanan e Shortliffe, 1984). Trata-se de uma estrutura independente de domínio, construída a partir do sistema MYCIN, menos seu conhecimento específico sobre medicina. Suas características influenciaram os arcabouços desenvolvidos a partir do seu surgimento.

Utiliza uma linguagem em formato de regras, com certa similaridade com a notação do ALGOL. Por conseqüência, é mais simples que o LISP quanto a expressividade e mais concisa do que o subconjunto de linguagem natural utilizada pelo MYCIN.

Uma outra característica também herdada do MYCIN é um esquema de indexação de regras, que permite a organização destas em grupos, de acordo com seus parâmetros.

Sua estrutura de controle é baseada em encadeamento regressivo, adequado a sistemas de diagnóstico, como o MYCIN, que pode ser compreendido como uma árvore de operações lógicas. Sob este ponto de vista, os pontos de ramificação desta árvore são dados que podem ser consultados em tabelas ou requisitados junto ao usuário.

O EMYCIN incluí ainda um editor de conhecimento denominado TEIRESIAS. Sua função é auxiliar o engenheiro de conhecimento com o desenvolvimento e manutenção da base de conhecimento do sistema. Uma vez que não possui qualquer conhecimento específico sobre o domínio da aplicação ou da estratégia de resolução de problemas utilizadas, concentra-se na análise da sintaxe das regras de produção. Esta característica lhe garante uma certa generalidade quanto a sua utilização, uma vez que a análise é aplicável a regras de qualquer domínio. Entretanto, a responsabilidade de garantir a correção do sistema é toda do engenheiro de conhecimento.

O TEIRESIAS opera a partir de um conjunto inicial de regras, que pode ser visto como um protótipo do sistema. Para isso, ele executa estas regras de acordo com alguns casos armazenados para que o especialista gere uma avaliação do desempenho do sistema. Segundo esta avaliação, o especialista pode gerar novas regras, ou ainda, modificar aquelas existentes. Ao TEIRESIAS cabe uma monitoração destas mudanças quanto a consistência e coerência. A análise do TEIRESIAS é baseada em modelos de regras, que são, essencialmente, generalizações sobre os tipos de regras encontradas em programas de análise de desempenho. Modelos de regras podem ainda serem vistos como meta-regras, uma vez que fazem declarações gerais a respeito de regras, ao invés de declarações sobre objetos do domínio.

O TEIRESIAS provê ainda facilidades quanto a inclusão de novas instâncias de tipos de dados, evitando que se dê à nova instância uma estrutura equivocada ou mesmo não a integrando devidamente no sistema. Uma abstração de dados que permita a criação de novas instâncias é conhecida como um *esquema*, dentro do TEIRESIAS. Basicamente, esquemas são descrições de tipos de dados, assim como tipos de dados são generalizações sobre dados. Podem ser organizados hierarquicamente, obedecendo às relações de herança comumente estabelecidas.

Esta estruturação de tipos de seu nível mais superior, até sua instanciação permite ao TEIRESIAS operar sob diferentes níveis de generalidade. Num primeira instância de desenvolvimento opera-se em conhecimento *específico do domínio*, ou seja, sobre os objetos do mesmo. A seguir, permite-se uma *representação específica* do conhecimento sobre tipos de dados, e finalmente sobre um conhecimento *independente do domínio*, no que diz respeito a declarações, ou modelos de regras.

Em resumo, o EMYCIN é uma generalização do MYCIN para outros domínios. Entretanto, sua utilização mostrou que ele é mais adequado para alguns tipos de domínios do que outros, o que

acabou por limitar a sua aplicação em larga escala.

### 2.5.2 ONCOCIN/OPAL

Os primeiros sinais de uma mudança de paradigma puderam ser observados a partir do surgimento do sistema ONCOCIN/OPAL. Assim como a dupla EMYCIN/TEIRESIAS, o ONCOCIN (Shortliffe et al., 1981) é um arcabouço de sistemas especialistas utilizado especificamente para terapia de câncer baseado em protocolos. Sua base de conhecimento é formada por regras de produção e outras estruturas de dados que capturam a semântica dos protocolos de teste clínicos de câncer. O OPAL (Musen et al., 1987), por sua vez, é uma ferramenta computacional utilizada para auxiliar o engenheiro de conhecimento e o especialista na aquisição de conhecimento para o ONCOCIN. O OPAL sugere a aproximação do especialista com o processo de construção da base de conhecimento. Nos sistemas EMYCIN/TEIRESIAS, sua participação estava restrita as etapas de elicitação e teste. Com o OPAL, o especialista tem participação direta na fase de implementação da base de conhecimento.

Esta participação direta do especialista é feita a custa de uma dependência do conhecimento do domínio, o que evidencia mais uma diferença em relação ao TEIRESIAS. O OPAL usa conceitos e relações específicos do domínio para apresentar a oncologistas formulários cuidadosamente projetados para a entrada de dados estruturados. Ao invés de codificação de regras de produção individuais, os próprios médicos descrevem protocolos de câncer pelo preenchimento de formulários gráficos de propósito especial. O OPAL, então, transcreve este conhecimento dentro da representação interna do ONCOCIN.

Segundo a nomenclatura do domínio, o ONCOCIN descreve um protocolo como um tratamento específico a ser indicado, ou seja, combinações específicas de drogas por um período de tempo, junto com testes de laboratório e terapia radioativa. Este protocolos são armazenados na base de conhecimento na forma de planos. A operação do sistema é dada pela seleção de um protocolo, e sua instanciação é dada pelo especificação das drogas, rotinas de administração, entre outros aspectos.

Em resumo, a principal contribuição do OPAL não foi a ferramenta em si, mas sim, o conceito de aquisição de conhecimento utilizando aspectos específicos do domínio. Dentre outras vantagens, esta abordagem acelerou o tempo de desenvolvimento de um sistema especialista, o que até então era medido na casa de anos de projeto. Entretanto, sua aplicabilidade estava limitada ao domínio e abrangência, no caso, terapias de câncer. Por outra lado, a participação mais ativa do especialista no desenvolvimento do sistema, minimizou problemas relativos à comunicação.

Desta forma, estes dois aspectos, participação ativa do especialista e inserção de aspectos específicos do domínio, constituiriam diretrizes nas metodologias vindouras.

### 2.5.3 KADS/CommonKADS

EMYCIN/TEIRESIAS e ONCOCIN/OPAL constituíam-se nas principais referências para o desenvolvimento de sistemas especialistas por volta do final dos anos 70 e início dos anos 80. Entretanto, as limitações existentes fomentaram a busca de respostas mais satisfatórias, especialmente no gargalo do processo, a aquisição de conhecimento. O que se observou neste período foi o deslocamento de soluções em torno de ferramentas computacionais para metodologias de construção de sistemas e aquisição de conhecimento, de natureza mais flexíveis. Mais do que ferramentas, as metodologias, ou modelos, buscam a definição de uma série de passos ou processos através dos quais um engenheiro de conhecimento garantirá a produção de sistemas de alta qualidade. Em outras palavras, tais sistemas devem ser confiáveis, sujeitos a manutenção e devem estar de acordo com os requisitos do usuário. Metodologias, portanto, posicionam-se num nível de abstração mais elevado, o que serviria de diretrizes para o desenvolvimento das próximas ferramentas computacionais. Neste novo contexto, mais generalizado, sistemas especialistas passam então a ser denominados de SBC.

Segundo Plant e Gamble (2003) estas metodologias podem ser divididas em dois tipos, segundo os requisitos do sistema a ser desenvolvido: modelos industriais, que primam pela velocidade de implementação e execução do sistema, e modelos formais, cuja principal característica é o rigor formal que garanta um desenvolvimento com correção de sistemas mais complexos. Entre os modelos industriais de destaque estão o KBSDLC, também conhecido como metodologia de Weitzel e Kerschberg (Weitzel e Kershberg, 1989), o modelo de Miller (Miller, 1989), o modelo AISE ANSI/AIAA (ANSI, 1992), e o MOKA (Stokes, 2001). A característica comum a todos é a uma prototipagem rápida sujeita a ciclos de refinamento até um sistema considerado satisfatório. Já as principais metodologias formais são as baseadas em tarefas e as baseadas em meta-conhecimento. Estas metodologias visam essencialmente atenuar a ausência de um maior rigor matemático na especificação de um SBC. Para tanto, estes modelos propõem níveis de refinamento, onde um modelo conceitual inicial evolui para uma série de processos, onde cada um descreve um visão do domínio, como por exemplo, conhecimento, representação e processos cognitivos. O principal produto da metodologia formal é o KADS/CommonKADS (Schreiber et al., 1999).

A principal argumentação da metodologia KADS é a de que um SBC não é simplesmente um *container*, que deve ser preenchido com conhecimento extraído de um especialista, mas um modelo operacional que exibe algum comportamento desejado dentro do seu ambiente. A metodologia

KADS pode ser vista como o marco que delimita uma mudança de paradigma, já destacada neste capítulo, quando do deslocamento da abordagem por transferência para a abordagem por modelos. Este é o momento em que a engenharia de conhecimento deixa de ser simplesmente o desenvolvimento de técnicas de aquisição de conhecimento, caracterizada apenas pela captura e representação de um conhecimento elicitado, e passa a ter uma perspectiva em nível de conhecimento, segundo a denominação de Newell (1982).

Sob este novo modelo de desenvolvimento, o KADS sustenta o projeto de um SBC sob o mesmo conjunto de etapas definido por Buchanan et al. (1983), entretanto, a distinção entre elas não é evidente. Esta nova dimensão dada ao modelo de Buchanan et al. (1983) pode ser compreendida a partir de três princípios básicos da metodologia:

1. A introdução de múltiplos modelos como um meio para o gerenciamento da complexidade do processo de engenharia de conhecimento.
2. Uma estrutura em três camadas principais para a modelagem da perícia requerida.
3. A reutilização de componentes genéricos como estruturas para aquisição de conhecimento sob uma abordagem *top-down*.

A complexidade do sistema é agora dividida em uma coleção de modelos, onde cada um captura aspectos específicos do SBC e do ambiente. São eles: *Modelo de Organização*, *Modelo de Tarefa*, *Modelo de Agente*, *Modelo de Comunicação*, *Modelo de Perícia*, *Modelo de Projeto*, cujas relações são descritas pela figura 2.6.

O *modelo de organização* permite estruturar o sistema a partir da especificação das funções que deve desempenhar, distinguidas em unidades organizacionais. Desta forma permite-se a identificação dos processos do sistema, que subseqüentemente podem ser depurados e otimizados.

O *modelo de tarefa* apresenta uma estrutura hierarquica de tarefas nas quais podem ser organizadas as unidades organizacionais definidas no modelo de organização. Estas tarefas são definidas a partir de suas entradas, saídas, controle interno, restrições de ambiente e capacidades requeridas. O KADS denomina este processo de *análise de tarefa*. O modelo inclui ainda a especificação de que agentes são designados para as diferentes tarefas.

O *modelo de agente* define a capacidade de cada agente envolvido em alguma das tarefas do domínio. Isto é feito pela especificação do conjunto de propriedades relevantes dos agentes para as diferentes tarefas.

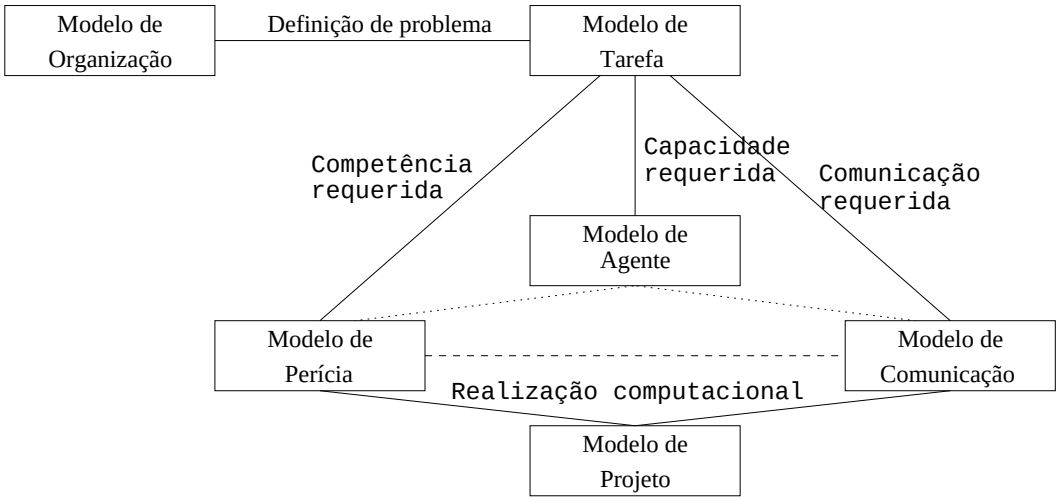


**Figura 2.6:** *CommonKADS e as relações entre seus modelos segundo Plant e Gamble (2003)*

O *modelo de comunicação* é responsável por identificar a decomposição de tarefas, distribuí-las e ainda definir as interfaces entres os agentes. A distribuição deve ser feita segundo as restrições do modelo de agente e do modelo de perícia.

O diferencial da abordagem KADS está no *modelo de perícia,* que distingue três tipos de conhecimento para a resolução de uma tarefa específica. Grosso modo, corresponde a etapa de AC da metodologia. Estes tipos correspondem à visão estática, à visão funcional e à visão dinâmica do SBC. Na nomenclatura KADS são denominados *conhecimento do domínio*, *conhecimento de inferência* e *conhecimento de tarefa*.

O *conhecimento de domínio* especifica todo o conhecimento necessário para a resolução de uma tarefa. Este conhecimento inclui a conceitualização do domínio em uma ontologia de domínio (Guarino e Giaretti, 1995), e uma teoria declarativa deste conhecimento. Esta estrutura permite reutilização.

O *conhecimento de inferência* especifica os processos de raciocínio utilizados pelo SBC. As *ações de inferência* são escolhidas de acordo com o *papel* executado pelo conhecimento de domínio tratado. Uma *estrutura de inferência* determina as dependência entre as suas ações e o papel. Esta noção de papel facilita as questões de reuso deste conhecimento uma vez que lhe provê uma visão independente do domínio.

O *conhecimento de tarefa* determina a decomposição das tarefas em subtarefas e ações de inferência, associando-as a metas e como estas metas podem ser alcançadas. Este conhecimento também provê os mecanismos de controle sobre as subtarefas e as ações de inferência.

Com a separação explícita entre o conhecimento de domínio da descrição genérica dos métodos de resolução de problemas e do conhecimento de tarefa, habilitam-se pelo menos dois tipos de reuso. O primeiro tipo é o reuso de conhecimento de domínio para a resolução de diferentes tipos de tarefas por diferentes métodos de resolução de tarefas. O segundo tipo é o reuso de métodos de resolução de problemas em diferentes domínios, definindo uma nova visão de um outro conhecimento de domínio. A extensão do KADS, o CommomKADS, permite o reuso e a configuração destes componentes do modelo de perícia.

O modelo de perícia é especificado por uma linguagem semi-formal, a Linguagem de Modelagem Conceitual (do inglês "Conceptual Modelling Language") (CML), e por uma linguagem formal, a ML, baseada em lógica de predicados de primeira ordem. A primeira é utilizada para a comunicação entre engenheiro de conhecimento e especialista, enquanto a segunda é utilizada para a formalização do modelo.

Finalmente, *o modelo de projeto* é a realização dos requisitos funcionais especificados nos modelos de perícia e comunicação. O modelo de projeto, então, especifica a arquitetura do sistema e os mecanismos computacionais para a implementação as ações de inferência.

O KADS suporta o desenvolvimento de atividades segundo uma construção gradual dos diferentes modelos, segundo um modelo cíclico, como o modelo de desenvolvimento de software em espiral (Pressman, 1995). Sua utilização deve-se, especialmente, a uma abordagem que visa a minimização de riscos de falhas de projeto.

### 2.5.4 Protégé

Nenhuma outra metodologia reflete com tanta afinidade as mudanças de paradigmas ocorridas na Engenharia do Conhecimento como o Protégé. De fato, o Protégé evoluiu de uma simples ferramenta de AC até uma metodologia completa para desenvolvimento e pequisa em SBC's. Seguindo as diretrizes estabelecidas pelo OPAL, seu objetivo era minimizar o papel do engenheiro de conhecimento na construção da base de conhecimento. Ainda derivado do OPAL, advém a concepção de acelerar o processo por intermédio da adequação da metodologia a aspectos específicos do domínio. O Protégé em sua versão atual, é uma suite de ferramentas para a criação de ontologias de domínio e geração de ferramentas de AC, como o próprio OPAL por exemplo, para aplicações particulares.

As principais características do Protégé foram sendo incorporadas com o tempo, e portanto, a análise destas características corresponde a uma análise histórica do desenvolvimento da ferramenta.

Em sua primeira versão, o Protégé já previa uma clara distinção entre classes de usuários, e seus respectivos papéis. O engenheiro de conhecimento é responsável por prover os conceitos estruturais do domínio e construir uma ferramenta de AC específica. Ao especialista cabe a tarefa de utilizar esta ferramenta para a construção e edição das bases de conhecimento. O usuário final interage com o SBC para o suporte de decisão. A metodologia geral assume que o conhecimento é adquirido em estágios, e a informação adquirida em cada estágio deve ser utilizada como meta-conhecimento para os estágios subseqüentes. Deste modo, o Protégé-I aproxima o especialista da construção da base de conhecimento, diminuindo a probabilidade de ruídos semânticos quanto ao conhecimento adquirido.

Em contrapartida, o Protégé-I falhava nos aspectos de reutilização de componentes bem como na generalização de sua aplicação. As hipóteses desta primeira versão ainda limitavam sua aplicação a domínios médicos.

Com o Protégé-II veio a concepção de métodos de resolução de problemas reutilizáveis. Permite-se, assim, a construção de mecanismos de inferência como um componente inteiramente a parte, desenvolvido independentemente da base de conhecimento. O benefício da separação entre os métodos de resolução de problemas e as bases de conhecimento é que os primeiros são relativamente mais estáveis. Empiricamente já é possível constatar que o conhecimento do domínio necessita de constante atualização e requer substanciais modificações. Uma vez definido o método de resolução de problemas, este pode ser reutilizado com diferentes bases de conhecimento.

Um outro benefício desta reutilização está na possibilidade de decomposição dos métodos. Os sub-métodos podem ser compostos, por sua vez, por sub-tarefas, até que se alcance os métodos primitivos, denominados dentro do Protégé de *mecanismos*. Permite-se, desta forma, a configuração de métodos genéricos para tarefas específicas, selecionando os sub-métodos apropriados ao problema.

O suporte destas novas características é dado pela introdução da noção de *ontologia*, um modelo formal para o compartilhamento de discurso (Guarino e Giaretti, 1995). O Protégé considera ontologias de domínio como a base para a geração de ferramentas de AC, que também é utilizada para as necessidades de representação de conhecimento e comunicação.

Dentro do Protégé, ontologias utilizam um formalismo baseado em quadros. Sua linguagem, portanto, baseia-se nos conceitos padrões de quadros como *classes, instâncias, slots* e *facetas*, além de permitir herança entre classes.

O uso de ontologias explicita o conflito entre conhecimento estrutural sobre um domínio e o conhecimento mais específico, fornecido pelo especialista. O Protégé aborda o primeiro tipo por meio de classes, enquanto que o segundo tipo é abordado por instâncias.

Ontologias podem ser usadas de diferentes maneiras, e o Protégé as abordas segundo três classes distintas: *ontologias de domínio*, *ontologias de método* e *ontologias de aplicação*.

Uma ontologia de domínio define uma conceitualização compartilhada do domínio. Juntamente com os métodos de resolução de problemas, são componentes reutilizáveis do SBC.

Uma ontologia de método especifica as entradas e saídas de cada método de resolução de problemas, provendo sua funcionalidade. Estabelece, portanto, a relação entre a ontologia de domínio e os métodos de resolução de problemas, definindo o grau de reutilização destes componentes.

Já a ontologia de aplicação define conceitos que são específicos de uma aplicação particular, e portanto, não são passíveis de reutilização.

A associação entre as ontologias de método e aplicação são feitas por diferentes tipos de mapeamentos, que podem ser:

- *Mapeamento de Renomeação*: usado para traduzir termos específicos do domínio em termos específicos do método;
- *Mapeamento de Filtragem*: prove meios para que um subconjunto de instâncias de domínio correspondam a conceitos do método;
- *Mapeamento de Classe*: permite a instanciação de conceitos de métodos a partir dos conceitos da aplicação.

Um dos desafios no projeto do Protégé era permitir aspectos específicos de um domínio na construção da base de conhecimento, como na utilização do OPAL, sem no entanto, perder em generalidade de aplicação. O objetivo é permitir a interação direta entre SBC e especialista para a construção da base de conhecimento, o que causa uma aceleração no desenvolvimento do sistema. Mesmo que paradoxal em um primeiro instante, esse requisito é possível a partir do uso de ontologias distribuídas na três classes citadas acima.

Para que essa especialização sem perda de generalidade seja possível, o Protégé disponibiliza uma estratégia de aquisição de conhecimento em meta-nível. Esta estratégia possibilita a geração de ferramentas de AC específicas para o domínio. A ontologia é base para a geração desta ferramenta. Este processo é possível pela utilização de três sub-componentes do Protégé: o Maître, para a construção de ontologias, o Dash, para a manipulação do interface da ferramenta de AC, e o Meditor, que o especialista usa para construir e editar bases de conhecimento.

Com a ferramenta de AC pronta, o próximo passo é a construção da base de conhecimento. Em síntese, significa a instanciação das classes pré-definidas na ontologia de domínio. O diferencial é que agora esta instanciação é feita pelo próprio especialista, enquanto o papel do engenheiro de conhecimento está restrito à construção da ontologia. A principal dificuldade neste modelo de desenvolvimento acontece quando há necessidade de se efetuar mudanças nas ontologias. Estas mudanças, depois de construídas as instâncias, são complicadas de tratar, uma vez que a definição de classes afeta todas as instâncias da base, além de afetar a própria ferramenta de AC gerada.

A partir desta versão, as principais mudanças no Protégé deram-se em sua interface, e em escala menos relevante, em sua metodologia.

### 2.5.5 Análise Comparativa entre as Metodologias

Apesar da falta de padronização entre as metodologias existentes, é possível destacar algumas características comuns que permitem diferenciá-las tendo em vista uma análise comparativa. Mesmo que esta análise não cubra a totalidade das metodologias e ferramentas, esta seleção atende ao critério de classificação de Studer e Benjamins (1998), apresentado no item 2.4.3. Desta forma, foram selecionadas duas ferramentas para cada abordagem, de acordo o critério de relevância do autor. A tabela 2.1 apresenta um resumo comparativo entre os aspectos comuns das metodologias, permitindo avaliações, especialmente qualitativas, sobre sua viabilidade nas mais variadas aplicações destinadas a SBC's.

| *Critério/Metodologia* | EMYCIN/TEIRESIAS | ONCOCIN/OPAL | KADS/CommonKADS | Protégé |
|---|---|---|---|---|
| Abordagem | Transferência | Transferência | Modelo | Modelo |
| Relação com Domínio | Independente | Dependente | Independente | Independente |
| Representação de Conhecimento | Regras de Produção | Regras de Produção | Múltiplos | Quadros |
| Níveis de Concepção | Simbólico | Simbólico | Conhecimento e Simbólico | Conhecimento e Simbólico |
| Linguagens | Formal | Formal | Semi-Formal e Formal | Semi-Formal e Formal |
| Mecanismos de Inferência | Encadeamento Regressivo | Encadeamento Regressivo | Múltiplos | Múltiplos |
| Grau de reutilização | Médio | Baixo | Alto | Alto |
| Meta-Conhecimento | Ausente | Presente | Presente | Presente |

**Tabela 2.1:** *Tabela comparativa entre metodologias para SBC's*

O primeiro critério diz respeito à abordagem de construção do sistema, segundo a classificação de Studer e Benjamins (1998). O EMYCIN/TEIRESIAS e o ONCOCIN/OPAL funcionam segundo

a abordagem de transferência, em que, após uma etapa de elicitação junto ao especialista, o engenheiro de conhecimento transfere este conhecimento para a representação do sistema. Já na abordagem por modelos, o engenheiro de conhecimento estrutura o conhecimento do especialista segundo um conjunto de modelos. Estas estruturas servem para, entre outras funções, permitir uma maior aproximação do especialista na construção da base de conhecimento, minimizando possíveis fontes de problemas. No KADS/CommonKADS, esta estruturação dá-se por seis modelos distintos, que uma vez especificados pelo engenheiro de conhecimento, devem ser instanciados pelo conhecimento do especialista. Já no Protégé, o engenheiro de conhecimento define um conjunto de classes que englobam os conceitos principais do sistema e que devem ser instanciadas diretamente pelo especialista.

O critério *relação com o domínio* estabelece o grau de ligação entre a metodologia e um domínio específico. Em uma análise inicial, tem-se apenas o shell ONCOCIN/OPAL como dependente do domínio, indicado para protocolos de terapias de câncer. Entretanto, o sistema EMYCIN/TEIRESIAS, mesmo que reivindique sua independência, vê suas possibilidades restritas a domínios adequados a uma representação de conhecimento em regras de produção e mecanismos de controle sob encadeamento regressivo. Um exemplo de domínios deste tipo são sistemas de planejamento e diagnóstico. Quanto a relação com o domínio, o KADS/CommonKADS e Protégé tem um contexto mais geral, uma vez que é bastante flexível quanto a estes critérios.

Concluí-se que, de fato, os critérios que determinam a aplicabilidade de uma metodologia a um domínio são os formalismos de representação de conhecimento, e os métodos de resolução de problemas disponibilizados. Quanto menos restritos, mais geral o contexto da metodologia.

Os demais critérios têm relação direta com os processos de verificação do sistema e velocidade de execução do projeto.

A concepção em diferentes níveis de abstração de projeto e as linguagens utilizadas determinam o grau de participação do especialista no projeto. Independente das verificações de nível lógico e sintático, esta presença mais ativa atenua os ruídos semânticos do processo, garantindo uma maior correção do sistema final. A exceção dá-se com o OPAL, que aproxima o especialista ao projeto pelo seu conhecimento específico do domínio. Por outro lado, a presença de linguagens semi-formais permitem ao engenheiro de conhecimento estabelecer um canal de comunicação com o especialista, sem o rigorismo de linguagens formais, próprias da formação do projetista.

A utilização de meta-conhecimento tem por objetivo criar uma estrutura conceitual que permita também a aproximação do especialista, sem que a ferramenta perca sua independência com relação domínio. O Protégé, por exemplo, utiliza este meta-conhecimento para a criação de uma ferramenta de AC específica às necessidades do domínio e ao conhecimento do especialista.

Uma conclusão importante é a relação entre a generalidade da metodologia e a complexidade associada. O que se observa é que quanto mais geral, mais complexa é a utilização da metodologia. Nesse sentido, o exemplo mais representativo é dado pelo KADS/CommonKADS. A concepção de um SBC passa pela estruturação de seis modelos, que abrangem aspectos diferentes do conhecimento, e a determinação da relação entre eles. Não obstante, o modelo de perícia deve ainda estruturar o conhecimento específico do especialista em outros três níveis - domínio, inferência e tarefa. Por outro lado, o ONCOCIN/OPAL atende as mesmas etapas apenas com o preenchimento de formulários dedicados especificamente aos protocolos terapêuticos.

Em resumo, a escolha da metodologia deve ser orientada pelas seguintes diretrizes: complexidade do problema, velocidade do projeto, adequação ao domínio e conhecimento do especialista. Quanto mais geral o conhecimento tratado, aconselha-se sistemas mais gerais como o KADS/CommonKADS ou Protégé. Esta escolha, por outro lado, tem relação com o tempo necessário para o desenvolvimento do projeto, que neste caso, pode ser na casa de anos de trabalho até que tenha-se um sistema de acordo com seus requisitos mínimos. Metodologias mais simples, como os shells, podem ser indicados para problemas de menor complexidade, diminuindo o tempo necessário de projeto. O problema neste caso seria a falta de generalidade destas soluções.

## 2.6 Considerações Finais

Neste capítulo, o objetivo foi descrever a área em que se contextualiza o tema desta tese, ou seja, o desenvolvimento de SMA segundo uma abordagem baseada em conhecimento. Além disso, buscou-se estabelecer referenciais teóricos para as contribuições deste trabalho por intermédio de metodologias e/ou abordagens correlatas.

Desta forma, as conclusões estão divididas em dois aspectos: o aspecto social, formado pela avaliação das metodologias de desenvolvimento de SMA, e o aspecto individual, que concebe agentes segundo uma abordagem baseada em conhecimento, formado pela avaliação das metodologias de desenvolvimento de SBC's.

A maioria das metodologias de desenvolvimento de SMA derivam daquelas aplicadas ao desenvolvimento de programas orientados a objeto. A diferença está na extensão do conceito de objeto de forma a compreender as propriedades de autonomia e comunicação próprias da tecnologia de agentes. Contudo, um paradigma orientado à agentes deve necessariamente estar a apto a representação de conceitos fundamentais, como papéis, organização, interação e coordenação. Além disso, seria interessante que um mesmo formalismo fosse utilizado durante todo o processo de desenvolvimento,

facilitando o mapeamento e as relações entre as fases do processo global. A fim de reduzir a complexidade da especificação da inúmeras interações, a metodologia deve estabelecer as relações entre os agentes por meio de mecanismos de coordenação, como planejamento por exemplo, independente de protocolos e linguagens. No nível de estruturação interna do agente, para que se possa chegar o mais próximo possível do seu nível de implementação final, propor-se-ia restrições de arquiteturas baseadas em conhecimento, como uma alternativa a arquitetura BDI, menos geral, e por vezes, mais complexa.

Entretanto, mesmo as metodologias baseadas em conhecimento estão sujeitas a uma avaliação para a sua implementação segundo as diretrizes estabelecidas no parágrafo anterior. Neste caso, seria interessante que o formalismo que descrevesse o nível social do SMA pudesse ser utilizado também no nível individual, tornando a relação entre os conceitos sociais e sua implementação nos agentes explícita e direta. Não obstante, a fim de garantir uma maior generalidade à metodologia e/ou abordagem, é importante que o nível individual seja capaz de reconhecer qualquer representação de conhecimento. Finalmente, uma restrição interessante seria a possibilidade de estruturar o conhecimento em múltiplos níveis de abstração, de forma a facilitar o processo de aquisição.

Dadas estas restrições, o próximo capítulo apresenta as Redes de Petri, formalismo utilizado para a estruturação do SMA segundo a abordagem apresentada no capítulo 4.

# Capítulo 3

# Redes de Petri

## 3.1 Introdução

Neste capítulo faz-se uma introdução às Redes de Petri (RP) a partir de dois aspectos fundamentais. Primeiro, justificando-as como linguagem de especificação na abordagem para especificação de conhecimento para Sistemas Multiagentes (SMA), a ser descrita no próximo capítulo. Segundo, apresentando uma revisão bibliográfica sobre RP como ferramenta na especificação e implementação de SMA, seja no nível social ou individual do processo.

O capítulo é assim organizado. A seção 3.2 apresenta o modelo básico de uma RP, destacando seu modelo formal e a análise de suas propriedades. A seção 3.3 faz uma introdução aos conceitos relativos aos modelos de alto-nível de RP, destacando as diferenças entre os modelos, e destes com o modelo ordinário. A seção 3.4 apresenta o processo de hierarquização entre RP, o que garante ao modelo maior modularidade na construção de sistemas. A seção 3.5 apresenta uma revisão bibliográfica sobre a utilização de RP no desenvolvimento de SMA e Sistemas Baseado em Conhecimento (SBC) em geral. E finalmente, a seção 3.6 apresenta as conclusões do capítulo.

## 3.2 Modelo Básico

A RP é um modelo gráfico/matemático utilizado na especificação, modelagem e verificação de *Sistemas a Eventos Discretos*. Sistemas deste tipo são caracterizados por variáveis de estado que podem mudar abruptamente em instantes determinados. Contudo, os próximos valores destas variáveis são determinados a partir do estado atual do sistema.

As RP foram propostas em 1962 com a tese de Carl Adam Petri (Petri, 1962), sob o título *Comunicação com Autômatos*, defendida na Universidade de Darmstadt, Alemanha. Hoje em dia são aplicadas na modelagem de atividades que envolvem concorrência, sincronia entre eventos, distribuição, paralelismo e não-determinismo.

Sob esta abordagem, sistemas são construídos a partir de seus *eventos*, *atividades* e *processos*. Um conjunto de processos é caracterizado pelo modo como evoluem, que pode ser simultâneo ou não. Em se tratando de evoluções simultâneas, o grau de independência entre processos é medido pela necessidade de pontos de *sincronização*. Neste aspecto, as RP se diferenciam de outros modelos, como os autômatos por exemplo, por permitirem a representação de atividades paralelas, sejam elas no sentido de cooperação, competição ou apenas concorrência.

Uma RP é implementada a partir da utilização de três elementos fundamentais, que podem ser interpretados livremente: lugares, transições e fichas. De acordo com Cardoso e Valette (1997), são assim caracterizados:

- **Lugar**: pode ser interpretado como uma condição, um estado parcial, uma espera, um procedimento, um conjunto de recursos, um estoque, uma posição geográfica, um sistema de transporte, etc. Representado por um círculo, em geral, todo lugar tem um predicado associado, como por exemplo, *jogador livre*, *posse de bola*, como no exemplo da figura 3.1.

- **Transição**: Representado por uma barra ou retângulo, é associada a um evento que ocorre no sistema, como a transição *t* na figura 3.1.

- **Ficha**: é um indicador significando que a condição associada ao lugar é verdadeira. Representado por um ponto num lugar, pode indicar um objeto num determinado estado. Em redes ordinárias tem um caráter meramente binário, mas que pode também representar estruturas de dados mais complexas em redes de alto nível. Por exemplo, uma ficha no lugar *jogador livre* indica que um jogador está livre. A ausência de fichas neste lugar indica que não há jogadores livres.

A dinâmica do sistema é dada pela movimentação de fichas na rede. As transições estão normalmente associadas aos eventos do sistema, de forma que a ocorrência destes eventos levam à movimentação de fichas na rede. No entanto, o disparo de uma transição só é permitido caso existam fichas nos respectivos lugares de entrada desta transição. Caso esta condição seja verdadeira, a ocorrência de um evento leva a retirada das fichas dos lugares de entrada associados à transição em

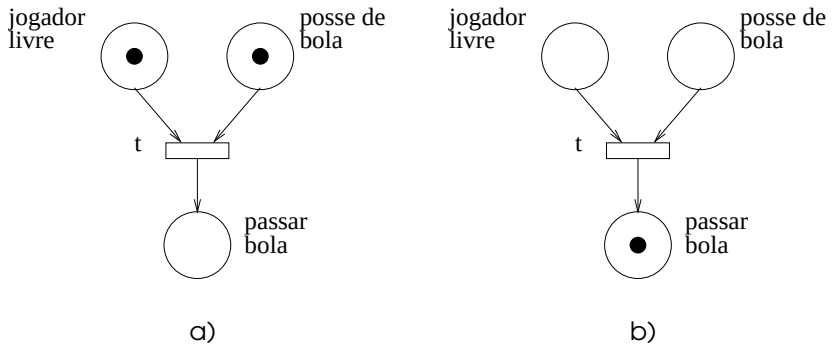


**Figura 3.1:** *Rede de Petri*

questão e imediatamente deslocando-as para os lugares de saída desta mesma transição. O estado do sistema é determinado, portanto, pela repartição de fichas na RP.

Na figura 3.1, a ocorrência do evento *executar passe*, associado à transição *t*, só é permitido uma vez que há um jogador livre (ficha no lugar *jogador livre*), e que o agente em questão tenha a posse de bola (ficha no lugar *posse de bola*). O disparo da transição *t* tem o efeito de retirar as fichas pertencentes aos seus lugares de entrada e coloca-las no lugar de saída *passar bola*, disparando o conjunto de comportamentos relativos a ação descrita. Desta forma, os predicados associados aos lugares de entrada da transição *t* deixam de ser verdadeiros com o desaparecimento das fichas ali contidas, e o predicado associado ao lugar de saída passa a ser a nova condição verdadeira do sistema.

Ao disparar a transição *t*, que corresponde ao evento *e* no sistema, este sai do estado atual $E_i$ para o próximo estado $E_{i+1}$. Um dado estado $E_i$ do sistema é representado pela marcação $M_i$ na RP. Desta forma, diz-se que, assim como a ocorrência do evento *e* leva o sistema do estado $E_i$ para o estado $E_{i+1}$, o disparo da transição *t*, associada ao evento *e*, leva a RP da marcação $M_i$ para a nova marcação $M_{i+1}$.

**Definição 1** *Graficamente, uma RP é uma n-tupla*

$R = \langle P, T, Pre, Post \rangle$

*onde:*

- *P é um conjunto finito de lugares com dimensão n.*
- *T é um conjunto finito de transições com dimensão m.*
- *Pre é uma função de incidência anterior:* $Pre : P \times T \rightarrow \mathbb{N}$, *onde* $Pre(p,t) = w$ *implica que p é um lugar de entrada da transição t, conectados por um arco com peso w.*

- *Post é uma função de incidência posterior: Post* : $P \times T \rightarrow \mathbb{N}$, *onde* $Post(p,t) = w$ *implica que p é um lugar de saída da transição t, conectados por um arco com peso w.*

**Definição 2** *Uma rede marcada N é uma dupla*

$N = \langle R, M \rangle$

*onde:*

- *R é uma RP.*
- $M : P \rightarrow \mathbb{N}$ *é a marcação da rede e* $M(p)$ *é o número de fichas contidas no lugar p.*

O comportamento dinâmico do sistema é orientado por um conjunto de regras de transição. Uma dada transição é dita *habilitada* caso o número de fichas contidas nos seus lugares de entradas sejam maiores ou iguais aos respectivos pesos de arco, ou seja, uma transição $t$ está habilitada quando $M(p) \geq Pre(p,t)$, para cada um dos lugares de entrada $p$. O *disparo* de uma transição ocorre retirando-se $Pre(p,t)$ fichas de cada lugar de entrada $p$ de $t$, e colocando $Post(p,t)$ fichas em cada lugar de saída $p$ de $t$.

Além da representação em grafos, as RP possuem uma representação matricial, o que lhe dá o rigor formal necessário a alguns dos processos de análise de propriedades. A formalização da representação matricial de um RP pode ser obtida em Cardoso e Valette (1997, cap. 2), Murata (1989) e David e Alla (1994).

Uma das vantagens das RP é o suporte à análise de propriedades dos sistemas especificados. Estas propriedades são de dois tipos: uma dependente da marcação inicial, denominadas *propriedades do modelo* ou comportamental, e outra independente da marcação inicial, denominadas *propriedades estruturais*.

As propriedades do modelo derivam da análise de uma rede marcada, e são agrupadas sob o nome genérico de *boas propriedades*. São elas: *vivacidade, limitabilidade* e *reiniciabilidade*. A verificação destas propriedades dá-se no contexto do conjunto das marcações acessíveis da rede, ou seja, sua árvore de cobertura, dependente da marcação inicial.

### Limitabilidade

Um RP $N = \langle R, M_0 \rangle$ é dita *k-limitada* se e somente se todos os seus lugares são k-limitados em todo $M \in A(R, M_0)$. Um lugar k-limitado significa que para um dado lugar $p$ qualquer de $N$,

$M(p) \leq k$. Se $k = 1$ em $N$, a rede é dita *binária* ou *segura*. Fisicamente, uma rede limitada implica um sistema com gerenciamento seguro de recursos, sem a previsão de sobrecargas.

### Vivacidade

Uma RP $N = \langle R, M_0 \rangle$ é dita *viva* se e somente se é possível alcançar qualquer marcação $M' \in A(R, M_0)$ a partir de uma marcação $M \in A(R, M)$. Isto implica dizer que o sistema projetado é livre de bloqueios mortais, que causam o travamento do mesmo em um determinado estado. Uma rede viva é conseqüência de um conjunto $T$ de $R$ que possui apenas transições vivas, ou seja, transições disparáveis a partir de qualquer marcação da RP.

### Reiniciabilidade

Uma RP $N = \langle R, M_0 \rangle$ é dita *reiniciável* ou *reversível* se e somente se, para cada marcação $M \in A(R, M_0)$ é possível alcançar $M_0$. Isso significa dizer que uma rede é reiniciável se a partir de qualquer marcação acessível é possível encontrar uma seqüência de disparo que a retorne ao estado inicial.

As propriedades estruturais de uma RP dependem apenas da topologia da rede. Uma vez que tais propriedades são verificadas para qualquer marcação, elas independem de $M_0$. Analisadas sob a luz da respectiva matriz de incidência $C$ da rede, determinam os chamados *componentes conservativos* e *componentes repetitivos* da estrutura.

### Componentes Conservativos e Invariantes de Lugar

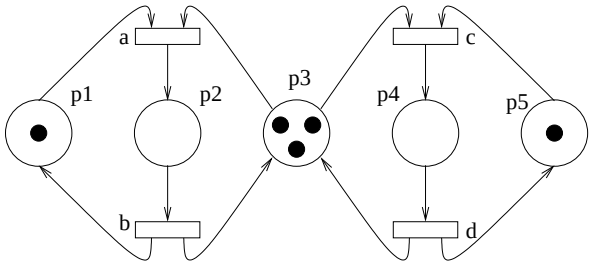


**Figura 3.2:** *Invariantes*

Seja a RP da figura 3.2. Mais especificamente, considere a sub-rede formada pelos lugares $p_1$ e $p_2$ e pelas transições $a$ e $b$. Assumindo a marcação inicial descrita na figura, observa-se a seguinte

relação: $M(p_1)+M(p_2)=1$. Supondo o disparo de *a* temos agora $M'(p_1)+M'(p_2)=1$. A conclusão direta é que o tiro da transição *a* não modifica a referida soma. Mesmo o tiro das demais transições não alteram a relação de soma entre os lugares $p_1$ e $p_2$. Generalizando a todas as marcações acessíveis da rede a partir da marcação inicial, tem-se:

$$\forall M \in A(R,M_0), M(p_1)+M(p_2)=M_0(p_1)+M_0(p_2)$$

A sub-rede de uma RP a qual respeita a relação de soma entre seus lugares, assim como descrito no exemplo da figura 3.2, é chamada *invariante linear de lugar*. O conjunto de lugares $p_1$ e $p_2$ formam um *componente conservativo* da rede.

Ainda na mesma rede, a sub-rede formada pelos lugares $p_2$, $p_3$ e $p_4$ e pelas transições *a*, *b*, *c* e *d* formam um outro componente conservativo, onde observa-se a relação:

$$\forall M \in A(R,M_0), M(p_2)+M(p_3)+M(p_4)=3$$

.

Um invariante linear de lugar é uma restrição sobre a evolução de estados no sistema, que sempre se verifica, independente da marcação inicial.

**Componentes Repetitivos e Invariantes de Transição**

A sub-rede formada pelas transições *c* e *d* e os lugares $p_3$, $p_4$ e $p_5$, na figura 3.2 formam um *invariante de transição*, ou seja, o disparo de uma determinada seqüência, no caso $s = cd$, a partir da marcação inicial, leva de volta a mesma marcação. O conjunto das transições *c* e *d* do invariante formam um *componente repetitivo estacionário* da rede. Na mesma figura, a seqüência $s = ab$ é também um invariante de transição.

Existem, basicamente, três tipos de análises de RP. A primeira é a análise de *alcançabilidade*. Este método é constituído pela construção de uma árvore de cobertura de forma a enumerar o conjunto de estados alcançáveis pela RP. A partir desta árvore é possível uma verificação das boas propriedades da rede.

Um segundo tipo de análise, baseada nos componentes conservativos e repetitivos da rede, permite, em alguns casos, conclusões mais rápidas e simples, sem perda de informação.

Em um último caso, para redes estruturalmente mais complexas, provê-se *métodos de redução*. Este método permite que redes com um conjunto grande de marcações acessíveis seja reduzida a uma estrutura mais compacta, onde é possível a utilização das outras técnicas de análises.

## 3.3 Redes de Petri de Alto Nível

O nível de compactação de um RP ordinária não alcança índices satisfatórios, principalmente em sistemas complexos que envolvam tarefas de diferentes níveis de abstração, mesmo em uma representação algébrica. Além disso, a ficha como portadora de informação está limitada a valores numéricos inteiros. Esta condição impõe ao projetista restrições no que diz respeito à delimitação entre o controle e os dados de um sistema. A saída, normalmente, é a construção de redes onde a separação destes elementos é absolutamente inexistente. Como conseqüência pontual, processos semelhantes são representados de forma genérica, gerando uma estrutura mais compacta, porém com perda de informação, ou são representados individualmente, incluindo suas relações, em uma estrutura gráfica excessivamente grande, seja no tamanho da rede ou pelas interações existentes.

Este nível de compactação é vencido a partir da extensão das RP ordinárias naquilo que é conhecido como *RP de Alto Nível* (RPAN). As características básicas comuns aos modelos existentes incluem o conceito de dobramento de lugares e transições, e a extensão do poder de representação de fichas, de forma a portar informações do sistema. O que diferencia um modelo de outro é como estas extensões são implementadas. Especificamente, em alguns modelos, a idéia de hierarquia entre redes é introduzida, mesmo que este conceito possa ser generalizado.

Os três modelos mais conhecidos são as RP coloridas (Jensen, 1996), as RP Predicado/Transição (Genrich, 1991), a as RP a objetos, cujos diferentes tipos derivam de um modelo básico proposto em Sibertin-Blan (1985). Dentro de um contexto que sintetiza a proposta desta tese, a abordagem a ser apresentada, ou mais especificamente, a linguagem de especificação utilizada por ela, nada mais é do que a extensão da capacidade de expressão dos diferentes modelos de alto nível aos diferentes métodos de representação de conhecimento.

### 3.3.1 Redes de Petri Coloridas

Mais do que um modelo teórico, as RP Coloridas podem ser abordadas como uma linguagem completa para projeto, especificação, simulação, validação e implementação de grandes sistemas de

software. Prevê-se ainda sua utilização como um meio de comunicação entre homens e/ou máquinas baseado em um conjunto de regras formais.

Sua vantagem mais evidente em comparação com as redes ordinárias está no seu poder de representação de modelos em estruturas mais sucintas e de descrição gerenciável, sem perda de informação em análises formais.

A complexidade do modelo é distribuída entre a estrutura da rede, suas inscrições e declarações, incluindo-se a possibilidade de manipulação de estruturas de dados complexas.

Assim como no modelo ordinário, prevê-se diferentes formas de representação, neste caso limitado em duas. Uma representação baseada em *expressões* permite a apresentação da rede na forma de guardas, expressões de arco e outros recursos, utilizados principalmente na etapa de projeto da rede. Uma representação baseada em *funções* permite a apresentação desta mesma rede agora sob a forma de um conjunto de estruturas algébricas matriciais, de fundamental importância na etapa de análise de propriedades.

A definição formal de uma RP Colorida pode ser encontrada em Jensen (1996).

O método mais direto e prático para análise de uma RP Colorida é a *simulação*, servindo como meio de depuração do sistema e, até mesmo, como meio para verificação de propriedades. É normalmente suportado por alguma ferramenta computacional, mas não é suficiente como prova da correção do sistema. Sua aplicação dá-se, na maioria das vezes, na etapa de projeto, enquanto outros métodos formais são utilizado no processo de validação.

### 3.3.2 Redes Predicado/Transição

Comparada a outros modelos de alto nível, uma Rede Predicado/Transição (RPT) aumenta o poder de representação e concisão de seu modelo pela substituição da lógica proposicional como linguagem para as regras associadas as transições no modelo ordinário, pela lógica de primeira ordem, ou seja, utilizando regras passíveis de variáveis. Uma transição, portanto, é vista como uma regra, descrevendo, agora, não apenas um evento, mas um conjunto de eventos. O disparo das transições dependem da substituição das variáveis por valores. No caso das RP Colorida, a substituição era efetuada por cores.

Os lugares da rede representam as relações dinâmicas dos sistema, indicadas por predicados. Por este motivo, o modelo é chamado Predicado/Transição.

Ao contrário das redes coloridas, não há restrições quanto a dinâmica de fichas nos lugares, o que é tratado pela própria sintaxe da rede. As regras associadas às transições devem ser aplicadas a marcação vigente. O conjunto de substituições aptas às variáveis determinam o modo como a transição será disparada. O mecanismo que rege as substituições de variáveis pelas constantes é denominado *unificação*. Na medida em que o mecanismo de unificação provê mais de uma solução, gera-se uma situação de *conflito*. A escolha das fichas e transições disparadas é determinada pelas condições suplementares, especificadas, por exemplo, como fórmulas lógicas e operadores relacionados às variáveis da transição. Entretanto, estas condições só se aplicam às constantes associadas às fichas suscetíveis de serem deslocadas, quando substituídas às variáveis no disparo da transição. Da mesma forma, as ações associadas às transições devem ser escritas respeitando o conjunto de variáveis alocadas.

A definição formal da Rede Predicado/Transição pode ser encontrada em Genrich (1991).

A principal desvantagem da utilização de RPT dá-se na etapa de análise do modelo do sistema. A determinação, por exemplo, do conjunto de invariantes da rede envolve complexas combinações de equações simbólicas lineares, além de projeções e transformações para diferentes tipos de representações. Uma descrição detalhada dos métodos de análise podem ser encontradas em Genrich (1991).

### 3.3.3 Redes de Petri com Estruturas de Dados

As RP com Estruturas de Dados (Sibertin-Blan, 1985), também conhecida na literatura como RP a Objeto (Cardoso e Valette, 1997), têm como característica básica, uma perfeita adequação à modelagem de Sistemas de Informação de qualquer ordem, em que dados têm relevância destacada no comportamento do sistema. Seu modelo de ficha é generalizado, adequando-se, por exemplo, a maioria dos métodos de representação de conhecimento, fazendo deste modelo de alto nível altamente apto a descrição de SBC.

Contrariando a ordem dos outros modelos de alto-nível, as RP com estruturas de dados não detém sua força na concisão, mas sim na representação de estruturas mais complexas de informação.

O modelo de dado substitui fichas por entidades, similar ao modelo usado em Teoria de Base Dados. Além disso, associa-se a cada transição uma *pré-condição*, testando a entidade associada, e uma *ação* modificando-a.

O diferencial de uma RP com Estrutura de Dados (RPED) está, justamente, no modelo de dado utilizado. Este modelo modifica a concepção básica de ficha estabelecida até aqui, aumentando

seu poder de representação. Esta extensão é obtida graças a incorporação de três conceitos ao novo modelo: *propriedade*, *classe de entidade* e *entidade*.

Por *propriedade* entende-se a descrição de um atributo, um pedaço de informação, um item de um objeto. É definida por um nome e um intervalo de valores. Este intervalo pode ser definido por um conjunto de constantes pré-definidas (inteiros, valores booleanos, seqüência de caracteres, ...), ou uma classe de entidade. O valor de uma propriedade é um elemento ou um subconjunto do seu intervalo de valores.

Uma *entidade de classe* é um conjunto de propriedades, com um valor inicial definido associado a cada propriedade.

Uma *entidade* é um instância de sua classe de entidade. É definida por seu nome, sua classe de entidade, e valores adequados às propriedades de sua classe. Dentro de um RPED, as entidades é que determinam a marcação da rede.

De forma a computar valores, é possível a definição de *funções* no intervalo da propriedade em questão. Estas funções podem ser operações aritméticas padrão, ou funções específicas explicitamente definidas.

Este modelo de dado tem uma clara relação com o modelo de quadros proposto por Minsky (1975), e por conseqüência, uma relação direta com o modelo de objeto utilizado principalmente em Engenharia de Software (Pressman, 1995). A partir deste modelo observa-se duas vantagens em sua utilização.

Primeiramente, seu modelo conceitual está fundamentado em um paradigma estabelecido dentro, não só da Ciência da Computação, como em outras áreas. Esta relação com o modelo de objeto provê uma relação mais direta entre as estrutura de dados e de controle do sistema. Não obstante, este modelo é denominado RP a Objetos, de onde partem uma lista grande de extensões que derivam deste modelo original (Hong e Bae, 1998, Lakos, 1995, 1997, Bastide, 1995).

A segunda vantagem está na adaptabilidade do modelo a qualquer forma de representação do sistema. Em termos de uma base de conhecimento, por exemplo, observa-se que o modelo de dado proposto adequa-se a qualquer método de representação de conhecimento, permitindo que a mesma rede represente o sistema integralmente.

A definição formal do modelo é apresentada em Sibertin-Blan (1985).

## 3.4 Redes de Petri Hierárquicas

A inserção do conceito de hierarquia em RP vem de encontro a crítica que discorre sobre a sua falta de composicionalidade. Numa nova analogia às linguagens de programação, níveis hierárquicos em RP devem ser vistos como subrotinas, e o modo como estas introduz modularidade nos programas desenvolvidos.

Em redes hierárquicas, uma rede individual, como tratamos RP até agora, é denominada *página*. Basicamente existem cinco formas de relacionar redes individuais, chamadas de *construções hierárquicas*:

- substituição de transições;
- substituição de lugares;
- invocação de transições;
- fusão de lugares;
- fusão de transições.

Destas construções, as mais utilizadas são aquelas baseadas em subtituições de elementos da rede. As demais construções são detalhadamente apresentadas em Huber et al. (1990).

Informalmente, a *substituição de transição* pode ser compreendida como a substituição de uma transição por uma nova RP, permitindo uma descrição mais detalhada e complexa da atividade associada a transição substituída. Assim como a subrotina em um programa, permite-se, nesse caso, a integração da especificação mais simples com sua respectiva estrutura de detalhamento, gerando uma semântica significativa para o sistema combinado. O mesmo princípio é válido para as *substituições de lugares*, com a correspondente substituição.

Considerando o comportamento de uma RP hierárquica, cada página possui sua própria marcação. Permite-se que uma página simples substitua várias transições de substituição, o que indica a existência de várias *instâncias de página*, cada uma com sua marcação individual.

A *fusão de lugares* permite a especificação de um conjunto de lugares idênticos, porém representados em diferentes instâncias, de acordo com as necessidades do projeto. Isso implica que dado a soma ou remoção de uma ficha em um destes lugares, o mesmo deve acontecer para todos os demais. Os lugares de um conjunto de fusão podem pertencer a uma mesma página, ou a várias páginas diferentes.

A utilização de membros de um conjunto de fusão em uma mesma página dá-se mais no sentido de uma reestruturação gráfica, evitando, por exemplo, o cruzamento de um número excessivo de arcos. Porém, a utilização de conjuntos de fusões em diferentes páginas associa uma complexidade semântica a rede, aumentando seu poder de representação e capacidade modular.

Os conjuntos de fusão podem ser *globais*, de *página* ou de *instância*, diferenciando-se pelo seu contexto de expressão. A conjuntos de fusão global permite-se ter membros em diferentes páginas, enquanto nos demais apenas em páginas simples. Contudo, em uma mesma página unificam-se todas as instâncias de seus lugares resultando em um único lugar compartilhado semanticamente por todas as instâncias da página, enquanto em conjuntos de fusão de instância este compartilhamento serve apenas para a mesma instância de página.

Em (Jensen, 1996) há uma completa apresentação do processo de hierarquização de RP, especialmente aplicado aos modelo colorido. Entretanto, ele é perfeitamente aplicável a qualquer modelo de alto nível.

## 3.5 Trabalhos Relacionados

Esta seção apresenta uma análise sobre uma série de trabalhos que utilizam RP em algum dos níveis de desenvolvimento de SMA, seja em sua especificação social ou individual.

### 3.5.1 Especificação Multiagente

A aplicação mais natural para RP no desenvolvimento de sistemas inteligentes é a especificação de SMA, com o propósito de definir a coordenação entre os agentes. Nesse sentido, o objetivo é explorar as possibilidades de expressão de concorrências e paralelismos entre atividades, bem como os pontos de sincronização entre os agentes. Expandindo estas possibilidades, é possível também expressar planos multiagentes, de forma a explorar a execução de ações paralelas no sistema.

Em Holvoet e Verbaeten (1996) as RP são utilizadas para especificar as relações entre agentes de um ambiente. Isto acontece por meio de transições comuns entre agentes, que servem como pontos de sincronização de ações. Cada agente corresponde a uma rede, enquanto uma especificação inter-redes descreve a estrutura de cooperação. Este modelo, segundo os autores, é aplicável a sistemas abertos, flexíveis, altamente concorrentes e complexos, baseados em componentes. Segundo o modelo, as redes de cada agente mantêm partes encapsuladas de sua descrição, e operadores de

composição especificam suas relações. Estas composições são relações entre transições, e podem ser de dois tipos. Uma *transição sincronizada* é uma relação entre pares de transição, enquanto *blocos de sincronização* permitem a conjunção ou disjunção entre transições sincronizadas. Assim, semanticamente, um bloco de sincronização impõe uma restrição ao disparo de uma transição. O modelo não faz qualquer restrição quanto à RP utilizada, ficando ao critério do desenvolvedor escolher o modelo que melhor se adapta ao problema. Contudo, a dificuldade do modelo está na pouca flexibilidade quanto ao modo de se projetar o sistema, na medida em que ele deve ser todo baseado em pontos de sincronização. Além disso, para grandes sistemas, o modelo final pode se tornar ingerenciável, dado o tamanho da rede, uma vez que não se prevê o uso de hierarquização entre as redes.

Em Moldt e Wienberg (1997), RP Coloridas são utilizadas para especificar SMA, considerando uma sociedade de agentes autônomos, inteligentes e comunicantes. As RP, além de especificar o sistema global, servem para simular e implementar os agentes e suas relações. O autor propõe a construção de um sistema a partir de dois pontos de vista: o da sociedade de agentes e do agente. O princípio básico é a aplicação de Programação Orientada a Agentes (Shoham, 1993) por meio de RP. Cada agente usa um número arbitrário de provadores de teorema, encontrando soluções para requisições feitas a base de conhecimento. Assim, um agente pode tratar qualquer número de requisições ao mesmo tempo. Pela aplicação de Programação Orientada a Agentes, ações são representadas por predicados que tornam-se verdade. Neste caso, a RP do agente espelha esta idéia associando uma ação a uma transição, tornando-a habilitada no momento em que o predicado for verdade. A sociedade de agentes é vista como uma coleção de objetos com identidades diferenciadas. A relação entre objetos é feita por manipuladores de mensagens, e a classe é representada por RP. Contudo, o modelo não visa especificar o conhecimento do sistema, seja social ou individual. O objetivo é estruturar os mecanismos necessários à implementação dos agentes e da sociedade, de forma a garantir a correta coordenação entre seus membros, sem a necessidade de um relógio de sincronização.

Uma forma de definir uma estratégia coordenada de ações em um sistema multiagente pode ser por intermédio de planejamento multiagente. Em de Almeida et al. (2005) é proposto um modelo para verificação e validação de planos multiagentes baseado em diretrizes de modelagem e verificação. Nesse sentido, integram-se técnicas como Mapas de Seqüências de Caracteres (do inglês "*Message Sequence Charts*") (MSC) e RP Coloridas Hierárquicas. Fundamentalmente, o autor propõe uma série de diretrizes de modelagem que, uma vez aplicadas, geram um modelo formal do sistema. Este, por sua vez, está sujeito a uma série de diretrizes de verificação, que aplicadas, determinam problemas de projeto, ou ainda, podem ser utilizadas para melhorá-lo. O processo de análise é implementado por modelos de RP Coloridas Hierárquicas aplicadas a ferramenta Design/CPN (Jensen, 1996). Este

processo gera automaticamente um conjunto de MSC's possibilitando a checagem do modelo. Os MSC's gerados são úteis para a observação de diferentes traços de execução para os modelos, e permitem ainda uma abstração para a simulação da rede. Um MSC gerado automaticamente durante uma simulação é utilizado para definir predicados relacionados ao modelo, necessários à execução da checagem. Sendo assim, esses predicados são úteis para provar certas propriedades desejadas a um dado plano. O inconveniente deste método é ausência de um modelo global do sistema, na medida em que a sincronização é obtida pela devida correlação de predicados. Toda a especificação é feita em torno das entidades fundamentais do sistema, ou seja, os agentes, fazendo com que a descrição do sistema global surja da integração destes modelos.

Uma alternativa à abordagem anterior é proposta por Xu et al. (2003) e Xu et al. (2002). Neste caso, modela-se um problema multiagente pela representação das ações possíveis dos agentes como transições em uma Rede Predicado/Transição (PrT). A especificação de planos provê os agentes com um modelo mental compartilhado de como eles devem interagir para alcançar uma meta comum. Para analisar um modelo de Rede PrT multiagente, adaptam-se grafos de planejamento, como GRAPHPLAN (Blum e Furst, 1995), a um estrutura compacta para análise de alcançabilidade, coerente com a semântica concorrente. Assim, é possível determinar se ações paralelas especificadas em planos multiagentes podem ser executadas em paralelo, e se os planos podem alcançar suas metas analisando as relações de dependência entre as transições da modelo. A escolha de Redes PrT deve-se à relação direta entre a expressividade de suas proposições e a representação em lógica de primeira ordem, comum as análises de planejamento. O modelo é construído a partir das ações executáveis pelos agentes bem como pela representação do ambiente. As ações são especificadas por transições com pré-condições, pós-condições e inscrições, enquanto o ambiente é representado por predicados. Desta forma, transforma-se um problema de planejamento em um problema de alcançabilidade em Redes PrT. Este modelo, assim como os demais, permite a visualização do sistema multiagente pela composição das partes individuais, ou seja, a relação entre os agentes. Entretanto, não há uma representação global abstrata do planejamento, focando-se nas ações necessárias ao cumprimento das tarefas envolvidas no plano, sem distinção de quem deverá cumpri-las. Não obstante, ao utilizar Redes PrT como modelo base, os agentes estão restritos a um representação de conhecimento em lógica de primeira ordem.

Um exemplo de aplicação de RP na especificação social de um sistema multiagente pode ser verificado em Miranda e Perkusich (1999). Neste caso, o autor propõe a utilização de RP Colorida para a especificação do nível social dentro da arquitetura MATHEMA (de Barros Costa e Perkusich, 1996). O MATHEMA é um ambiente de aprendizado interativo, baseado na abordagem multiagente. Entre seus componentes, o principal é uma sociedade de agentes tutores inteligentes que interagem por in-

termédio de um interface com um dado aprendiz. Os agentes interagem em cooperação no sentido de promover atividades de ensino/aprendizagem em um ambiente computacional de aprendizado interativo. Na arquitetura MATHEMA, cada agente é definido de acordo com um arquitetura hierárquica, sob um Sistema Social responsável por promover as interações cooperativas entre agentes. A análise e verificação deste Sistema Social do ambiente MATHEMA é viabilizado por meio de RP Coloridas, modeladas e simuladas pela ferramenta Design/CPN. Para tanto, grafos de ocorrência são gerados e analisados no maior número de cenários possíveis, para garantir a maior correção possível.

De um modo geral, as abordagens citadas partem para uma especificação do sistema multiagente a partir do somatório dos conhecimentos individuais dos agentes, sem qualquer distinção quanto ao conteúdo. Desta forma, o sistema global é descrito sem uma instância social abstrata, que determina o conjunto de metas a serem cumpridas independente dos agentes do processo. A coordenação entre os agentes é obtida por intermédio de ações comuns, que podem ser ações que devem ser executadas por múltiplos agentes, ou ações de outros agentes que permitem a continuação de um plano de um outro agente. Seja por exemplo, a execução de uma jogada ensaiada em uma equipe de futebol. Segundo os modelos descritos acima, a descrição do sistema multiagente é feita a partir da especificação do que cada agente deve fazer para a realização da jogada, sem que necessariamente, cada um tenha conhecimento do plano que a gerou. Esta abordagem, por conseqüência, apresenta um grau de flexibilidade mínimo, diminuindo a robustez do sistema. Uma alternativa a esta abordagem seria a apresentação de um plano social, em que ficaria determinado o conjunto de sub-metas a serem realizadas para que a meta principal seja cumprida. Neste caso, cada agente, de acordo com seu papel dentro do sistema e o conjunto de ações de que dispõe, deliberaria autonomamente qual a sua contribuição. Desta forma tem-se um sistema mais flexível e robusto.

### 3.5.2 Especificação Individual

Há diversas formas de se utilizar as RP na especificação e modelagem de agentes e sistemas inteligentes em geral, dentro de um contexto de conhecimento e construção individual. Sua aplicação pode ser direta, por exemplo, durante a construção de alguns dos componentes de sua arquitetura e suas relações, ou ainda, pode fazer parte de uma metodologia e/ou abordagem de desenvolvimento formal, auxiliando na correção do modelo. A análise da literatura corrente sobre a utilização das RP neste contexto aponta para três abordagens principais:

- como método de representação de conhecimento, especialmente para conhecimento nebuloso, auxiliando na construção de sistemas de regras sujeitos a imprecisão;

- como método de verificação e validação de bases de conhecimento, por intermédio de uma análise estrutural destas, buscando minimizar a ocorrência de anomalias, que podem ser fontes de futuros erros do sistema;
- incorporado a abordagens de desenvolvimento, permitindo uma análise de especificação, buscando uma certa correção semântica a partir dos requisitos do sistema;

Na primeira abordagem, as RP, em uma das suas várias extensões utilizadas para representar sistemas nebulosos (Cardoso e Valette, 1997, cap. 8), servem como uma linguagem de especificação gráfica para a representação de regras de produção nebulosas. O formato destas regras de produção, bem como o modelo correspondente em RP, é basicamente o mesmo em quase todos os trabalhos existentes na literatura. O que varia é o objetivo a ser alcançado a partir desta representação. Na sua forma mais geral, uma base de regras $R$ é formada por um conjunto de regras $R = \{R_1, R_2, \ldots, R_n\}$, cuja formulação geral da i-ésima regra é:

$$R_i: \text{ IF } a \text{ THEN } c \ \ (CF = \mu), Th, w$$

onde $a = \langle a_1, a_2, \ldots, a_n \rangle$ é a conjunto de antecedentes da regras, que compreende uma ou mais proposições conectadas por *AND* ou *OR*, $c$ é a proposição conseqüente, $\mu$ é o fator de certeza da regra, $Th$ é o limiar, e $w$ é o peso associado a proposição antecedente. Sendo assim, uma regra $R_i$ é representada pela RP nebulosa da figura 3.3.

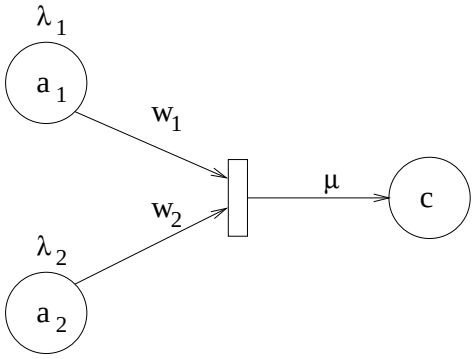


**Figura 3.3:** *Representação de uma regra de produção nebulosa*

Quanto a este tipo de representação, o trabalho pioneiro foi introduzido por Looney (1988), em que RP são utilizadas na representação de raciocínio nebuloso transformado em vetores de estado nebuloso. O resultado é uma rede neural onde uma transição é vista como um neurônio e lugares como condição.

Em Chen et al. (1990), RP nebulosas já são vistas como uma representação de conhecimento. Pelo modelo proposto, é possível quantificar a relação antecedente-conseqüente entre duas proposições $d_s$ e $d_j$. Para tanto, um algoritmo de raciocínio nebuloso baseado no modelo correspondente em RP gera uma análise de alcançabilidade, obtendo todos os caminhos possíveis entre as proposições existentes, e conseqüentemente, a relação entre elas.

Em Li et al. (2000), propõe-se uma estrutura dinâmica de inferência ajustável, adequada a variação da cognição humana. Fundamentalmente, trata-se de um modelo em RP nebulosa adaptativa, sujeita a variação dinâmica de seus pesos, fatores de certeza e limiares. Desta forma, tem-se um modelo com capacidade de aprendizado, assim como as redes neurais. Assim como estas, a RP está sujeita a um processo de treinamento, nos moldes do *Back Propagation*, onde seus parâmetros são aprendidos, para posterior aplicação no sistema. A RP, por sua vez, é utilizada apenas para expressar o sistema de regras nebulosas, sem que implique necessariamente em algum tipo de análise. Já em Li e Yu (2001) é proposto uma extensão a este modelo, onde a RP tem a função de auxiliar no tratamento da complexidade de grandes sistemas de conhecimento. Para isso, um modelo de RP nebulosa orientada a objeto permite dividir a complexidade de um sistema de conhecimento em vários subsistemas. O sistema é organizado por diferentes objetos, relativos aos tipos de conhecimento utilizados e níveis de abstração estruturados. Assim, um modelo de objeto de um conhecimento não trata sobre quais regras devem ser processadas, mas sim sobre a estrutura dos subsistemas e como eles se comunicam entre si. Do ponto de vista da orientação a objeto, um sistema é composto por um número finito de objetos e suas relações. Cada objeto é descrito por uma RP e as relações entre objetos são obtidas por lugares de mensagens comuns entre as subredes. Novamente, o objetivo não está na análise do modelo global, mas apenas na simplificação da estruturação do conhecimento de grandes sistemas.

Ainda tratando de sistemas nebulosos, Koriem (2000) propõe não apenas uma abordagem para modelagem de sistemas deste tipo, mas também um processo de análise a partir da especificação em RP nebulosas. Neste caso, além de um modelo que permite a especificação de uma base de conhecimento nebulosa, a abordagem propõe uma metodologia de análise para a checagem da consistência do sistema projetado. Assim, mais do que uma representação de conhecimento, as RP são vistas como uma linguagem de especificação visual. A análise de consistência restringe-se a detecção de anomalias tais como redundâncias, conflitos e bloqueios mortais. A verificação dá-se por intermédio de análises de propriedades estruturais e comportamentais da RP. Seu ponto fraco, porém, está na ausência de mecanismos para o tratamento de sistemas mais complexos, com bases de conhecimento com grande número de regras e variáveis. Neste caso, o sistema torna-se intratável por este tipo de abordagem.

Seguindo a classificação da literatura do início da seção, o trabalho citado no parágrafo anterior encontra-se na fronteira entre a primeira e a segunda abordagem quanto a utilização de RP em Engenharia do Conhecimento. Assim, além de utilizá-la como formalismo para representação de conhecimento nebuloso, ele ainda propõe uma metodologia de verificação da base desenvolvida. Entretanto, esta última característica já vinha sendo investigada no âmbito de sistemas de regras clássicas.

Uma das diferenças entre a primeira abordagem, que utiliza RP como formalismos de representação de conhecimento, e a segunda, que as utiliza num contexto de verificação e validação de bases de conhecimento, é o modo como a rede é inserida no modelo. No primeiro caso, a rede é utilizada durante a fase de especificação do sistema, ou seja, ela não só antecede a construção da base de conhecimento como esta é construída a partir da rede modelada. No segundo caso, o processo de verificação e validação da maioria das citações existentes, já pressupõe a existência de uma base que deve ser convertida no respectivo modelo em RP. A partir disso, a rede é analisa segundo a metodologia utilizada.

Seguindo esta linha de trabalho, Nazareth (1993) investiga a aplicabilidade de RP na verificação de sistemas baseados em regras. Nesta abordagem, um sistema baseado em regras é transformado em uma RP ordinária, tornando o processo de verificação em um problema de alcançabilidade de estados específicos na rede. O escopo de verificação está restrito a problemas estruturais, ou anomalias, como redundâncias, conflitos, circularidades e ausência de conhecimento, sem considerar erros semânticos. Utilizando redes ordinárias, cada regra é representada por uma transição, de tal forma que seus lugares de entrada representam as premissas da regra, e os lugares de saída representam as conclusões. As fichas, por sua vez, representam a validade dos fatos correspondentes aos lugares. Desta forma, o modelo restringe-se a uma representação em lógica proposicional. A verificação obtida por intermédio de uma análise de alcançabilidade pode ser viabilizada de dois modos. No mais direto, é gerada uma árvore de alcançabilidade onde é possível visualizar o conjunto de marcações acessíveis da rede dada uma marcação inicial. As anomalias correspondem a não verificação das chamadas boas propriedades da rede. Entretanto, este tipo de análise torna-se inconveniente a partir do crescimento do sistema, gerando, conseqüentemente, uma árvore de alcançabilidade extremamente grande. A alternativa é a utilização da representação em matrizes, cuja análise é viabilizada pela determinação dos componentes conservativos da rede. Todavia, este método não é decidível quanto as propriedades da rede.

Na mesma linha do modelo anterior, está o trabalho de Wu e Lee (1997). A contribuição está na generalização do modelo de tal forma que se permita a representação de variáveis e sentenças negativas. Para isso, é utilizado um modelo de RP Colorida. As cores da rede são distribuídas com o tipo de predicado associado a um lugar. Desta forma, tem-se cores específicas para sentenças

variáveis, negativas e afirmativas. Quanto ao processo de análise, ele é praticamente o mesmo descrito por Nazareth (1993). Entretanto, o modelo agora permite uma representação de conhecimento não apenas em lógica proposicional, mas em lógica de primeira ordem. De qualquer forma, as questões relativas ao desenvolvimento de sistemas mais complexos continuam sem uma abordagem adequada.

Uma proposta alternativa quanto ao método de verificação é apresentada por Zhang e Nguyen (1994). Assim como nos modelos anteriores, uma RP é construída a partir da base de conhecimento do sistema e o objetivo é a detecção de anomalias, ou falhas estruturais da base. Entretanto, o modelo utiliza Redes Predicado/Transição, sobre lógica de primeira ordem. Esta rede é, então, editada em uma ferramenta, o PREPARE, que implementa o processo de verificação baseado em uma análise de padrões sintáticos de erros. Este reconhecimento de padrões sintáticos é baseado em conceitos da teoria de linguagens formais, onde os padrões de uma classe são representados como expressões de uma linguagem. Assim, o processo de reconhecimento de padrões torna-se o processo de determinação de que linguagem uma expressão pertence. Para isso, define-se uma série de padrões associados a cada tipo de anomalia a ser detectada. A partir da correta modelagem da rede, o PREPARE transforma-a em uma linguagem específica $L(G)$, onde $G$ é a Rede Predicado/Transição. Se alguns dos padrões definidos como anomalias pertencem a linguagem, o PREPARE identifica a anomalia em análise. Contudo, assim como os demais modelos apresentados, a metodologia proposta não está apta a tratar de grandes sistemas, além de estar limitada a representação em lógica de primeira ordem.

Quanto a especificação individual de sistemas de conhecimento, as RP podem ainda serem utilizadas dentro de metodologias formais para o desenvolvimento deste tipo de software. Neste caso, uma RP é instanciada no sentido de estabelecer uma linguagem de especificação gráfica para algumas das etapas anteriores a implementação do sistema. Além disso, é possível a sua utilização não só na análise estrutural do sistema, mas também quanto a uma análise semântica, baseado no seu conjunto de requisitos.

Em Lee e Lai (2002) é proposto o uso de RP de Alto Nível para a verificação de estruturas de tarefa, checando a consistência do modelo e as especificações dos processos. Um SBC especificado por estruturas de tarefa possui dois componentes principais: o *modelo*, relativo às propriedades estáticas, e os *processos*, relativo às propriedades dinâmicas do sistema. Entre as propriedades estáticas está o modelo do domínio e entre as propriedades dinâmicas estão as transições entre estados do sistema (funcionalidade) e as seqüências de tarefas. Por sua vez, uma estrutura de tarefa pode ser compreendida por uma seqüência de estados, ou mesmo, como unidade funcional descrita por pré-condições, proteções e pós-condições. Estas estruturas são, então, utilizadas em uma rede de restrições cuja função é descrever as mudanças de estado de um sistema. Neste trabalho, o autor propõe o uso de

RP de Alto Nível na especificação destas estruturas de tarefa. Assim, uma tarefa é vista como uma transição de estado em uma RP, e seus lugares de entrada e saída correspondem as suas pré e pós-condições. É possível ainda a decomposição da tarefa utilizando RP Hierárquicas. Desta forma, no processo de verificação, a RP está restrita a análise de relações entre tarefas. Seu papel é determinar se o conjunto de estados de uma tarefa é alcançável. A partir desta análise, implementa-se um processo de verificação das especificações do modelo baseado na rede de restrições do componente dinâmico do sistema, garantindo a consistência entre os diversos níveis.

Goc et al. (2002) apresenta uma proposta semelhante, onde um modelo conceitual em KADS faz o papel de estruturas de tarefa do trabalho anterior. Desta forma, o método propõe a transformação do modelo conceitual de um sistema em KADS em um modelo operacional em RP de Alto-Nível. Este modelo operacional permite a simulação da dinâmica do sistema e posteriormente a validação de sua especificação. Considerando a metodologia KADS, particularmente a fase de especificação, diz-se que um modelo conceitual, de natureza informal, precede o modelo de especificação. Assim, a partir deste último, parte-se para o modelo de projeto. Basicamente, um modelo conceitual estrutura-se em três níveis: domínio, inferência e tarefa[1]. Estes níveis são descritos por RP de Alto Nível de acordo com um mapeamento pré-definido, incluindo mecanismos de relação entre elas. A simulação da rede representa o modelo de especificação funcional do sistema, permitindo uma análise do seu comportamento. Ao contrário dos trabalhos anteriores, este modelo prevê a manipulação de grandes sistemas pela decomposição em múltiplos níveis, utilizando estruturas hierárquicas, além de permitir a manipulação de múltiplos formalismos de representação de conhecimento. Todavia, a especificação do sistema não inclui mecanismos formais para a sua inclusão em um ambiente social.

Em síntese, as principais vantagens dos métodos aqui apresentados são:

- o rigor formal dos modelos, no sentido de garantir um processo de análise que detecte o maior número de erros possíveis;
- mecanismos de análise automáticos, baseados na bagagem teórica existente sobre RP.

Contudo, em sua maioria, os modelos não são aptos a manipular grandes bases de conhecimento, gerando processos de análise intratáveis nestes casos. Além disso, estão restritos a representações de conhecimento baseados apenas em lógicas de diferentes tipos.

[1] c.f. item 2.5.3

### 3.5.3 Conclusões da Literatura

Tendo em vista as restrições impostas por problemas tratados por SMA atualmente, os modelos apresentados falham especialmente no que se refere ao tratamento de sistemas que demandam híbridos de soluções além de uma diversidade de tipos de conhecimento. O rigor obtido por um modelo quanto aos processos de verificação é alcançado ao custo de sua aplicabilidade, geralmente, em sistemas de baixa complexidade. Além disso, via de regra, estão associados a representações de conhecimento baseadas apenas nos diferentes tipos de lógica. Já as estratégias para especificação multiagente não apresentam meios para a descrição do sistema global em um nível de abstração apenas social. Os modelos apresentados descrevem o sistema multiagente a partir do somatório das suas entidades individuais, no caso, seus agentes.

Os problemas atuais demandam abordagens aptas a manipular grande quantidade de conhecimento, nos mais diversos aspectos, seja planejamento, coordenação de ações ou classificação de estados do ambiente. Seria também interessante, a representação do sistema global a partir de uma especificação abstraída das atividades e tarefas individuais, mas focado apenas no conjunto de ações necessárias para a realização das metas do ambiente. Nesse sentido, os agentes deliberariam de forma autônoma suas metas e ações individuais, independente das metas e ações dos demais agentes, escolhidas ao encontro da meta global do sistema. Não obstante, seria interessante que estes modelos permitissem a especificação do sistema em seu nível de conhecimento, a partir de estruturas genéricas viabilizando a utilização de diversos formalismos de conhecimento, e não apenas as diferentes formas de lógica.

## 3.6 Considerações Finais

As RP, tema deste capítulo, devem ser vistas como o núcleo da abordagem para especificação de conhecimento, ou seja, parte da solução que é proposta deste trabalho. É por meio delas que descreve-se as relações entre o nível multiagente e o nível individual, bem como as relações entre os diferentes níveis de decisão interagentes de um sistema. Não obstante, as RP têm ainda o poder de especificar, dentro de cada nível, o conhecimento existente e as sua relações, apresentando-se, desta forma, como uma ferramenta única para a especificação, modelagem, análise e validação de SMA cognitivos, desde sua concepção social, até o nível de atuação no ambiente.

Para que esta abordagem seja possível, um sistema multiagente cognitivo deve ser caracterizado como um *sistema a eventos discretos*, classe de sistemas aos quais uma RP se propõe a tratar. Intuitivamente, um SED descreve sistemas onde as mudanças de estados são dirigidas, não pelo tempo,

mas sim por eventos. Deste modo, um sistema é descrito pelos *eventos*, *atividades* e *processos* que o compõem. Dentre as suas características está a possibilidade de representação de atividades que envolvam *cooperação*, *competição*, e *paralelismo*.

Dada a natureza de um sistema multiagente, as RP adequam-se perfeitamente a sua representação. Em sua concepção básica, um sistema multiagente congrega uma sociedade de agentes com o propósito de *cooperar* no sentido de atender as demandas de um ambiente. Este processo de cooperação deve ser estabelecido pela *sincronização* de atividades, sejam elas *seqüenciais* ou *paralelas*, entre os agentes constituintes da sociedade. Já internamente ao agente, as relações entre os níveis de decisão, sejam elas de cooperação, paralelas, ou até mesmo, de competição entre recursos, também encontram nas RP uma ferramenta idealizada para a sua especificação.

Contudo as vantagens da utilização de RP não se resumem aos mecanismos de modelagem e especificação de sistemas, incluem também os métodos de análise e validação. Inicialmente, em redes ordinárias, há uma série de métodos que buscam a verificação das chamadas *boas propriedades* de um sistema: *vivacidade*, *limitabilidade* e *reiniciabilidade*. A análise é feita a partir de dois modelos básico: uma análise de comportamento, que depende diretamente da marcação inicial da rede, e uma análise estrutural que independe de $M_0$.

Porém, a complexidade estrutural de alguns sistemas não encontra suporte suficiente no modelo ordinário de RP. A limitação quanto ao poder de representação de uma ficha restringe uma eventual necessidade de especificação conjunta de dados e controle do sistema. Não obstante, esta dificuldade é transposta, quando possível, pela modelagem de uma rede grande, implicando um difícil, e até mesmo intratável, processo de análise. Nestes casos sugere-se a utilização das redes de alto-nível, cuja vantagem está na associação de informação à ficha o que implica a capacidade de dobramento de alguns elementos da rede. Redes de alto nível prevêem a utilização de variáveis e substituições de forma a simplificar o modelo final. A forma como se dão estas simplificações dos elementos, e o tipo de estrutura de informação que se associa as fichas é o diferencial entre os modelos de alto nível.

A simplificação obtida nos modelos de alto nível agregam uma complexidade extra aos mecanismos de análise. Supondo agora a utilização de variáveis, e as respectivas substituições que as instanciam, uma análise comportamental deve levar em consideração diferentes conjuntos de parâmetros. Para cada conjunto de parâmetros utilizados, uma nova análise deve ser gerada. Para o caso de uma análise estrutural, a dificuldade de análise é diretamente proporcional á complexidade associada a estrutura de dados utilizada. É possível afirmar, portanto, que a simplicidade obtida no modelo de alto nível é compensada pelo complexo trabalho de análise feito na etapa seguinte.

Para completar o conjunto de mecanismos que auxiliam na especificação de sistemas por meio

de RP, é possível a utilização de redes hierárquicas. Para isso, provê-se uma série de construções hierárquicas que viabilizam a relação formal entre redes de diferentes contextos de abstração, agregando modularidade ao modelo.

Inserindo as RP no contexto da tese, a literatura apresenta um grande número de trabalhos que a utilizam, seja na especificação do sistema multiagente, ou na especificação individual do conhecimento dos agentes. Na especificação social, os trabalhos apresentados têm em comum a especificação a partir da representação das capacidades individuais dos agentes. O modelo global é descrito pela sincronização de atividades específicas entre os agentes, e não por um modelo de organização ou coordenação coletiva. Por outro lado, no nível de especificação individual, os trabalhos existentes utilizam RP como mecanismo para verificação e validação estrutural de sistemas de regras. São essencialmente voltados para sistemas cuja representação de conhecimento é baseada em algum tipo de lógica, proposicional, de primeira ordem ou até mesmo nebulosa. Não obstante, são aplicáveis a sistemas com poucas regras, ou com baixo encadeamento entre regras, dada a limitação de uma RP, mesmo em um modelo de alto nível, para o gerenciamento de grandes sistemas. Observa-se, de maneira geral, a ausência de mecanismos que prevejam a manipulação de diferentes níveis de abstração do conhecimento de um sistema. E finalmente, não há referências a modelos que integrem os mecanismos de especificação social e individual, de forma que um mesmo modelo de rede conceba todo o sistema, em seus diversos níveis.

É viável, portanto, a utilização de RP no auxílio à construção de SMA cognitivos. A proposta é utilizá-la na etapa de aquisição de conhecimento, portanto, na especificação do planejamento social, e das bases de regras dos agentes. Para este fim, as RP justificam-se por duas vias distintas. Primeiro, pode servir como uma linguagem comum ao engenheiro de conhecimento e ao especialista no domínio, prestando-se como um mecanismo gráfico-cognitivo único. Por outro lado, auxilia na estruturação do domínio, provendo meios formais para a especificação dos seus objetos e relações. A complexidade do domínio pode ainda ser vencida pela conveniente estruturação de dados, e ainda, pela divisão hierárquica de designações cognitivas, sejam sociais ou individuais.

# Capítulo 4

# Especificação de Conhecimento para Sistemas Multiagentes Cognitivos

## 4.1 Introdução

Após a revisão da literatura sobre o tema deste trabalho - Engenharia do Conhecimento em Sistemas Multiagentes (SMA) - e sobre a base formal da solução sugerida - Redes de Petri (RP) - este capítulo descreve a abordagem proposta para especificação de conhecimento em SMA Cognitivos, utilizando um modelo específico de RP de Alto Nível como linguagem de especificação.

Esta abordagem estrutura-se sobre duas visões de desenvolvimento do sistema: a primeira é uma visão externa, em que o sistema é especificado no seu nível social e os respectivos níveis individuais do conhecimento. A segunda é uma visão interna, em que o conhecimento de cada um dos níveis correspondentes a visão externa é refinado segundo um ciclo de conceitualização-operacionalização-implementação, de acordo com as diretrizes básicas de um processo de aquisição de conhecimento. Este ciclo é implementado com o auxílio de uma linguagem de especificação baseada em um modelo específico de RP. Com as RP é possível estruturar o conhecimento em múltiplos níveis de abstração, segundo a visão externa do sistema, além de prover mecanismos genéricos para a utilização dos diferentes formalismos de representação de conhecimento existentes.

Este capítulo é estruturado da seguinte forma. A seção 4.2 apresenta um modelo de cognição com base nas funções executivas do cérebro, e algumas arquiteturas de Sistemas Baseado em Conhecimento (SBC) aptas a sustentar tal modelo. A seguir, a seção 4.3 descreve as diferentes visões da abordagem de especificação de conhecimento para o desenvolvimento de SMA Cognitivos. Na

seção 4.4 descreve-se formalmente o modelo de RP que possibilita a especificação dos agentes do sistema, considerando uma estruturação de conhecimento em múltiplos níveis de abstração. A seção 4.5 apresenta uma discussão sobre a abordagem proposta dentro do contexto de utilização das RP, estabelecida no item 3.5, e dos formalismos existentes para construção de SBC e SMA apresentados nos itens 2.3 e 2.5. Finalmente, as conclusões do capítulo são apresentadas na seção 4.6.

## 4.2 Estruturação de Conhecimento por intermédio de Hierarquização Cognitiva

A abordagem de especificação de conhecimento a ser apresentada na próxima seção sustenta-se na idéia de que o conhecimento de um domínio pode ser estruturado em múltiplos níveis de abstração. Nesta seção apresenta-se um modelo de cognição e os mecanismos necessários à arquitetura de uma abordagem baseada em conhecimento que permite esta concepção, a partir de uma estruturação dos mecanismos sociais.

Em geral, o conhecimento a respeito de um domínio pode ser estruturado em diferentes aspectos, pois para resolver um problema, o especialista utiliza diferentes aspectos da inteligência. Esta modularização do conhecimento encontra um referencial teórico em um modelo de cognição genérico baseado nas funções executivas do cérebro (Barkley, 1997). O mérito deste modelo é estabelecer quais são estes aspectos e a hierarquia subjacente.

As funções executivas estão associadas aos lóbulos frontais do cérebro, considerados a última etapa da evolução do sistema nervoso central, somente presente nos seres humanos (Goldberg, 2002). Dentre estas funções estão aquelas relativas à *intencionalidade*, *proposicionalidade* e *tomada de decisões complexas*.

Segundo o modelo, considera-se que os aspectos fundamentais da inteligência, que compõem as funções executivas, são, pela ordem hierárquica:

- **Planejamento**: onde o agente, baseado em seu conjunto de modelos mentais do mundo, estabelece suas prioridades, definindo suas metas no ambiente de atuação.
- **Coordenação de Ações**: uma vez estabelecida uma meta, o agente deve identificar a melhor seqüência e coordenação de ações no sentido de atendê-la.
- **Atividades Motoras**: utilização dos órgãos sensoriais e motores com o objetivo de simultaneamente atualizar seu modelo interno do mundo e gerar ações no ambiente.

As funções executivas têm papel central na formação de metas e no estabelecimento das respectivas prioridades, além de criar planos de ações para alcançar tais metas. Além disso, selecionam as habilidades cognitivas requeridas para implementar os planos, coordenam estas habilidades, e as aplicam na ordem correta. Finalmente, estas funções possuem também a responsabilidade de avaliar as ações, como sucessos ou fracassos, de acordo com as intenções do agente.

Entretanto, segundo este modelo de cognição, o módulo de planejamento é inerente a socialização, ou seja, ele surge na medida em que o indivíduo inicie interações com os demais membros de uma sociedade. Assim, este indivíduo passa a ter um *papel* a desempenhar, de acordo com suas habilidades. Entende-se que assim, todo o planejamento de ações individuais tem como origem a socialização do indivíduo, na busca por atingir as metas desta sociedade cumprindo as funções do papel que ele exerce dentro da mesma.

Inspirado no modelo proposto por Barkley (1997), propõe-se um modelo ajustado às necessidades de um processo de aquisição de conhecimento, de forma a estruturar o conhecimento em módulos diferenciados segundo o aspecto da inteligência abordado. Este modelo é semelhante ao proposto por Bittencourt (1997) e sua implementação computacional, o agente autônomo concorrente (da Costa e Bittencourt, 1999b). [1]

De acordo com o modelo proposto, as interações sociais dão-se a partir do reconhecimento, por parte do agente, dos objetivos do grupo no qual ele está inserido. Estes objetivos sociais são, então, instanciados pelo módulo de planejamento, de forma que o agente possa reconhecer aquilo que é de sua responsabilidade dentro do ambiente e dentro do planejamento estabelecido. Portanto, o planejamento do agente tem como ponto de partida a identificação do plano social de ações, e a partir disso, a correspondente definição de metas e prioridades internas. Outrossim, este modelo de cognição está de acordo com a teoria que afirma que a inteligência tem relação direta com o contexto individual (Ferber, 1999).

A abordagem de especificação de conhecimento proposta adota uma divisão genérica do conhecimento de acordo com os aspectos da inteligência necessários à solução de um problema. Este modelo genérico de cognição é descrito na figura 4.1.

A modularidade do modelo é baseada na especificação formal do subconjunto de informações trocadas entre os módulos. O sistema de planejamento define a meta atual do agente. O sistema de coordenação, por sua vez, define uma seqüência de ações para alcançar a meta especificada. A forma como cada módulo toma suas decisões, seja baseado em informações provenientes diretamente do

[1] c.f. item 5.3

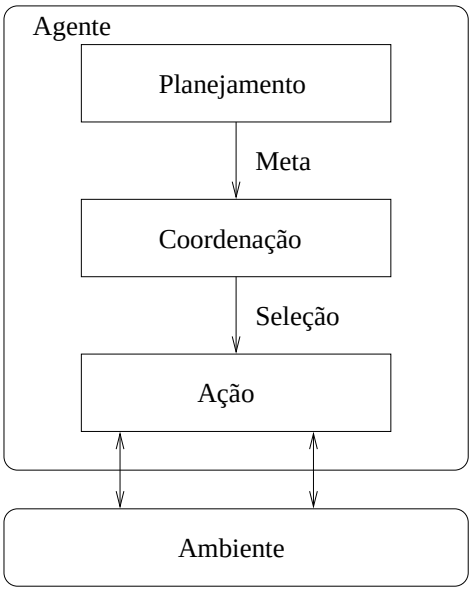


**Figura 4.1:** *Modelo de Aquisição de Conhecimento Genérico*

ambiente, filtradas pelos próprios módulos inferiores, ou por qualquer outro módulo de uma arquitetura específica, é independente do modelo.

Entretanto, é necessário definir uma arquitetura genérica que suporte este modelo de cognição. Para isto, considera-se que cada módulo, ou aspecto da inteligência abordado, pode ser implementado por um ou mais SBC's.

Estes SBC's, entretanto, devem estar de acordo com uma ontologia de organização, onde definem-se os mecanismos sociais do sistema, como a ontologia apresentada por Coutinho et al. (2005). Esta ontologia, define uma organização por meio de conceitos como *papéis*, *grupos*, *planos* e *metas globais* e *normas* a serem seguidas. Pela definição destes mecanismos, os agentes derivam seu conhecimento segundo a coordenação e os objetivos do sistema.

Quanto sua arquitetura interna, um SBC pode ser implementado de duas formas. A primeira, considera um único SBC, com uma única base de fatos e um motor de inferência, mas dividido em múltiplas bases de regras, como descrito pela figura 4.2.

Este tipo de arquitetura é especialmente indicado quando há a necessidade de se refinar o conhecimento de um dos módulos cognitivos da figura 4.1. Um exemplo é o módulo de planejamento. Neste caso, uma abordagem seria a especificação de uma base de regras para o planejamento social e outra para o planejamento individual do agente. A primeira seria uma instância do planejamento de ações estabelecido para o sistema multiagente. A segunda corresponderia ao planejamento individual de ações do agente segundo o seu papel no plano social.

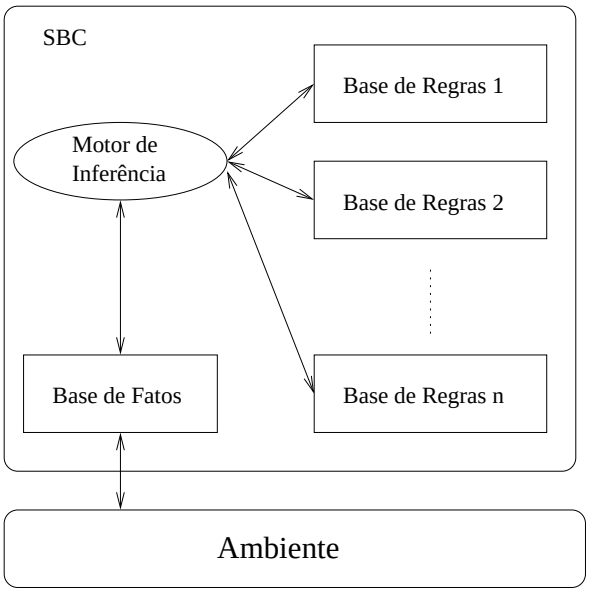


**Figura 4.2:** *Agente constituído por um SBC com múltiplas bases de regras*

A segunda forma de diminuir a complexidade do conhecimento de um sistema é dividi-lo em múltiplos sistemas baseados em conhecimento, como descrito pela figura 4.3. Cada módulo SBC na figura, pode ser formado segundo o modelo descrito na figura 4.2. A relação entre estes SBC's é obtida pelo envio de resultados específicos dos processos de inferência entre SBC's adjacentes.

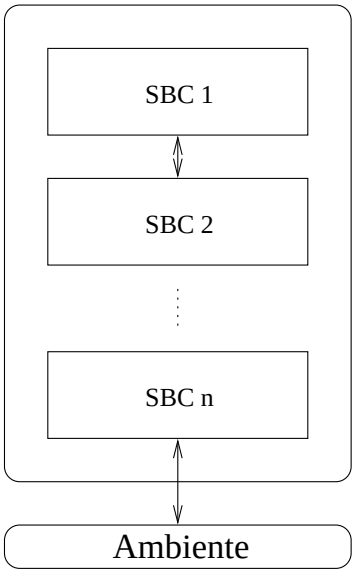


**Figura 4.3:** *Sistema constituído por múltiplos SBC'S*

Eventualmente, o último módulo SBC da figura 4.3 pode ser substituído por um sistema reativo, como uma rede neural, um controlador clássico ou nebuloso, ou um sistema de Brooks (Brooks, 1986)

para ações reativas.

De modo geral, este última arquitetura corresponde a uma arquitetura genérica cujo modelo da figura 4.1 é apenas uma instância. De qualquer forma, cada módulo da figura 4.1 pode ser constituído ou por um SBC como o a figura 4.2 ou uma arquitetura como a da figura 4.3. A escolha depende do nível de complexidade do conhecimento a respeito do domínio, dos tipos de métodos de resolução de problemas e dos formalismos de representação de conhecimento adotados.

## 4.3 Visões da Aquisição de Conhecimento Social e Individual

Em um SMA, há dois níveis de conhecimento que devem ser capturados: a especificação social e a especificação individual do conhecimento para execução das estratégias multiagente. A especificação social corresponde a organização do SMA de forma a estabelecer uma estratégia coordenada de ações no ambiente. A especificação individual corresponde ao conhecimento que cada agente deve possuir para fazer parte do SMA e atuar no sentido de seus objetivos.

Diferentemente dos modelos apresentados no item 3.5.1, a especificação social é baseada em uma abstração coletiva do conhecimento, sem considerar as ações individuais necessárias a realização das metas do sistema. Por outro lado, o agente instancia este conhecimento social de acordo com seu papel na equipe, elaborando uma estratégia pessoal de ações, para que ele possa atuar no ambiente.

Desta forma, o processo de aquisição de conhecimento, em uma *visão externa* do desenvolvimento, é constituído de dois níveis:

- *Nível de Especificação Social*: onde são definidas as estratégias de ação do sistema multiagente. É implementado pela aplicação de um modelo organizacional onde são definidos conceitos como *papéis*, *grupos*, *metas* e *planos* sociais. Dentre estes modelos estão o AGR (Ferber et al., 2004), uma extensão do modelo AALAADIN (Ferber e Gutknecht, 1998), o $\mathcal{M}$ oise$^+$ (Hübner et al., 2002), ou Esquemas Sociais (Lugo et al., 2001). A escolha do modelo depende de características específicas do modelo. O resultado final deste nível deve ser a geração de uma estratégia de coordenação, planejamento por exemplo, que deverá ser instanciada pelo nível individual.

- *Nível de Especificação Individual*: a partir do nível de especificação social e baseado no papel do agente no sistema e suas aptidões, especifica-se o conhecimento individual do agentes refinando-o segundo a estrutura cognitiva apresentada na seção anterior, até o nível de ação

reativa. É implementado pela aplicação das técnicas de aquisição de conhecimento em SBC's. Seu resultado final deve ser a geração das bases de conhecimento dos agentes que formam a sociedade.

A visão interna corresponde à aplicação dos conceitos de organização de SMA e aquisição de conhecimento, como aqueles definidos nos itens 2.2.4 e 2.4.3, para cada um dos níveis de abstração definidos pela visão externa do processo.

Sendo assim, cada módulo correspondente a um nível de abstração social ou individual é construído segundo as seguintes etapas:

- **Especificação**: o processo de especificação interna da base de conhecimento inclui a elicitação, a estruturação e organização do conhecimento. Divide-se em dois níveis de constituição:
  - **Nível Conceitual**: onde o conhecimento elicitado é organizado em unidades conceituais de conhecimento. Este nível pode ser compreendido como o nível de conhecimento de Newell (1982), ou seja, o conhecimento é organizado abstraindo os aspectos de implementação do SBC.
  - **Nível de Operação**: onde o conhecimento é organizado por meio de relações entre as unidades conceituais definidas no nível anterior. Esta organização pode ser obtida por uma linguagem de especificação, que no nosso caso, é baseada em um modelo particular de RP. A partir do nível conceitual, aspectos específicos de implementação são incorporados à linguagem, deixando engenheiro de conhecimento e especialista livres para tratar apenas do conteúdo da base. Assim, a responsabilidade de transformar o conteúdo elicitado e organizado em uma base de conhecimento adequada ao SBC é da linguagem de especificação.
- **Implementação**: na etapa de implementação, o conhecimento estruturado no nível de operação é transformado nas bases de conhecimento dos SBC que constituem os agentes. Neste ponto, os aspectos específicos devem ser respeitados, como o mecanismo de inferência e formalismos de representação de conhecimento utilizados.

Para compreender melhor o processo de aquisição de conhecimento proposto, supõe-se um problema de natureza multiagente em um ambiente real. Trata-se da construção de uma estratégia de jogo em uma equipe de futebol. O objetivo é identificar as etapas deste processo, do ponto de vista da organização e estruturação do conhecimento envolvido na estratégia, e determinar as etapas equivalentes em um processo de aquisição de conhecimento.

Inicialmente, o treinador define uma estratégia em uma especificação abstrata de execução. Uma estratégia possível consiste em explorar os benefícios das jogadas de linha de fundo, onde a bola deve preferencialmente ser levantada para o interior da área com o objetivo de facilitar a conclusão por um companheiro de equipe, enquanto, simultaneamente, dificulta a ação dos defensores. Para isso, o treinador define que as atividades coletivas da equipe devem convergir para ações nas laterais do campo. Uma vez alcançada a aproximação ideal da linha de fundo pela lateral do campo, a ação coletiva é concluída com um cruzamento para área.

Observe que esta primeira descrição é feita sem focar-se em ações individuais, mas apenas em nível de coordenação e organização coletiva. Trata-se de uma estratégia de ação social.

Dado o conhecimento individual e seu papel na equipe, cada jogador é capaz de identificar suas possibilidades de contribuição para a execução da estratégia definida pelo treinador. Ou seja, o jogador, autonomamente, sem que o treinador especifique o que cada um deve efetivamente fazer, define sua estratégia de ação pessoal. Já dentro do campo, em treinamento ou em uma partida, baseado na estratégia coletiva e na sua estratégia pessoal, o jogador realiza suas ações individuais em benefício da meta do time.

A figura 4.4 descreve como o modelo global do SMA é visualizado a partir da utilização das RP como linguagem de especificação. Cada nível de abstração corresponde a uma RP Hierárquica, segundo o modelo a ser apresentado na próxima seção.

## 4.4 Redes de Petri como Linguagem de Especificação em Engenharia de Conhecimento

A abordagem de especificação de conhecimento proposta nas seções anteriores completa-se com a apresentação da sua linguagem de especificação.

Basicamente, uma linguagem de especificação possui duas funções: serve como formalismo capaz de integrar os múltiplos níveis do sistema; e serve como um mapeamento semi-formal entre o conhecimento elicitado e as estruturas de representação de conhecimento[2].

Nesse sentido, propõe-se a utilização de um modelo particular de RP de Alto Nível como linguagem de especificação, cujo papel é intermediar o nível conceitual e o nível operacional do processo, e posteriormente, transformar o conhecimento elicitado e estruturado em uma base de conhecimento. Dentre os motivos que justificam a utilização de RP estão:

[2] c.f. item 2.4.3

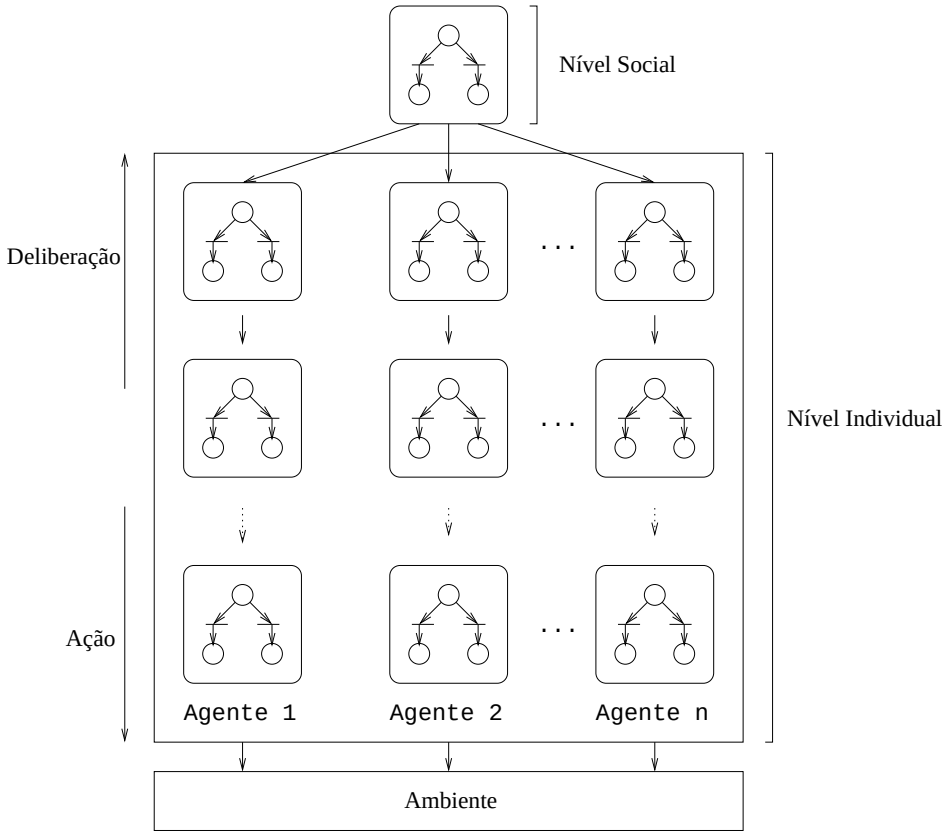


**Figura 4.4:** *Descrição geral da abordagem de especificação de conhecimento para SMA*

- seu modelo gráfico é utilizado como linguagem diagramática permitindo ao engenheiro de conhecimento compreender como o especialista "pensa" o sistema;
- seu modelo matemático pode ser usado no processo de verificação de anomalias, como inconsistências, ambiguidades e redundâncias;
- permitem uma composição hierárquica, respeitando a modularidade da abordagem de especificação de conhecimento subjacente, ou seja, sua adequação a uma composição em múltiplos níveis de abstração do sistema;
- é possível a tradução automática da informação contida na rede em uma base de conhecimento, por intermédio de uma adequada estruturação de dados, respeitando a sintaxe da base de regras e os formalismos de representação de conhecimento utilizados;

Para uma abordagem baseada em conhecimento usando RP, é necessário estender o poder de expressão das fichas. Assim, permite-se a representação de manipulações em bases de conhecimento ocorrida quando do disparo de uma regra. Para este propósito, modelos de alto nível são bem adequados, como por exemplo os modelo de Sibertin-Blan (1985) ou Jensen (1996). A idéia é integrar um mecanismo estruturado de representação de dados ao formalismo de uma RP.

A partir de uma perspectiva epistemológica, o componente central de uma abordagem baseada em conhecimento é a *base de conhecimento* (Russel e Norvig, 1995). Informalmente, uma base de conhecimento é formada por um conjunto de descrições de fatos. Uma base de conhecimento genérica, independente do formalismo de representação de conhecimento adotado, pode ser formalizada pela definição de duas funções de acesso, *Tell* e *Ask*, que permitem, respectivamente, incluir um novo fato e perguntar sobre um fato à base de conhecimento.

Mesmo no nível de especificação social, objetivo é obter uma descrição do conhecimento social do sistema, que deverá ser instanciado pelos agentes da sociedade. Desta forma, a RP, neste nível, deve ser visto como um SBC social, cuja implementação impõe a estratégia de coordenação entre os agentes.

### 4.4.1 Termos como Mecanismos Genéricos para Representação de Conhecimento

Formalmente, seja $KB$ o conjunto de todas as possíveis bases de conhecimento e $\phi$ uma expressão da linguagem formal usada pelo método de representação de conhecimento adotado. Sem perda de generalidade, diz-se que $\phi$ é um termo. Seja $\mathcal{V}$ o conjunto de símbolos de variáveis, $\mathcal{C}$

o conjunto de nomes de entidades primitivas no domínio e $\mathcal{F}$ o conjunto de nomes de funções. O conjunto de todos os termos $\mathcal{T}$ é definido como:

- $\mathcal{V} \subseteq \mathcal{T}$;
- $\mathcal{C} \subseteq \mathcal{T}$;
- se $t_1, t_2, \ldots, t_n \in \mathcal{T}$ e $f \in \mathcal{F}$, então $f(t_1, t_2, \ldots, t_n) \in \mathcal{T}$.

Seja $\mathcal{S}$ o conjunto de todos os mapeamentos possíveis $\mathcal{V} \rightarrow \mathcal{T}$, ou seja, o conjunto de todas as substituições de variáveis, e $\mathcal{T}^*$ o conjunto de todos os termos *fechados*, ou seja, termos em que não há a ocorrência de variáveis. Deste modo, é possível definir:

$$Tell : KB \times \mathcal{T}^* \rightarrow KB$$

$$Ask : KB \times \mathcal{T} \rightarrow \mathcal{S}$$

Desta forma, estabelece-se uma representação que permite a especificação do sistema a partir de seu nível de conhecimento. No nível individual esta representação implica um formalismo de representação de conhecimento, como aqueles apresentados no item 2.4.2. Já no nível social, esta representação deve englobar os conceitos inerentes a uma organização social, como aqueles definidos no item 2.2.4.

Inicialmente, demonstra-se a formalização de uma representação de conhecimento social, ou seja, o formalismo base para a estruturação de uma base de conhecimento social, segundo um modelo de organização. O conjunto de termos a seguir diz respeito a uma estratégia de cooperação baseada em Conhecimento Social Dinâmico (CSD) (da Costa e Bittencourt, 2002).

Segundo este modelo, uma meta global $g_i$ pode ser alcançada por um dos planos que constituem um conjunto de planos $P_i$. Cada plano $p \in P_i$, é constituído por um conjunto de tarefas $T$, e um conjunto de agentes $A$ que podem executá-las. Assim, seja o conjunto de termos $\mathcal{T}$ aptos a representação deste tipo de estrutura:

- $\mathcal{C}_1 = \{p, T, A\}$
- $\mathcal{C}_2 = \{p_o, p_1, p_2, t_1, t_2, a_1, a_2, a_3\}$
- $\mathcal{C} = \mathcal{C}_1 \cup \mathcal{C}_2$

- $\mathcal{V} = \{x_1, x_2, ..., x_n\}$ tal que $n \in \mathbb{N}$
- $\mathcal{F} = \{f, c\}$
- $f(c(p, x_i)\ c(T, x_{i+1})\ c(A, x_{i+2}))$

De acordo com os termos apresentados, a RP pode manipular três planos $(p_0, p_1, p_2)$, duas tarefas $(t_1, t_2)$, em três agentes distintos $(a_1, a_2, a_3)$. A lógica por traz desta representação e como manipulá-la faz parte da interpretação da rede.

O seguinte exemplo mostra a generalidade da representação por termos ao implementar uma representação do tipo $(objeto\ atributo\ valor)$, análoga a do MYCIN, utilizando termos.

Desta forma, suponha um conjunto de termos $\mathcal{T}$ tal que:

- $\mathcal{C}_1 = \{objeto, atributo, valor\}$
- $\mathcal{C}_2 = \{meta, bola, estado, atual, controle, defesa, ataque, ativa, sucesso, falha, nenhuma, minha-equipe\}$
- $\mathcal{C} = \mathcal{C}_1 \cup \mathcal{C}_2$
- $\mathcal{V} = \{x_1, x_2, x_3\}$
- $\mathcal{F} = \{f, c\}$
- $f(c(objeto, x_1)\ c(atributo, x_2)\ c(valor, x_3))$, tal que:

estabelecendo, assim, o método de representação de conhecimento. Por intermédio de $\mathcal{F}$, a intenção é restringir a abrangência dos termos cujos tipos são representados pelos elementos de $\mathcal{C}_1$ a subconjuntos disjuntos dos elementos representados em $\mathcal{C}_2$. Analogamente às linguagens de programação, trata-se de um processo de tipagem de variáveis, de forma a estabelecer uma representação do tipo $(objeto\ atributo\ valor)$.

Vejamos agora um exemplo destacando a utilização de quadros como formalismo de representação de conhecimento. No contexto deste exemplo, um quadro é formado pelo conjunto de objetos no campo de visão de um agente, considerando um domínio qualquer, obtido a partir das informações sensoriais geradas pelo ambiente. Estas informações sensoriais são geradas a partir de um modelo numérico do ambiente. Sendo assim, seja o conjunto de termos $\mathcal{T}$:

- $\mathcal{C}_1 = \{time, name, distance, direction\}$, que identificam o instante de tempo em que o quadro foi gerado, o nome de um objeto e sua direção e distância com relação ao agente.
- $\mathcal{C}_2 = \{$conjunto de objetos que pertencem ao modelo numérico do ambiente$\}$.
- $\mathcal{C} = \mathcal{C}_1 \cup \mathcal{C}_2$
- $\mathcal{V}_1 = \{x_1, x_2, \ldots, x_n\}$.
- $\mathcal{V}_2 = \{y, t_s\}$.
- $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2$
- $\mathcal{F} = \{f, c\}$
- $f(c(time, t_s), c(name, x1), c(y, x_i))$, tal que: $i = 2, \ldots, n$.

O conjunto $\mathcal{C}_1$ define os atributos de um objeto do quadro, que se referem a sua caracterização dentro do campo de visão do agente, onde *name* designa a identificação. Por intermédio de $\mathcal{F}$, o conjunto de identificadores de objetos é restrito ao conjunto de objetos que pertencem ao modelo numérico do ambiente. Além disso, os atributos de um objeto são variáveis, ou seja, um atributo pode pertencer ou não a um objeto em um dado instante de simulação. Por sua vez, um quadro é identificado pelo tempo de simulação $t_s$ em que foi gerado.

É importante salientar, que a utilização de termos permite definir apenas a sintaxe de uma representação de conhecimento. Entretanto, para que seja garantida sua correção semântica, é necessário que o engenheiro de conhecimento defina a interpretação dos mecanismos da rede segundo a correta adequação semântica da representação ao ambiente, segundo as condições definida no item 2.4.2.

### 4.4.2 Utilização de Termos na Construção de uma Redes de Petri para a Construção de Bases de Conhecimento

**Pré-requisitos para a construção da RP**

Para gerar a RP, é necessário que se estabeleça um protocolo de comunicação entre o engenheiro de conhecimento e o especialista, associado à linguagem de especificação da abordagem. Trata-se de um mecanismo, semi-formal ou formal dependendo da complexidade do domínio, que permite apresentar o conhecimento em unidades conceituais e as relações entre as mesmas. Este

protocolo dá ao engenheiro de conhecimento meios para que ele possa vir a estruturar e organizar o conhecimento elicitado em um conjunto de RP hierárquicas. Tal protocolo deve ser definido a partir de características próprias ao domínio. Este processo é descrito pela figura 4.5

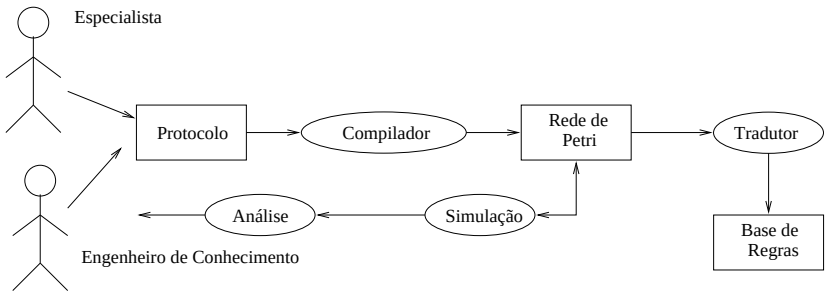


**Figura 4.5:** *Construção das RP*

Engenheiro de Conhecimento e especialista interagem com o objetivo de levantar os conceitos e as relações entre estes conceitos. De acordo com a figura 4.5, este processo é representado pelo retângulo etiquetado com a denominação **Protocolo**. Entre os tipos de relações existentes entre conceitos do domínio, podem ser utilizadas relações de precedência, complementaridade, entre outras. Todavia, tal protocolo tem relação direta com aspectos específicos do domínio, e deve ser definido a partir deste pressuposto básico.

O resultado deste processo passa por uma etapa de compilação - bloco **Compilador** - em que a linguagem utilizada neste protocolo é então mapeada em uma representação em RP, conforme o modelo apresentado acima. A rede resultante é então simulada e analisada - bloco **Análise** - e o resultado serve de realimentação para que o engenheiro de conhecimento, em conjunto com o especialista, possa modificar o sistema, se necessário. Uma vez que o resultado da análise das redes seja considerado satisfatório, seu conteúdo é traduzido - bloco **Tradutor** - no conjunto de bases de conhecimento do agente - bloco **Base de Regras**. A distribuição destas bases é definida de acordo com a arquitetura do agente.

Um exemplo desta abordagem pode ser observado em Cardoso et al. (2004). O problema aqui é definir um protocolo para que um professor possa estruturar o modelo pedagógico de um Sistema Tutor Inteligente. Este modelo pedagógico controla as interações entre os aprendizes e os agentes do sistema. Cada agente é responsável por um sub-domínio do conhecimento abordado. Neste caso, o professor organiza o conteúdo de um domínio em um protocolo baseado na ordenação hierárquica de currículos. Cada currículo, por sua vez, é constituído por Unidades Pedagógicas (UP), e cada UP é constituída por problemas que devem ser resolvidos para que o aluno possa prosseguir no processo de aprendizagem. Este conteúdo é organizado por intermédio de uma interface baseada em uma

linguagem gráfica, cujo grafo subseqüentemente será transformado em RP. O conteúdo desta RP é então traduzido na base de conhecimento do modelo pedagógico do sistema.

Observa-se neste modelo que o especialista, neste caso o professor, utiliza uma linguagem diagramática para estruturar as unidades conceituais do domínio - currículos, unidades pedagógicas e problemas. Portanto, para definir este protocolo foi necessário basear-se em aspectos específicos do domínio.

É importante ressaltar que o engenheiro de conhecimento e o especialista não utilizam RP como um "vocabulário" ou "linguagem" comum para que possam conversar a respeito do domínio. Quando se diz que RP permitem a comunicação entre estes agentes do processo, por tratar-se da linguagem de especificação adotada, esta comunicação dá-se em nível de interpretação do conhecimento. Isto significa dizer que as RP permitem a avaliação do conhecimento por intermédio de suas ferramentas de análise estrutural. Após o processo de verificação, o engenheiro de conhecimento tem meios de passar ao especialista os erros encontrados na processo de estruturação do conhecimento previamente elicitado, especialmente quanto a relações de dependência entre as unidades conceituais utilizadas.

**Elementos da RP**

Na sua formulação mais básica, uma regra é constituída por premissas e conclusões. No nível mais inferior de abstração do sistema, uma regra pode ainda concluir por ações no ambiente, contextualizadas a partir de suas condições e efeitos. Para introduzir estas premissas e conclusões, condições e efeitos, extende-se a definição de RP[3] da seguinte forma.

Uma ficha é definida como um elemento do conjunto *KB*, ou seja, a ficha agora representa a base de conhecimento. Assim, a função *Ask* recebe um termo em $\mathcal{T}$ e uma base de conhecimento $k \in KB$ e retorna uma substituição $\theta \in \mathcal{S}$. Em seu nível de abstração mais inferior, sua forma geral é:

$$\begin{aligned}
&\theta_1 \leftarrow Ask(k, \phi_1)\\
&\vdots\\
&\theta_n \leftarrow Ask(k, \phi_n)\\
&\theta \leftarrow Combine(\theta_1, \ldots, \theta_n)
\end{aligned}$$

onde $\phi_i \in \mathcal{T}$ são termos dependentes do domínio, possivelmente contendo variáveis, que são usadas para interrogar a base de conhecimento *k* e *Combine* é uma função que combina substituições.

[3] c.f. item 3.3.3

Por sua vez, a hierarquização entre redes é introduzida quando for necessária a troca de elementos de $\mathcal{T}$ entre os níveis de abstração do sistema. Neste caso, a função *Ask* deve ser substituída por expressões como:

$$\theta_i \leftarrow Run(R, \phi)$$

onde $R$ é uma RP de nível inferior e $\phi \in \mathcal{T}$ é um termo dependente do domínio, possivelmente contendo variáveis, que é usado para interrogar a base de conhecimento de nível de abstração inferior, após a execução de RP $R$. Assim como *Ask*, esta função retorna uma substituição $\theta \in \mathcal{S}$.

Finalmente, a transição é disparada pela aplicação do procedimento:

$$\begin{array}{l} \text{if } Cond(\theta) \text{ then} \\ Tell(k, \psi\theta) \wedge Action(\alpha, \theta) \end{array}$$

onde $Cond : \mathcal{S} \rightarrow \{V, F\}$ é uma condição sobre os valores das variáveis em $\theta$ e $Action(\alpha, \theta)$ uma especificação de ação a ser gerada no ambiente.

Caso o resultado da função *Cond* seja verdadeiro ($V$), a função *Tell* modifica a base de conhecimento $k$ de acordo com o que é especificado pelo termo fechado $\psi\theta$. Opcionalmente uma ação $\alpha$ é executada no ambiente, possivelmente levando em conta os valores das variáveis em $\theta$. Para definir a função *Cond* faz-se necessário a definição de uma linguagem que permita expressar condições sobre um conjunto de variáveis simbólicas e/ou numéricas.

### 4.4.3 Mapeamento das Redes de Petri em Regras de uma Base de Conhecimento

De forma genérica, uma RP mapeia uma regra da seguinte forma. Seja a RP da figura 4.6.

Cada transição da RP corresponde a uma regra em uma base de regras. O campo *Ask* corresponde a premissa da regra. O campo *Tell* corresponde à conclusão da regra, enquanto o campo *Action*, que é opcional, corresponde a alguma ação que deve ser gerada no ambiente. As premissas e as conclusões devem estar de acordo com o formalismo de representação de conhecimento definidos pelo conjunto de termos da rede. Dadas estas condições, a regra gerada a partir da figura 4.6 é:

```
SE <premissa> ENTÃO <conclusão> AÇÃO <ação>
```

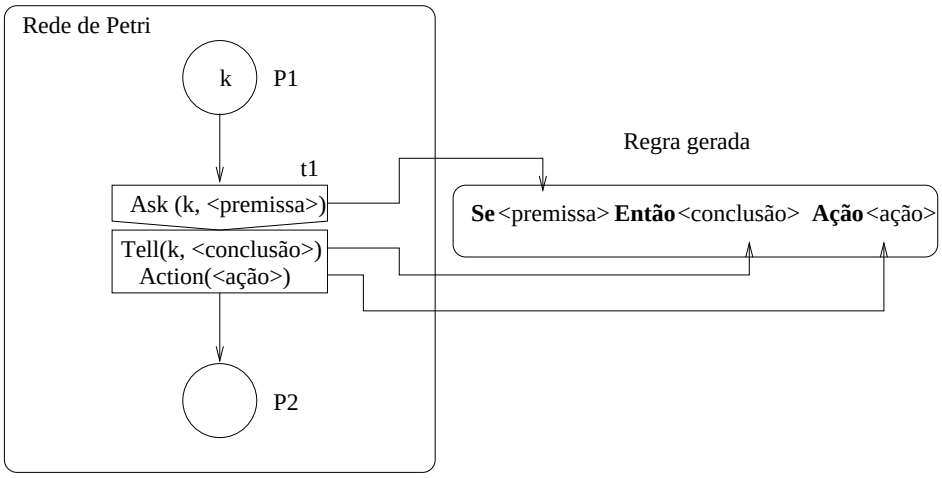


**Figura 4.6:** *RP genérica*

Foram suprimidos, por questões de simplificação notacional, a expressão do retorno das substituições e a aplicação da função *Cond*.

Seja, por exemplo, a RP da figura 4.7. Nela se utiliza o conjunto de termos para a representação de conhecimento segundo o formalismo lógico $(objeto\ atributo\ valor)$. Trata-se do nível de planejamento de um agente, segundo a especificação definida na seção anterior, para as estratégias de uma equipe de futebol.

A marcação inicial da rede indica que há uma ficha no lugar $P_1$, representando o estado inicial da base de conhecimento.

A transição $t_1$ tem como pré-condição a função *Ask*, cujos parâmetros são *k* e os padrões lógicos $(meta\ atual\ nenhuma)$ e $(meta\ estado\ nenhuma)$. Isto significa dizer que a base de conhecimento *k* é indagada sobre a presença dos referidos padrões lógicos. Uma pré-condição é representada por um pentágono no modelo gráfico da transição. Como não há a utilização de variáveis, a função retorna um valor booleano *true* se o padrão lógico é verificado, e *false* caso contrário. Seja a seguinte leitura de *k*:

```
(meta atual nenhuma)
(meta estado nenhuma)
```

Desta forma, a condição da transição $t_1$ é atendida.

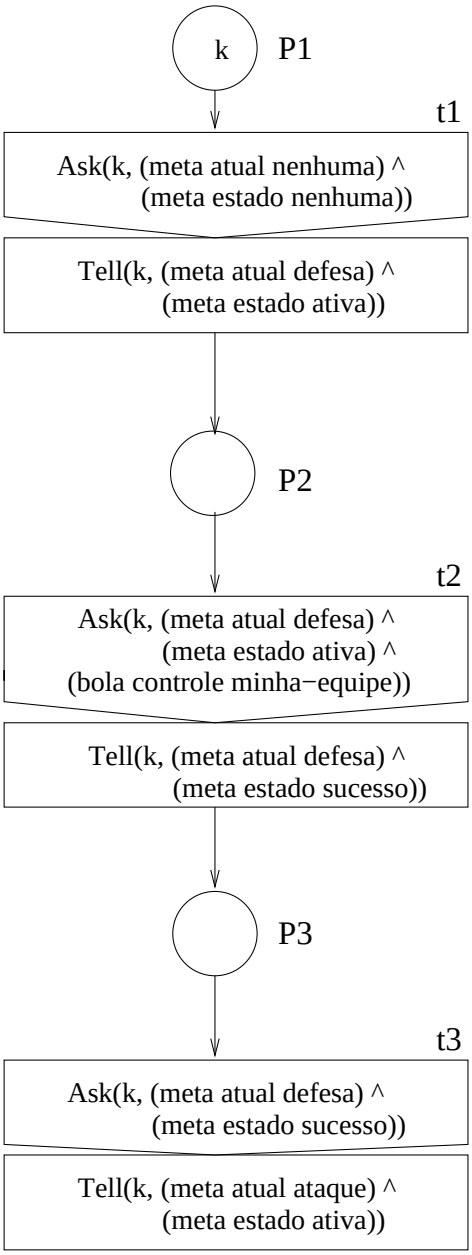


**Figura 4.7:** *RP da base*

Sendo assim, uma ação incide sobre a ficha *k*, modificando-a segundo a função *Tell*. A ação incidente sobre *k* é representada no modelo gráfico da transição por um retângulo. Neste caso, a ficha *k*, representando a base de conhecimento do sistema, é modificada segundo os parâmetros da função. Como resultado, a base de conhecimento *k* passa a conter os seguintes padrões lógicos:

```
(meta atual defesa)
(meta estado ativa)
```

O mesmo processo se repete até que *k* alcance a transição-fim $t_3$, descrevendo seu estado final.

Observa-se que a rede descreve a dinâmica da base de conhecimento do sistema, onde um estado é representado pela distribuição de fichas, ou marcação da rede. Dada a abordagem de especificação de conhecimento, o objetivo é alcançar as metas do sistema. Estas metas podem ser descritas por um subconjunto específico de marcações da rede. Verificando que este é também um subconjunto do conjunto de marcações acessíveis da rede, diz-se que o sistema possui o conhecimento necessário para operar no ambiente. Este processo de verificação é viabilizado por algum dos métodos descritos na seção 3.5, que dentre outras possibilidades, está também a identificação de anomalias na base.

Garantida a correção da RP, esta gera o seguinte conjunto de regras, a partir da especificação da figura 4.7:

```
REGRA 1
  SE      (meta atual nenhuma)
          (meta estado nenhuma)
  ENTAO   (meta atual defesa)
          (meta estado ativa)
REGRA 2
  SE      (meta atual defesa)
          (meta estado ativa)
          (bola controle minha-equipe)
  ENTAO   (meta atual defesa)
          (meta estado sucesso)
```

```
REGRA 3
  SE      (meta atual defesa)
          (meta estado sucesso)
  ENTAO   (meta atual ataque)
          (meta estado ativa)
```

A seguir, a RP da figura 4.7 será modificada com o objetivo de descrever a presença de variáveis na premissa das regras. Sendo assim, seja a figura 4.8, mais especificamente na transição $t_2$. Nela, há a presença da variável $x_1$, reconhecida como tal devido à presença do caracter '?' antecedendo-a. Logo a seguir a definição da função *Ask*, especifica-se a substituição desejada para a verificação da pré-condição.

A dinâmica da rede é exatamente a mesma do exemplo anterior, exceto quanto à operação de substituição. Uma vez que há variáveis no modelo, uma lista de substituições $\theta$ é gerada, contendo as possíveis instâncias para $x_1$. Neste caso, o nível de interpretação da rede restringe as substituições válidas que devem ser aplicadas a função $Tell$. No caso de uma delas ser igual aquela especificada na rede, a pré-condição é verificada.

Nesta rede, a premissa associada à transição $t_2$ apresenta a variável $x_1 \in \mathcal{V}$, cuja substituição válida é $\theta = (x1 = minha - equipe)$. A verificação desta substituição garante o disparo de $t_2$. Assim a REGRA 2 passa a ser declarada da seguinte forma:

```
REGRA 2
SE    (meta atual defesa)
      (meta estado ativa)
      (bola controle ?x1)
FILTRO(?x1 = minha-equipe)
ENTAO (meta atual defesa)
      (meta estado sucesso)
```

Uma vez definido como utilizar o modelo dentro de um único nível de concepção, o próximo passo é a definição dos mecanismos de hierarquização, que permitem a especificação e a relação de múltiplos níveis de abstração do problema. Para a implementação deste procedimento, deve ser evocado a função *Run* da RP.

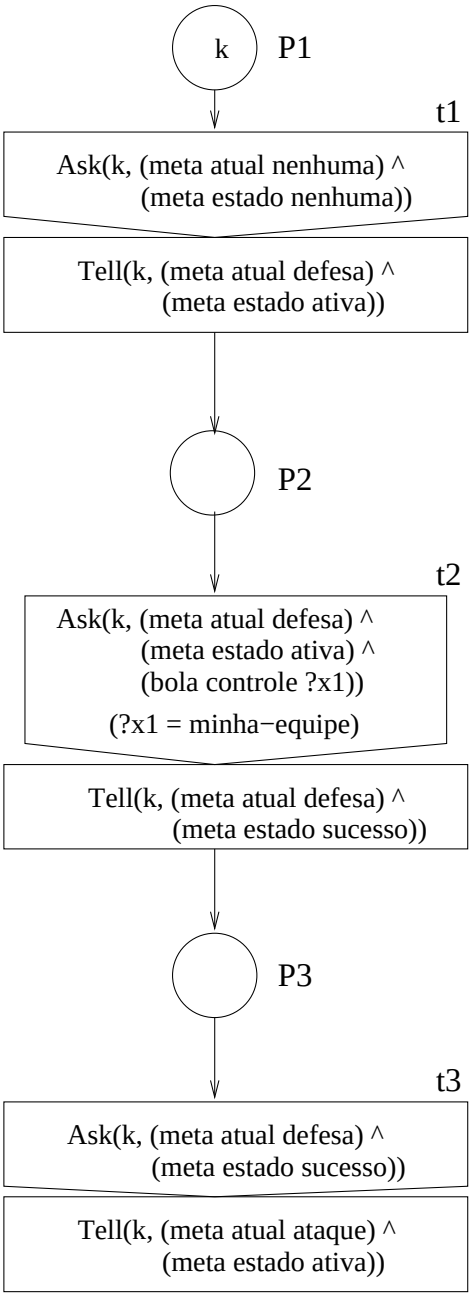


**Figura 4.8:** *RP modelando uma base de conhecimento com a presença de variáveis*

Este mecanismo é retratado no exemplo genérico da figura 4.9. Trata-se da mesma rede da figura 4.6, entretanto, contendo um nível de abstração gerado a partir de uma substituição de transição aplicada a $P_2$ na RP 1.

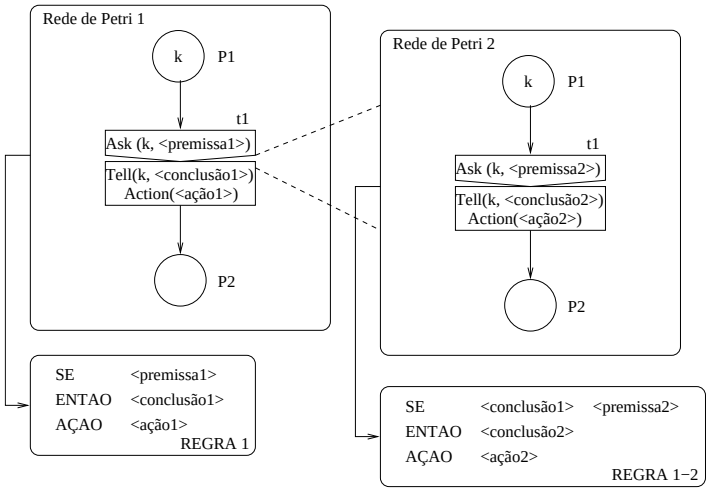


**Figura 4.9:** *RP Hierárquica genérica*

O mapeamento para a geração de regras é o mesmo para o caso anterior, entretanto com uma restrição quanto às regras geradas a partir do novo nível hierárquico. Assim, a regra gerada a partir da RP 1, designada por $REGRA\ 1$ é a mesma regra gerada no exemplo genérico anterior. Já a segunda regra, designada como $REGRA\ 1-2$ deve conter como premissa a leitura da base de fatos *k* que a gerou, além da premissa especificada em cada uma de suas transições. Neste caso a base de regras gerada a partir da figura 4.9 é:

```
REGRA 1
SE <premissa1> ENTAO <conclusão1> AÇAO <ação1>
REGRA 1-2
SE <conclusão1> <premissa2> ENTAO <conclusão2> AÇAO <ação2>
```

Com a inclusão do mecanismo de hierarquização, a codificação da base de regras deve respeitar a arquitetura do agente subjacente. Esta adequação é definida pelo nível de interpretação da rede, que não deve ser confundido com o nível de interpretação do sistema. Este nível de interpretação é capaz de descrever a interação entre os dados do conjunto de redes, a arquitetura subjacente e o ambiente externo. O nível de interpretação da rede pode, inclusive, ser especificado por uma outra RP (Cardoso e Valette, 1997, cap. 4). Contudo, esta possibilidade está fora do escopo desta tese, permanecendo em

aberto para futuras extensões deste modelo[4]. Fundamentalmente, o que deve ser compreendido, é que a estrutura da rede é incapaz de definir um mapeamento direto e formal com a arquitetura do sistema, ficando a cargo do engenheiro de conhecimento a correta adequação das regras às respectivas bases de conhecimento. Porém, a linguagem proposta está apta a representar qualquer tipo de arquitetura de forma genérica.

Seja por exemplo a rede da figura 4.10. Considere que nela especifica-se o nível de planejamento de uma base de conhecimento de um agente ou SBC. Este planejamento é definido pelas relações de precedência entre as metas do sistema. Entretanto, o modelo nada apresenta com relação aos processos e ações que levam a realização destas metas. Para isso, faz-se necessário a inclusão de novos níveis de abstração ao modelo, tantos forem necessários, até que se alcance o nível de especificação das ações no ambiente.

Sendo assim, considera-se inicialmente a marcação $M = [0\ k\ 0]$ na RP da figura 4.7. Neste estado, a base de fatos é:

```
(meta atual defesa)
(meta estado ativa)
```

Do ponto de vista da RP, a evolução é dependente da ocorrência do evento $(bola\ controle\ minha-equipe)$. Do ponto de vista do sistema, este evento implica a tomada de bola pela equipe do agente. Por isso, este termo é indicado pela diretiva *Run*.

Dentro do exemplo proposto, a chamada ao procedimento *Run* faz com que uma rede hierarquica de nível inferior seja instanciada. Assim, todo o conjunto de regras originadas na rede de nível inferior instanciada a partir de uma chamada *Run*, deve conter em suas premissas, a leitura da base de fatos deste lugar.

Considere que o lugar $P_2$ da rede da figura 4.10 explode na RP da figura 4.11. Esta nova rede, herda a estrutura de dados do nível hierárquico superior. Entretanto, do ponto de vista do sistema, tratam-se de duas bases de conhecimento independentes. Como restrição, este conjunto de regras deve ter também como premissa, a marcação que lhe deu origem, além da pré-condição das transições da rede.

Neste caso, o exemplo apresentado apenas utiliza o processo de hierarquização de redes com o objetivo de particionar a especificação da base em torno dos aspectos do conhecimento necessários

[4] c.f. item 6.1

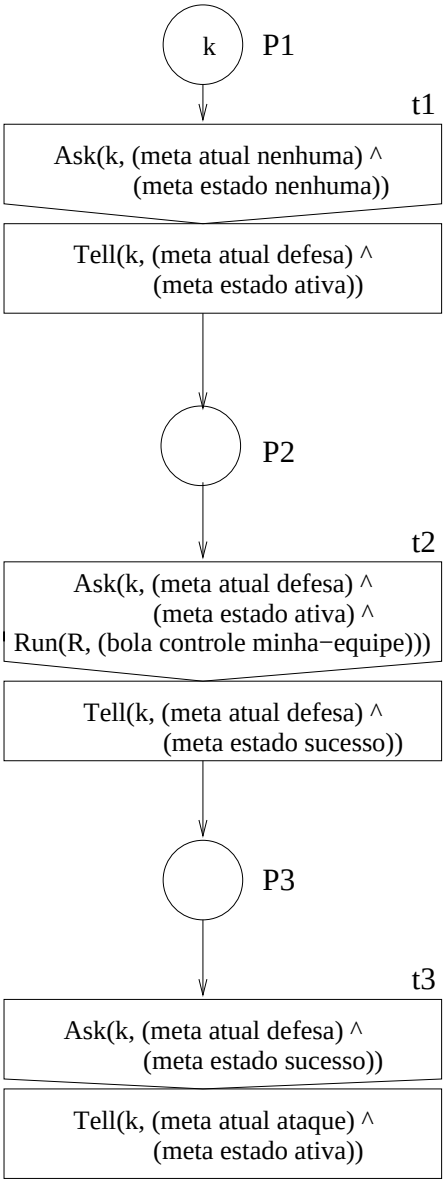


**Figura 4.10:** *RP com mecanismo de hierarquização*

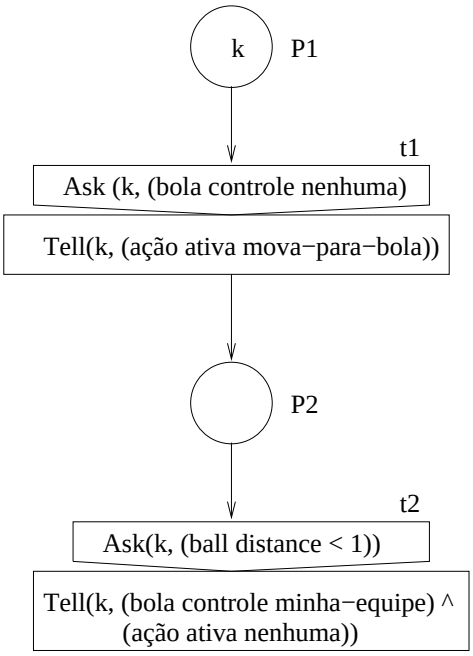


**Figura 4.11:** *RP de nível inferior*

para a operação no ambiente. Contudo, no nível de implementação é gerada uma única base de conhecimento. Neste caso a base gerada a partir do processo de integração das RP das figuras 4.7 e 4.11 é a seguinte:

```
REGRA 1
SE    (meta atual nenhuma)
      (meta estado nenhuma)
ENTAO (meta atual defesa)
      (meta estado ativa)
REGRA 1-1
SE    (meta atual defesa)
      (meta estado ativa)
      (bola controle nenhuma)
ENTAO (ação ativa vai-para-bola)
```

```
REGRA 1-2
SE    (meta atual defesa)
      (meta estado ativa)
      (ball distance < 1)
ENTAO (bola controle minha-equipe)
      (ação ativa nenhuma)
REGRA 2
SE    (meta atual defesa)
      (meta estado ativa)
      (bola controle minha-equipe)
ENTAO (meta atual defesa)
      (meta estado sucesso)
REGRA 3
SE    (meta atual defesa)
      (meta estado sucesso)
ENTAO (meta atual ataque)
      (meta estado ativa)
```

No momento em que a rede da figura 4.7 obtém a marcação $M = [0\ k\ 0]$, a rede da figura 4.11 é instanciada. Uma vez que a rede de nível hierárquico inferior alcança a transição-fim $t_2$ o fluxo de controle é devolvido à rede superior.

Quanto à geração de regras, este novo nível acrescenta as regras REGRA 1-1 e REGRA 1-2 da base acima. Observe que estas regras apresentam como parte da premissa, a marcação que lhe deu origem.

Segundo a abordagem de especificação de conhecimento das seções anteriores, a rede da figura 4.7 descreve o nível de planejamento do sistema, enquanto a rede da figura 4.11 descreve o nível de classificação, onde as metas são descritas em conjuntos de ações no ambiente. A relação entre estes níveis é estabelecida pelo envio da meta do sistema ao módulo relativo a classificação de ações. Da mesma forma, o nível de planejamento se mantém monitorando a validade das metas por intermédio das decisões e inferências do nível inferior.

Quanto à aplicação da abordagem na especificação do nível social, pressupõe-se que este será formalizado segundo um SBC social. Dito de outra forma, este nível de especificação corresponde ao

conhecimento que os agentes devem possuir a respeito da estratégia de coordenação entre os membros da sociedade, para que se possa convergir de maneira otimizada para os objetivos do sistema. Este tipo de conhecimento não está centralizado em alguma entidade do sistema, mas sim, difundido entre seus membros.

Esta abordagem é possível a partir da formalização de uma representação de conhecimento social, onde a RP especifica as relações dos conceitos de natureza social, como papéis, grupos, tarefas, planos e metas globais. Isto pode ser obtido pela aplicação de algum modelo de organização SMA, como uma estratégia de cooperação (da Costa e Bittencourt, 2002), o modelo $\mathcal{M}$ oise$^+$, a Rede Contratual (Smith, 1980), ou Esquemas Sociais (Lugo et al., 2001). O resultado final desta especificação deve ser um descrição do conhecimento social segundo um mecanismo de coordenação[5], por exemplo, planejamento.

Sejam, por exemplo, o conjunto de termos apresentados no item 4.4.1 para uma representação de conhecimento social segundo a estratégia de cooperação Conhecimento Social Dinâmico (da Costa e Bittencourt, 2002). A RP resultante da especificação social deve estabelecer a coordenação entre os três planos possíveis, ou seja, $(p_0, p_1, p_2)$. Além disso, as funções *Ask* e *Tell* podem manipular os termos que descrevem os agentes envolvidos $(a_1, a_2, a_3)$, e as tarefas possíveis, $(t_1, t_2)$. Um exemplo da aplicação destes termos, pode ser visualizado na figura 4.12.

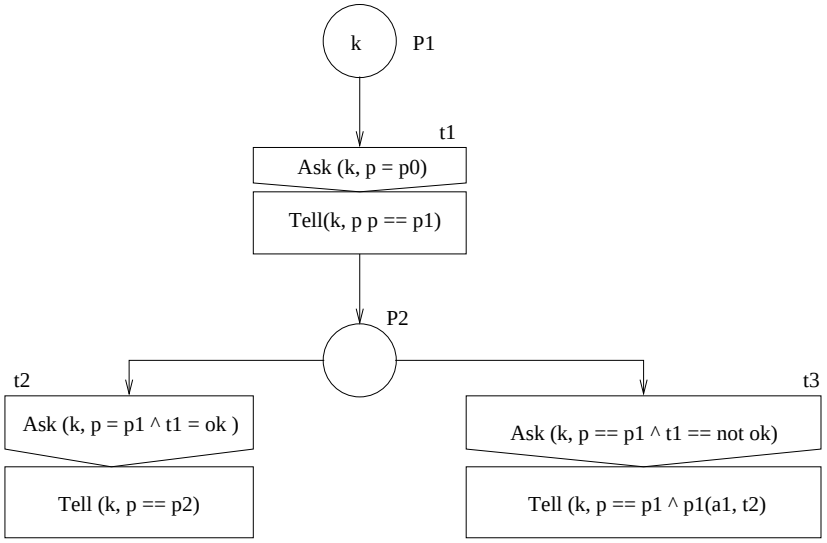


**Figura 4.12:** *Especificação social usando CSD*

Segundo a especificação social da figura 4.12, o sistema possui dois planos possíveis de ações, ou do ponto de vista da estrutura da rede, dois invariantes de lugar. Uma plano é formado pela

[5] c.f. item 2.2.4

seqüência $s_1 = [t_1\ t_2]$, e outro pela seqüência $s_2 = [t_1\ t_3]$.

A primeiro plano global para pela execução de $p_1$, que é executado pela conclusão de $t_1$, até alcançar a transição $t_2$, onde o sistema fica indefinidadamente preso na execução de $p_2$. Entretanto, se $t_1$ não for finalizado com sucesso, $p_1$ deve ser executado apenas pelo agente $a_1$ por intermédio da tarefa $t_2$.

Uma vez finalizada a especificação social, ela deve ser instanciada na base de conhecimento de cada agente membro da sociedade. A correta instanciação do conhecimento social garante a coordenação do SMA.

## 4.5 Discussão

O objetivo desta seção é discutir a proposta a partir do referencial teórico estabelecido pela análise dos métodos correlacionados na literatura. Esse referencial é divido em três aspectos principais: metodologias para desenvolvimento de SMA, aquisição de conhecimento e aplicações de RP como linguagem de especificação no processo de aquisição. Portanto, esta análise parte das sínteses comparativas apresentadas nos items 2.3, 2.5 e 3.5.

### 4.5.1 Desenvolvimento de SMA

A síntese comparativa do item 2.3 destaca cinco aspectos passíveis de contribuição para uma evolução das metodologias de desenvolvimento de SMA. Estes aspectos são explicitamente abordados na abordagem proposta.

Ao permitir que conceitos inerentes à organização social sejam formalizados pelo conjunto de termos da RP, o formalismo aborda, de forma natural, o paradigma de orientação à agentes, sem a necessidade de adaptações de formalismos da engenharia de software. Segundo a síntese do item 2.3, o que se tem, via de regra, é a aplicação da orientação a objeto ao desenvolvimento de SMA. Entretanto, o conceito de objeto é insuficiente para se obter propriedades como autonomia e interação entre os membros da sociedade, que compõe a solução do sistema.

Esta situação é evidenciada, por exemplo, em uma comparação com AUML. Trata-se de uma metodologia que busca extender o poder de especificação da UML, incluindo conceitos específicos da orientação a agentes. Além disso, o AUML considera um único modelo de organização do SMA.

A RP, por outro lado, apresenta-se como um formalismo melhor adaptado a representar questões como paralelismo de atividades e modelagem de protocolos.

Além de a abordagem proposta incluir os conceitos inerentes de forma natural, ela dá a liberdade ao desenvolvedor de utilizar qualquer modelo de organização do sistema, bastando formalizá-lo pelo conjunto de termos da RP social.

Um segundo aspecto tratado pela abordagem é a simplificação da especificação das múltiplas interações entre os agentes da sociedade pela definição da estratégia de coordenação. A partir desta estratégia, estabelece-se o conhecimento compartilhado entre os agentes, definido como conhecimento social, e a partir dele, os mecanismos de interação necessários a implementação do SMA. Entres estes mecanismos estão o protocolo e a linguagem de comunicação. De fato, esta estratégia de coordenação, representada por uma RP, é o resultado final do processo de especificação do nível social do SMA.

Em comparação à metodologia MaSE, por exemplo, a definição da coordenação simplifica e unificaria as etapas de construção de diagramas de seqüências, na fase de análise, e da implementação das Conversações entre classes de agentes, na fase de projeto.

O terceiro aspecto de contribuição da abordagem proposta é a forma como aborda-se a especificação do nível individual, ou seja, a construção dos agentes do SMA. Segundo as metodologias analisadas, existem duas formas para esta abordagem: ou generaliza-se a arquitetura dos agentes, e o resultado final é o apenas o conjunto de requisitos para a implementação, ou restringe-se a uma arquitetura específica, normalmente BDI, e o resultado final são agentes construídos segundo este modelo.

A abordagem proposta define uma arquitetura genérica, baseada em conhecimento, de modo que o resultado final desta fase seja a especificação dos agentes em nível de conhecimento. Deste modo, o que se tem, é a implementação da base de conhecimento destes agentes. A finalização do sistema dá-se pela implementação dos mecanismos de inferência adaptados aos requisitos do problema. A vantagem desta abordagem é a independência de aspectos relativos a implementação do sistema, porém em um estado mais avançado do que apenas requisitos para a construção de agentes. Além disso, tem-se uma alternativa a arquitetura BDI, nem sempre a melhor solução para um problema distribuído.

Um aspecto que também diferencia a abordagem proposta daquelas analisadas, é a utilização de um mesmo formalismo durante todas as fases de desenvolvimento do sistema. A RP estabelece a integração entre todos os níveis de abstração do sistema, pelo mecanismo de hierarquização. O que se tem, em todas as metodologias analisadas, é a utilização de diferentes ferramentas, sem qualquer tipo

de relação formal entre elas, nas mais diversas fases de desenvolvimento. Assim, com a RP, é possível visualizar o sistema desde seu mais alto conceito social, até a implementação da ação correspondente no ambiente.

Finalmente, a abordagem também diferencia-se daquelas apresentadas na medida em que é possível desenvolver o sistema nos dois sentidos, ou seja, dos agentes para o social, como do social para os agentes. No primeiro caso, tem-se um conjunto de especificações que relacionam as habilidades dos agentes que deverão ser organizados segundo um modelo social definido pela RP. Por outro lado, é possível também especificar o sistema a partir de seus requisitos sociais, e a partir daí, especificar e implementar os agentes que comporão a sociedade multiagente.

### 4.5.2 Engenharia do Conhecimento

No sentido de estabelecer uma relação da abordagem para especificação de conhecimento proposta neste capítulo com as metodologias correlatas descritas no item 2.5, deve-se considerar o processo de produção de um SBC como um todo. Esta abordagem se justifica na medida em que a aquisição de conhecimento só faz sentido dentro da respectiva ferramenta para o desenvolvimento de um SBC. Dependendo da ferramenta, é muitas vezes impossível delimitar o que corresponde à aquisição e o que corresponde à implementação dos mecanismos que dão suporte ao sistema.

Quanto à *abordagem* empregada no processo, entende-se que a abordagem proposta é baseada em *modelos*, assim como o KADS/CommonKADS e o Protégé. Esta designação se justifica uma vez que ela utiliza um modelo de cognição baseado em aspectos da inteligência com o objetivo de modularizar o processo de aquisição de conhecimento. Estes módulos de conhecimento podem ser constituídos por sub-bases de conhecimento, ou mesmo por um novo sistema baseado em conhecimento completo [6]. Uma alternativa é, ainda, a constituição de uma única base de conhecimento cujas regras são agrupadas de acordo com o aspecto da inteligência tratado. O KADS/CommonKADS utiliza a estruturação em torno de modelos com o objetivo de generalizar tanto o conhecimento elicitado como a arquitetura subjacente. Já o Protégé, que tem seu desenvolvimento centrado em ontologias, aborda esta estrutura sob três classes distintas: ontologias de domínio, ontologias de método e ontologias de aplicação. Desta forma, o engenheiro de conhecimento, por intermédio destas classes, engloba os principais conceitos do sistema, que devem ser instanciados pelo especialista. Por outro lado, o EMYCIN/TEIRESIAS e o ONCOCIN/OPAL têm suas abordagens de desenvolvimento baseadas em *transferência* de conhecimento. A abordagem proposta também mantém uma relação de similaridade com o primeiro uma vez que permite uma organização de regras em grupos. No caso

[6] c.f. item 4.2

do EMYCIN/TEIRESIAS, este agrupamento dá-se em torno de regras com parâmetros similares, enquanto na abordagem proposta este agrupamento dá-se em torno de regras que abordam aspectos similares do conhecimento.

O critério *relação com o domínio* estabelece o grau de dependência entre a metodologia e/ou abordagem e o domínio de aplicação. Nesse aspecto, entende-se que a proposta pode ser vista como uma abordagem de desenvolvimento *independente do domínio*. Entretanto, isto não implica dizer que ela pode ser utilizada em qualquer domínio. Desta forma, ela apresenta uma certa similaridade com o EMYCIN/TEIRESIAS. Apesar de ambas serem consideradas ferramentas independentes do domínio de aplicação, os mecanismos disponíveis para a implementação do sistema limitam as possibilidades a um subconjunto específico de domínios. Neste caso, os parâmetros que definem este conjunto são os mecanismos de inferência e os formalismos de representação de conhecimento disponíveis. Entende-se que assim, propô-se uma abordagem mais geral, uma vez que se tem um maior conjunto de opções para estes parâmetros se comparado com o EMYCIN/TEIRESIAS. Por outro lado, seu subconjunto de domínios é menor do que o oferecido pelo KADS/CommonKADS e Protégé. Estas metodologias, dada a maior flexibilidade e complexidade de suas ferramentas, permitem a aplicação a um maior número de domínios.

No que se refere a representação de conhecimento, a abordagem proposta apresenta-se como uma alternativa tão poderosa quanto aquelas mais gerais, como KADS/CommonKADS e Protégé. Não obstante, sua linguagem de especificação permite a estruturação de qualquer formalismo, independente dos formalismos disponíveis pela arquitetura subjacente. O KADS/CommonKADS, limita a sua linguagem de especificação formal à lógica de predicados de primeira ordem, contudo, assistida por uma linguagem de especificação informal, a Linguagem de Modelos Conceituais (CML). Já o Prótégé utiliza ontologias, e conseqüentemente, todo o seu poder de *reutilização*. O EMYCIN/TEIRESIAS e o ONCOCIN/OPAL restringem-se a uma representação em regras de produção.

As metodologias apresentadas dividem-se, também, entre aquelas que operam diretamente no *nível simbólico* , ou no *nível de conhecimento* do sistema, abstraindo aspectos de implementação (Newell, 1982). O EMYCIN/TEIRESIAS e ONCOCIN/OPAL estruturam o conhecimento elicitado diretamente em regras de produção. Desta forma, o especialista participa do processo de construção do sistema apenas na etapa de elicitação de conhecimento e de avaliação do sistema, sendo esta última, já na fase de protótipo. Desta forma, muitos erros de natureza estrutural do conhecimento, incluídos devido a não familiarização do engenheiro de conhecimento com o domínio, só podem ser detectados apenas na etapa final. O ONCOCIN/OPAL permite a participação direta do especialista uma vez que seu processo de aquisição de conhecimento é totalmente dependente do domínio. A concepção do

processo de aquisição no nível de conhecimento, por intermédio de uma linguagem de especificação mais intuitiva, permite que problemas deste tipos sejam minimizados, incluindo a participação do especialista nas etapas intermediárias do processo de construção do sistema. A abordagem proposta permite uma concepção em nível de conhecimento uma vez que se estabeleça as representações de conhecimento a serem utilizadas pelas fichas da RP de Alto Nível. Já os mecanismos oferecidos pelo KADS/CommonKADS não são tão intuitivos, na medida em que o engenheiro de conhecimento deve prover ferramentas específicas para cada tipo de domínio. Diz-se assim, que EMYCIN/TEIRESIAS e ONCOCIN/OPAL têm sua *concepção* viabilizada no *nível simbólico* do sistema, por intermédio de *linguagens formais*, enquanto a abordagem proposta, assim como o KADS/CommonKADS e o Protégé, tem sua concepção no *nível de conhecimento*, viabilizada por intermédio de *linguagens semi-formais*. Estas linguagens semi-formais, por sua vez, estruturam o conhecimento segundo a linguagem formal adotada no sistema.

De maneira geral, conclui-se que a abordagem proposta preenche uma lacuna entre as metodologias analisadas. É possível afirmar que a proposta resolve o problema dos arcabouços baseados em uma abordagem por transferência, mas sem a complexidade associada às metodologias baseadas em modelos. Isto é obtido aumentando a flexibilidade das abordagens por transferência, mas sem o excessivo número de funcionalidades das abordagens por modelos. Desta forma, abrange-se problemas que requerem soluções descentralizadas, baseadas em um conhecimento altamente especializado, mas para um número intermediário de agentes no sistema.

### 4.5.3 Redes de Petri

No contexto de utilização de RP como linguagem de especificação está a possibilidade de representação de múltiplos níveis de abstração do sistema. A partir deste contexto, a literatura que trata da aplicação de RP em Engenharia do Conhecimento pode ser compreendida sob dois aspectos: o da especificação e modelagem *social*, e especificação e modelagem *individual*.

As diretrizes fundamentais para a utilização de RP em nível social, de acordo com a análise apresentada na seção 3.5, estão sintetizadas em Holvoet (1995). O autor denomina *sincronização multi-modelo* a hierarquização entre os múltiplos níveis do sistema, do nível de organização e coordenação social à especificação do conjunto de ações no ambiente. Segundo este modelo genérico, o modelo global deve ser descrito pelos componentes individuais do sistema, ou seja, os agentes. O sistema passa a ser compreendido como social por intermédio dos mecanismos de sincronização entre os agentes. Segundo a síntese apresentada no item 3.5.1, estes mecanismos podem ser ações sincronizadas ou planos. Este modelo é descrito pela figura 4.13.

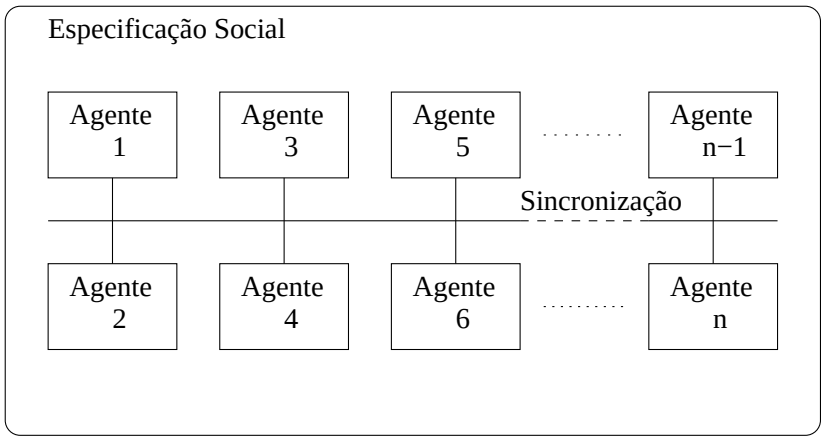


**Figura 4.13:** *Modelo social segundo Holvoet (1995)*

Segundo o modelo, a especificação social é compreendida pelo somatório do conhecimento dos *n* agentes do sistema. Desta forma, uma representação global torna-se impraticável, considerando um sistema com muitos agentes. A visualização global é possível apenas pelo conhecimento dos mecanismos de sincronização entre os agentes.

A utilização de RP como linguagem de especificação, dentro da abordagem para especificação de conhecimento proposta, prevê uma especificação social por intermédio de planos sociais, descritos segundo as metas do sistema. Esta RP representa uma base de conhecimento social, sem nível de implementação, utilizada pelos agentes no sentido de instanciar este conhecimento segundo seus papéis e aptidões no ambiente. Este modelo é apresentado pela figura 4.14.

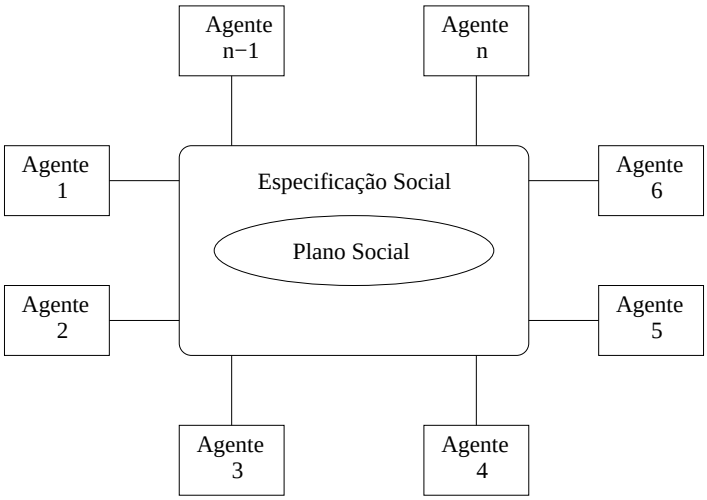


**Figura 4.14:** *Modelo social segundo a abordagem proposta*

A vantagem deste modelo é a flexibilidade. Neste caso, os agentes definem suas ações sem

a necessidade de pontos específicos de sincronização. Uma vez acordada a meta do sistema, cada agente consulta sua instância da base social e delibera seu planejamento individual de ações.

No nível de implementação, o conhecimento do sistema é descentralizado, formado pela soma do conhecimento de seus agentes. Entretanto, no nível de especificação, o sistema possui uma descrição abstrata de seus planos e metas, que, por sua vez, possui uma especificação particular para cada agente.

De um outro lado está a utilização de RP na especificação, modelagem e verificação de sistemas inteligentes em nível individual. Este indivíduo pode ser visto como o mecanismo de controle de um SBC ou mesmo um agente, considerado em suas mais variadas formas de arquitetura. Do item 3.5.2, conclui-se que, neste tipo de abordagem, a RP é utilizada de três diferentes formas:

- como método de representação de conhecimento, sobretudo relacionado a lógica proposicional, de primeira ordem e nebulosa;
- como método de verificação e validação de bases de conhecimento, por intermédio de uma análise estrutural destas, buscando minimizar a ocorrência de anomalias, que podem ser fontes de futuros erros do sistema;
- incorporado a metodologias e/ou abordagens de desenvolvimento, permitindo uma análise de especificação, buscando uma certa correção semântica a partir dos requisitos do sistema;

Dentro da abordagem de especificação de conhecimento, pelo menos no escopo desta tese, busca-se a integração destes aspectos segundo um processo de abstração que adota um modelo particular de RP como linguagem de especificação. Deste modo, mais do que um formalismo de representação de conhecimento, mecanismo de verificação estrutural ou semântica, a RP executa o papel de uma meta-representação de conhecimento, buscando estabelecer uma espécie de linguagem intermediária entre o engenheiro de conhecimento e o especialista. Para o especialista, a rede funciona como uma linguagem capaz de representar a forma como o engenheiro entende o conhecimento elicitado permitindo sua avaliação em etapas intermediárias do processo. Para o engenheiro de conhecimento, a rede permite a modularização do processo de desenvolvimento, avançando gradualmente em direção ao nível de implementação do sistema. Neste caso, os modelos já existentes para verificação estrutural de regras podem ser aplicados.

Não obstante, o modelo está de acordo com as restrições de uso de RP em sistemas de regras, estabelecidas em Cardoso e Pradin-Chézalviel (1997). Contudo esta relação não é direta na medida

em que o modelo proposto aqui visa uma representação genérica dos formalismos de conhecimento existentes. Em Cardoso e Pradin-Chézalviel (1997) o conjunto de restrições está estabelecido em torno da utilização dos diversos tipos de lógicas que podem ser representados por RP. Mesmo nesse caso, o modelo proposto está adequado às restrições, na medida em que estas se resumem ao atendimento das chamadas *verdades eternas* e a impossibilidade de representação de cláusulas disjuntivas ou conjuntivas como conclusões de regras em lógica clássica. Para isso, basta que a estruturação das diversas representações de conhecimento, e a estrutura lógica da base de conhecimento, considerem a ocorrência deste tipo de situação.

Entretanto, o diferencial com relação aos métodos já existentes é justamente a integração dos mecanismos de especificação social e individual em um único modelo. Desta forma, o sistema é concebido a partir da especificação social, onde são estabelecidas as metas globais do sistema, no sentido das ações individuais do conjunto de agentes da sociedade multiagente. Não obstante, o processo dá-se de maneira gradual e estruturada, estabelecidas as relações entre os múltiplos níveis de conhecimento necessários a construção do sistema.

Entende-se que as principais vantagens deste modelo são:

- a utilização de uma única abordagem de especificação entre os mais diversos níveis de decisão do sistema;
- possibilidade de aproveitamento do suporte teórico já desenvolvido em torno da utilização das RP em nível de especificação individual;
- possibilidade de estruturação de diversos formalismos de representação de conhecimento, como lógica, quadros e redes semânticas;
- modelo para especificação social que permite maior flexibilização e exploração do potencial de atuação dos agentes;
- transformação automática do conhecimento estruturado em RP em bases de conhecimento, independente da arquitetura adjacente.

O modelo, contudo, apresenta algumas limitações. A principal delas é a ausência de mecanismos que busquem uma verificação dos requisitos do sistema, garantindo um certo nível de correção semântica. Outra dificuldade está na ausência de um formalismo que estabeleça uma relação direta entre os diversos níveis hierárquicos da RP e arquitetura subjacente dos agentes. Esta relação, pelo menos por enquanto, é definida no nível de interpretação do sistema, estabelecida empiricamente pelo

engenheiro de conhecimento e/ou projetista. Finalmente, o mecanismo de reutilização de conhecimento fica restrito a possibilidades de difícil ocorrência, como no caso da reprodução fidedigna de níveis de especificação entre agentes.

## 4.6 Considerações Finais

O objetivo central deste capítulo é a apresentação da abordagem para especificação de conhecimento para SMA, ou seja, a descrição da solução para o problema tratado por esta tese. Tal abordagem é baseada em um modelo de cognição genérico, derivado de um modelo das funções executivas do cérebro. Tal modelo distribui a atividade cognitiva em diferentes aspectos da inteligência. A abordagem é desenvolvida por intermédio de uma linguagem de especificação baseada em um modelo particular de RP. A relevância desta abordagem é discutida em meio às abordagens correlatas existentes na literatura.

A abordagem tem o mérito tem abranger todas as etapas de desenvolvimento de um SMA, desde sua especificação social até a implementação dos agentes no ambiente em que atuarão. Isto é possível pela aplicação de uma série de restrições ás arquiteturas do sistema social e dos agentes.

O sistema social deve ser especificado a partir dos elementos de sua organização, ou seja, metas, papéis, planos globais, de tal modo que possam ser estruturados segundo uma representação de conhecimento social. A partir desta representação é possível determinar um esquema de coordenação social.

Os agentes devem ser estruturados segundo uma arquitetura cognitiva que permita uma especificação e implementação segundo uma abordagem por conhecimento. A arquitetura utilizada estrutura o conhecimento em aspectos da inteligência, relacionados hierarquicamente. Os aspectos mais superiores tratam da inserção dos agentes no contexto social, enquanto os aspectos mais inferiores tratam da atuação direta do agente no ambiente.

A especificação da relação hierárquica entre os níveis do sistema é viabilizada pelo modelo de Rede de Petri proposto. Além de ser utilizado como uma ferramenta para a verificação do conteúdo especificado, o modelo dispõe de uma linguagem de alto-nível que permite a estruturação de qualquer formalismo de representação de conhecimento, seja social ou individual. Esta estruturação dá-se pela determinação do conjunto de termos da rede segundo os formalismos a serem representados. A partir daí, os elementos da rede podem manipular estas estruturas, segundo as regras de funcionamento de um SBC.

Uma vez finalizada a modelagem e verificação das Redes de Petri, é possível a transformação do seu conteúdo na base de conhecimento dos agentes que constituem o sistema, por intemédio de um mapeamento entre elementos da estrutura da rede e da base.

Entretanto, a abordagem apresenta algumas limitações. A reutilização de conhecimento não é uma característica natural ao modelo. O compartilhamento de RP depende de uma série de compatibilidade entre estruturas de dados dos diferentes sistemas projetados. Além desta, o modelo não apresenta mecanismos formais que estabeleçam a relação entre as redes de nivel inferior, representando explicitamente o compartilhamento de bases de conhecimento quando isto acontece. Finalmente, a abordagem apresentada não cobre aspectos relativos a etapa de elicitação de conhecimento.

O objetivo final desta abordagem, situado além dos limites deste trabalho, é permitir a construção automática das bases de conhecimento de cada agente, a partir da especificação social do sistema e das funções e ações de cada agente. Instanciando o modelo social, cada agente é capaz de inferir o conhecimento necessário para que possa atuar no sentido das metas da sociedade multiagente, de acordo com suas potencialidades. A abordagem proposta é uma tentativa inicial de aproximação deste projeto, que inclui dentre outros problemas, a necessidade de uma camada que estabeleça interação entre especificação do conhecimento e as arquiteturas subjacentes.

# Capítulo 5

# Implementação e Resultados

## 5.1 Introdução

Após apresentar as bases teóricas deste trabalho, este capítulo descreve uma aplicação da abordagem em um domínio multiagente, o Soccerserver.

O Soccerserver é um simulador de futebol para agentes computacionais. Foi desenvolvido no âmbito da proposta mundialmente conhecida como Robocup, uma inciativa de um grupo de pesquisadores de IA com o objetivo de definir um problema padrão a ser resolvido: a implementação de robôs que joguem futebol. Nesse sentido, o Soccerserver abrange os problemas relacionados a implementação de softwares relativos a esta proposta.

Como arquitetura subjacente, é utilizado o agente autônomo concorrente (da Costa e Bittencourt, 1999b), uma implementação computacional do agente proposto por Bittencourt (1997). Para implementar este sistema, é utilizada a Expert-Coop++, um arcabouço para o desenvolvimento de Sistemas Multiagentes (SMA) Cognitivos sob restrições de tempo real do tipo melhor esforço.

Para os propósitos desta tese, o objetivo é implementar uma jogada entre três agentes dentro do simulador. Desta forma caracteriza-se uma abordagem multiagente, dando suporte à aplicação da proposta apresentada no capítulo anterior.

Dado este cenário, o capítulo é organizado da seguinte forma. A seção 5.2 descreve, de forma sucinta, o que é a Robocup e o simulador Soccerserver. Para maiores detalhes sobre o simulador deve ser consultado o apêndice A. A seguir, a seção 5.3 descreve a arquitetura de agente utilizada para o desenvolvimento da equipe. A seção 5.4 apresenta a Expert-Coop++ e sua sintaxe, além de aspectos

relativos à implementação do agente. A seção 5.5 descreve o processo de construção das bases de conhecimento do agentes para a implementação de uma jogada pré-planejada. Trata-se, na prática, da aplicação da abordagem descrita na seção anterior. Finalmente, a seção 5.6 apresenta as conclusões do capítulo.

## 5.2 Robocup e o Simulador Soccerserver

A Robocup emerge em meio a necessidade de domínios que especifiquem restrições impostas por problemas do mundo real cada vez mais distribuídos. A história da IA pode ser compreendida a partir da visão do agente em direção a sua inserção em um contexto social, onde suas potencialidades são expandidas na medida em que esta entidade individual absorve o conceito de cooperação, beneficiando-se da capacidade coletiva de soluções. Surge daí o conceito de Inteligência Artificial Distribuída, que engloba os SMA a as Soluções Distribuídas de Problemas (Durfee e Rosenschein, 1994).

### 5.2.1 Robocup

A Robocup surge em meio a um novo viés dentro da IA. Esta nova tendência aponta para a resolução de problemas por meio de soluções distribuídas (Durfee e Rosenschein, 1994) e pela integração das diferentes áreas nas quais a IA se fragmentou durante sua história (Alonso, 2002).

Desta forma, Kitano et al. (1995) oficializa a *Robot World Cup Initiative*, ou simplesmente, Robocup, como "uma tentativa de encorajar pesquisas em Inteligência Artificial e robótica inteligente provendo um problema padrão onde uma grande gama de tecnologias podem ser integradas e examinadas", especialmente aquelas de caráter distribuído. A idéia é estabelecer o futebol como domínio unificado no desenvolvimento de protótipos em nível de hardware e software. Permite-se, assim, que novas tecnologias desenvolvidas possam ser comparadas e validadas. Por outro lado gera-se um mecanismo de avaliação para o estado da arte em IA e robótica móvel, abrangendo temas como princípios de agentes autônomos, colaboração multiagente, *aquisição de estratégia*, raciocínio em tempo real e fusão de sensores.

Como projeto de pesquisa padrão, a Robocup, além de prover um domínio unificado, estabelece um conjunto de metas de curto, médio e longo prazo. Estas metas servem como diretrizes para as pesquisas relacionadas ao tema, fundamentando mecanismos de avaliação padrão. Estas devem

culminar com a montagem de um equipe de robôs humanóides em condições de vencer a seleção campeã da Copa do Mundo, em meados do atual século (Kitano et al., 1997).

De forma a abranger diferentes aspectos do desafio, a Robocup dividiu as competições em algumas ligas:

- Liga de Robôs de Pequeno Porte
- Liga de Robôs de Médio Porte
- Liga de Robôs com Pernas
- Liga do Simulador

Além destas, hoje já existe a Liga de Robôs Humanóides, atestando o avanço das pesquisas em direção à meta final, que é a construção de uma equipe em condições de vencer a seleção campeã do mundo, por volta de 2050. Existe ainda uma Liga de Robôs para Resgate, cujo propósito é a construção de robôs para resgates em desastres, como queda de prédios e barreiras, onde a presença humana pode não ser indicada devido a iminência de algum perigo (Asada, 2002).

### 5.2.2 Soccerserver

Dentre as diversas ligas da Robocup, a Liga do Simulador abrange assuntos relativos ao desenvolvimento de SMA, e as extensões que pode adquirir. Como exemplo, tem-se o aprendizado e síntese de ações em ambientes multiagente, e especificamente no caso deste trabalho, *aquisição de conhecimento em SMA*, dentre outros temas. Isto significa que a partir de uma abordagem de desenvolvimento distribuída, os vários módulos e componentes de um agente devem ser especificados no sentido da sua imersão em um ambiente social, adequando suas potencialidades, expressas em seu conjunto de metas locais, às necessidades do ambiente, expressas pelas metas do sistema.

O simulador, denominado *Soccerserver* [1] (Chen et al., 2001) é um sistema que implementa uma partida de futebol entre agentes computacionais escritos em qualquer linguagem de programação. Uma partida é executada sob o paradigma cliente/servidor: um servidor implementa um campo de futebol virtual, como o da figura 5.1, além de prover os métodos que simulam os movimentos dos

[1] Todo material apresentado sobre o Soccer Server refere-se a sua versão 7.07, utilizada nos experimentos deste trabalho. Atualmente o simulador encontra-se em versões mais avançadas, trabalhando inclusive em planos gráficos tridimensionais. No entanto, no sentido de viabilizar a conclusão deste trabalho, foi mantida a utilização da versão 7.07, o que de forma alguma inviabiliza a extensão dos resultados aqui obtidos para versões mais atualizadas, o que serve inclusive como uma perspectiva futura de implementação.

jogadores e da bola. Os clientes são os jogadores que interagem com o servidor por meio de sockets UDP/IP (Noda e Matsubara, 1996).

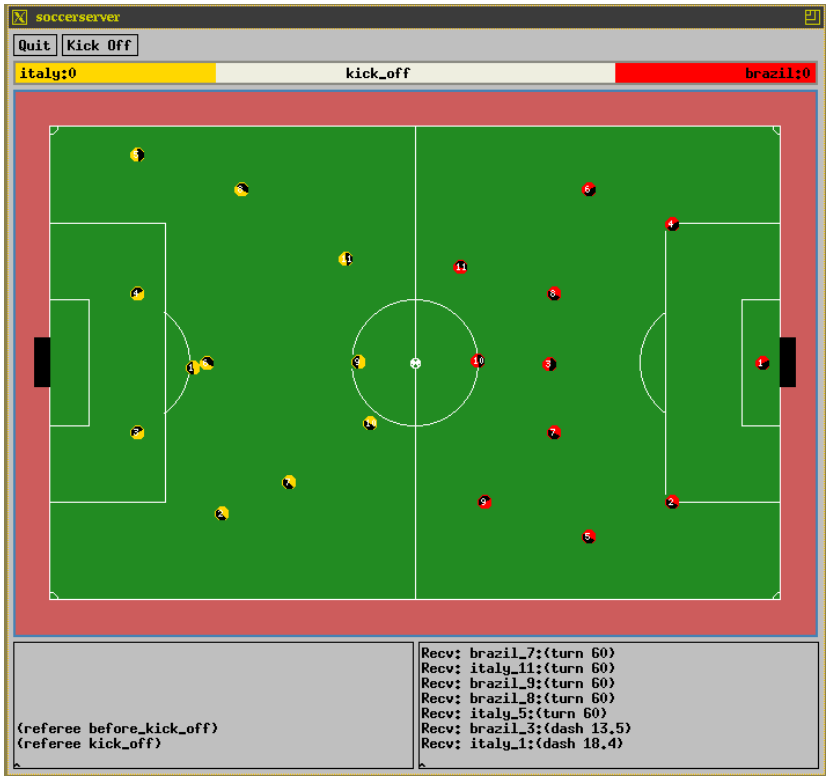


**Figura 5.1:** *Tela do monitor do Soccer Server*

O Soccer Server é constituído por dois programas executados em paralelo: o `soccerserver`[2] e o `soccermonitor`. O `soccerserver` é um programa servidor que simula os movimentos da bola e dos jogadores, comunica-se com os clientes, e controla o jogo de acordo com as regras. O `soccermonitor` é o programa encarregado de gerar um campo virtual na tela do computador a partir do modelo numérico do `soccerserver`. Além de prover uma interface com o servidor, o *soccermonitor* apresenta informações relativas ao placar do jogo, nome das equipes, e as posições de cada jogador e da bola. Uma descrição gráfica dos módulos que formam o Soccer Server pode ser vista na figura 5.2.

A comunicação entre cliente e o `soccerserver` é feita por uma conexão UDP/IP. Para o servidor, um cliente é visto como um processo em separado atribuindo-lhe uma porta específica. Pelo socket, o cliente recebe a informação visual e auditiva e envia os comandos até o servidor. Uma restrição feita é quanto a comunicação entre os clientes que deve ser feita por intermédio do si-

[2]Não confundir com o simulador. De fato, os responsáveis pelo projeto poderiam optar por nomes diferentes para evitar confusão.

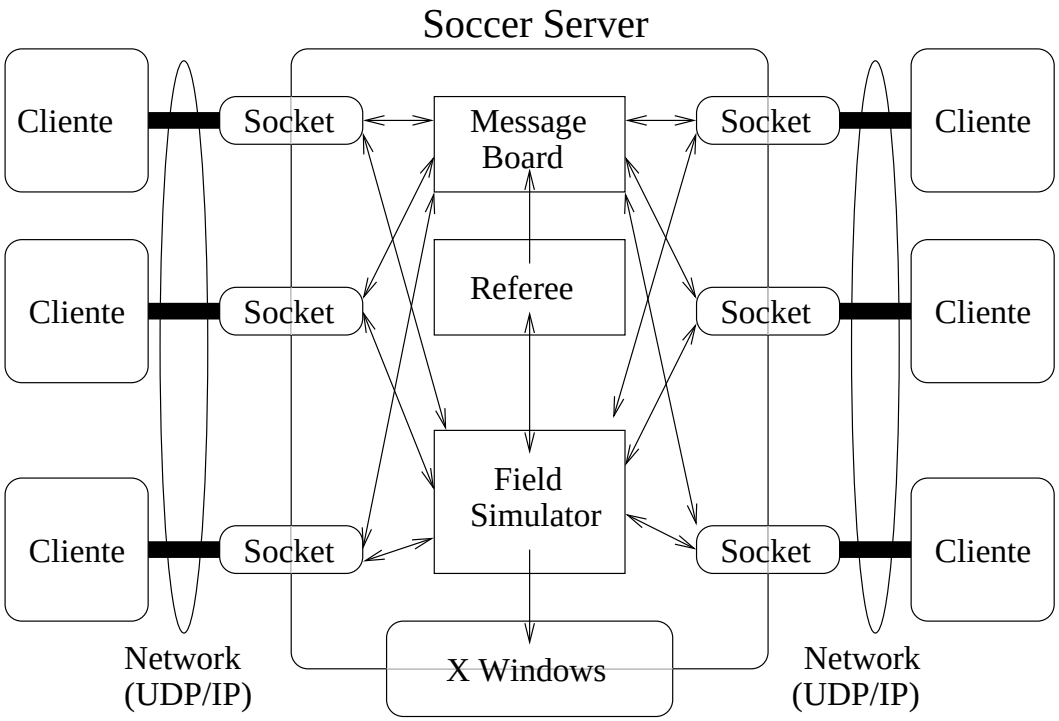


**Figura 5.2:** *Descrição do Soccer Server*

mulador e não diretamente entre eles. A escolha do protocolo UDP/IP, bem como a exigência de comunicação pelo simulador, deve ser entendida como mais um conjunto de restrições do ambiente, representando um meio não confiável e de banda limitada. No entanto, os projetistas são livres para escolher a linguagem de programação de sua preferência.

O Soccer Server, por meio de um módulo interno denominado *Árbrito*, provê a arbitragem da partida permitindo, todavia, a intervenção de um árbitro humano no caso de uma infração não prevista no código que o implementa. Questões específicas sobre a atuação deste módulo têm pouca relevância para esta tese. Na necessidade de informações mais apuradas, sugere-se a leitura do manual (Chen et al., 2001). No entanto, um aspecto importante quanto a este módulo, é a possibilidade de operá-lo no modo *Treinador* (do inglês "Coach"). Este modo é previsto, essencialmente, para sessões de treinamento da equipe ou treinamento individual. De fato, todas as simulações neste trabalho foram implementadas neste modo, indicado para a simulação de cenários simplificados.

O apêndice A descreve detalhadamente a interface do Soccerserver com os agentes/clientes.

## 5.3 O Agente Autônomo Concorrente

Um ambiente complexo, como o Soccer Server exige uma solução composta por componentes de baixo nível, como comportamentos reativos aptos às demandas de tempo real, bem como por componentes de alto nível, integrando aspectos deliberativos, como planejamento e metas, além de uma inserção em uma comunidade de soluções, de acordo com as especificações apresentadas na seção anterior. Neste sentido propõe-se um modelo de agente cognitivo baseado em três hipóteses fundamentais: dependência histórica, imersão em um ambiente evolutivo e auto-organização interna de seus componentes (Bittencourt, 1997). De acordo com este modelo, um agente apresenta três níveis de decisão: reativo, instintivo e cognitivo. Este modelo é instanciado em uma arquitetura computacional que dá suporte à idéia do agente autônomo concorrente como solução proposta para o Soccer Server. Esta arquitetura foi também escolhida uma vez que pode ser compreendida com uma instância da concepção genérica de aquisição de conhecimento descrita pela figura 4.1.

Para implementar uma equipe de agentes para o Soccer Server, a arquitetura cognitiva proposta em Bittencourt (1997) foi simplificada, devido a razões de eficiência de desempenho e também devido a restrições específicas do simulador. Do nível reativo faz-se uso apenas de controladores nebulosos, uma vez que o Soccer Server já provê informações perceptivas em um formato simbólico, descartando a necessidade de processamento de imagens. O nível instintivo é implementado por um sistema baseado em conhecimento (SBC) de ciclo único, onde o leitura do estado do jogo executa o papel de memória do agente. Já o nível cognitivo corresponde a um SBC simbólico, responsável pelas questões de planejamento individual e social, além de implementar a comunicação do agente.

Nesta abordagem, os parâmetros codificados geneticamente são aqueles que definem os conjuntos nebulosos dos controladores em uso (Gonçalves, 2001). Por outro lado, uma estratégia utilizando *aprendizado por reforço* (Sutton e Barto, 1998) é utilizada com o objetivo de aprender os limiares a respeito das condições do ambiente, inferidas no nível instintivo (Martins, 2003). Por se tratarem de componentes locais, estes parâmetros podem ser obtidos por meio de simulações *off-line*, onde alguns poucos agentes são inicializados no simulador.

O agente implementado apresenta os três níveis decisórios do modelo genérico, de acordo com a descrição da figura 5.3. Cada nível é associado a um processo implementado usando uma abordagem de programação multi-threads (Sun Microsystems, 1990). Assim, cada processo é dividido em duas threads concorrentes, onde a primeira é responsável pela manipulação da interrupção Unix SIGIO, que informa a existência de uma nova mensagem no socket do processo (mailbox). A outra thread executa as atividades do processo. Esta implementação evita o desperdício de tempo por

parte da thread principal verificando a existência de novas mensagens. A comunicação interprocessos é feita por sockets no domínio Unix, enquanto a comunicação com o simulador é feita por socket UDP/IP. Esta implementação foi escrita em linguagem de programação C++ e integra um ambiente de desenvolvimento para SMA cognitivos sob restrições de tempo real denominada *Expert-Coop++* (da Costa et al., 2003)[3].

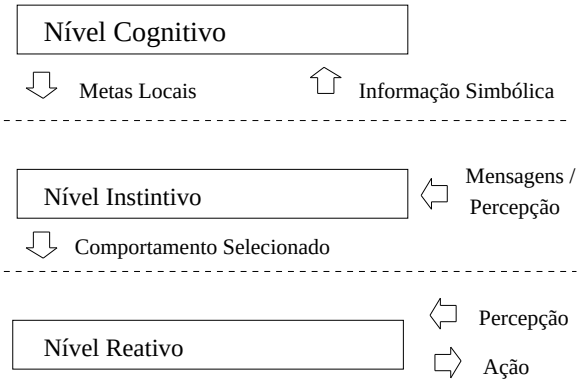


**Figura 5.3:** *Arquitetura do Agente para o Soccer Server*

### Nível Reativo

O nível reativo tem como função básica interagir com o ambiente, recebendo as informações de percepção e atuando por meio de um conjunto de ações, sendo, portanto, responsável pela resposta em tempo real do agente. É formado por um conjunto de controladores nebulosos, sendo que apenas um é ativo em um dado instante de simulação, segundo a ilustração da figura 5.4. Neste contexto, cada controlador presente no agente representa um *comportamento* específico, e as regras deste controlador definem os comandos enviados ao ambiente. De fato, um controlador está associado a um fundamento para a prática do futebol, como passar, chutar, driblar, entre outros. Para os experimentos relativos a este trabalho de tese foram utilizados o seguinte grupo de controladores:

- *Initialize-Player*: implementa o conjunto de mensagens iniciais para o reconhecimento do agente dentro do simulador;
- *Kick-Off-Position*: posiciona o agente para saída de bola no centro de campo;
- *Move-To-Position*: move o agente para uma determinada posição do campo;
- *Move-To-Ball*: move o agente em direção à bola;

[3](c.f. item 5.4)

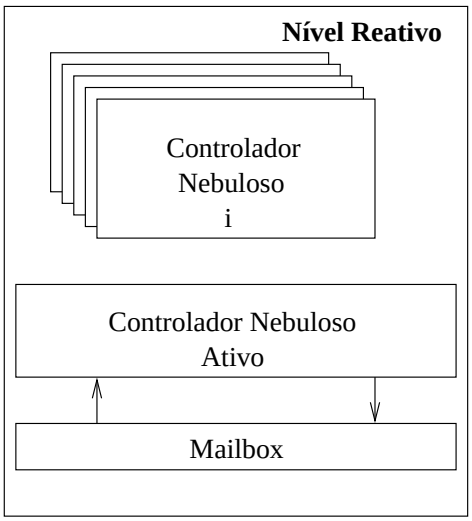


**Figura 5.4:** *Nível Reativo*

- *Pass-Ball*: passa a bola para um companheiro de equipe;
- *Kick-To-Goal*: chuta a bola no gol adversário;
- *Dribble-Opponent*: dribla o oponente dentro do campo de visão do agente;
- *Drive-Ball-Fwd*: conduz a bola a frente;
- *Get-Ball-Control*: busca uma aproximação com adversário na tentativa de tomar a bola;
- *Tackle*: obstrui a passagem do adversário;
- *Follow-Opponent*: segue um oponente buscando a sua marcação;
- *Rounding-Opponent*: cerca o oponente sem tentar tomar a bola;
- *Watch-Ball*: busca e mantém a bola dentro do seu campo de visão;
- *Catch-Ball*: defende uma bola chutada no gol.

O conjunto de controladores de um agente é determinado pelo grupo o qual pertence. Assim, um goleiro não necessita de um controlador que chute no gol adversário, da mesma forma que um atacante não precisa de um controlador que defenda a bola.

A utilização de controladores nebulosos se justifica com relação, por exemplo, aos controladores clássicos na medida em que estes últimos exigem um modelo matemático do ambiente, o que é inviável, mesmo que aproximadamente. Desta forma é possível garantir as exigências de tempo

real do domínio, uma vez que um controlador nebuloso é um sistema determinístico. Não obstante, há ainda a adequação deste tipo de solução às demandas evolutivas do nível reativo, permitindo a codificação dos seus parâmetros por meio de um mapeamento genético.

### Nível Instintivo

O nível instintivo é responsável pelo reconhecimento do estado do ambiente e pela execução de sua meta local. Para tanto, um SBC de ciclo único atualiza um conjunto de informações simbólicas e as envia até o nível cognitivo, além de determinar o controlador nebuloso ativo no nível reativo de acordo com a meta atual, como demonstrado na figura 5.5. A base de fatos é formada pela informação de percepção recebida do ambiente, além da meta local escolhida pelo nível cognitvo. A base de regras é gerada pelo usuário/especialista do sistema de forma a codificar o conhecimento necessário para o funcionamento do nível. É admitido o uso de uma representação de conhecimento híbrida, envolvendo regras de produção, quadros (do inglês “frames”) (Minsky, 1975) e lógica.

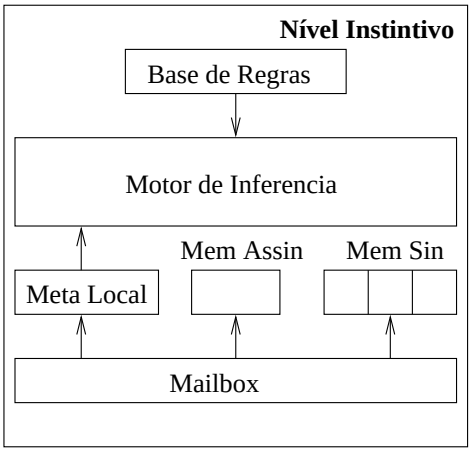


**Figura 5.5:** *Nível Instintivo*

Uma meta é perseguida por meio de uma seqüência de comportamentos reativos. No entanto, o nível instintivo verifica, a cada mudança de estado, as condições associadas a meta atual, que, uma vez invalidada, solicita ao nível cognitivo uma nova meta local para o sistema. Uma nova meta determina a escolha de um novo conjunto de regras utilizada pelo nível instintivo.

O estado do jogo é definido por um conjunto de variáveis simbólicas que devem ser inferidas e posteriormente enviadas ao nível cognitivo. As condições destas inferência dependem de alguns valores limiares determinados por algum método de aprendizado, como o aprendizado por reforço Martins (2003).

De acordo com o modelo genérico, o nível instintivo apresenta uma memória - cujo tamanho é um parâmetro da implementação - capaz de armazenar a percepção do ambiente. O parâmetro associado a memória determina a quantidade de quadros que podem ser usados em um ciclo de inferência. Isso significa dizer que, supondo uma memória de tamanho 3 e uma taxa de atualização de informação de percepção de 150ms, o SBC poderá fazer uso da informação sensorial contida nos instantes $t$, $t-150$ e $t-300$.

A memória é ainda divida em um buffer síncrono e outro assíncrono. O primeiro armazena a informação visual do ambiente e o segundo as informações enviadas pelo árbrito da partida. Essa medida faz-se necessária uma vez que qualquer mensagem enviada pelo árbrito tem o poder inerente de modificar o estado do jogo.

Como exemplo, suponha a meta local `mark_ball`. Esta meta remete à marcação do jogador adversário que está com a bola. Uma vez que o agente responsável por esta meta encontra-se a uma certa distância da bola, ele deve executar a seguinte seqüência de comportamentos reativos, considerando a lista de controladores apresentada na subseção anterior: *Watch-Ball*, *Move-To-Ball*, *Rounding-Opponent*. Na medida em que o agente consegue implementar esta seqüencia, ele atualiza suas variáveis simbólicas sinalizando ao nível cognitivo que a meta foi cumprida, e fica aguardando a determinação de uma nova. No entanto, caso o conjunto de condições que sustentam a meta atual não mais se verifica, o nível instintivo comunica o fato ao nível cognitivo e fica na expectativa de uma nova meta a ser cumprida.

**Nível Cognitivo**

O nível cognitivo tem o papel tem integrar o agente dentro do SMA, viabilizando a coordenação dos objetivos gerais com ações individuais estratégicas. Para isso, ele define socialmente as metas globais do agente, para posteriormente adequá-las as suas metas locais. Não obstante, o nível cognitivo provê ainda outros mecanismos deliberativos de alto nível, como planejamento em nível social e local, o que permite transformar o ambiente de acordo com os seus propósitos.

O nível cognitivo não exerce uma interferência direta sobre o nível reativo. No entanto a especificação do modelo genérico é atendida uma vez que a escolha de uma meta local seleciona um conjunto específico de regras no nível instintivo, o que indiretamente, determina o comportamento reativo mais adequado ao ambiente.

Um SBC simbólico opera por meio de uma base de fatos fomentada pelas informações simbólicas enviadas pelo nível instintivo, bem como pelas mensagens enviadas pelos demais agentes da comu-

nidade, utilizando lógica como representação de conhecimento e PARLA (da Costa e Bittencourt, 1997) como linguagem de comunicação interagentes. Além disso, o nível cognitivo dispõe de duas bases de regras, uma social e outra local, como descrito na figura 5.6. A primeira estabelece regras para a escolha da meta global, que deve ser definida por processo de cooperação entre os agentes. Estabelecido o acordo, a base local, escolhe uma meta local de acordo com o papel do agente na equipe.

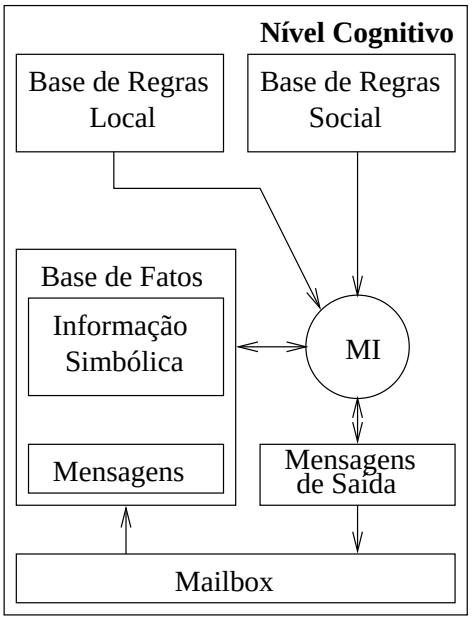


**Figura 5.6:** *Nível Cognitivo*

Dada a hierarquia de decisões dentro do agente, permite-se um ciclo de execução de maior complexidade para o nível cognitivo. Desta forma, é possível implementar um SBC orientado por planos, sem as restrições de tempo real do ambiente, abordando aspectos deliberativos de mais alto nível.

## 5.4 A Expert-Coop++

Para a construção do UFSC-Team - equipe desenvolvida utilizando a arquitetura do agente autônomo concorrente para atuação no Soccerserver - foi utilizada a Expert-Coop++, uma biblioteca, orientada a objetos, destinada a auxiliar no desenvolvimento de SMA cognitivos sob restrições de tempo real do tipo melhor esforço. O ambiente consiste em uma biblioteca de classes, implementada na linguagem de programação C++ (da Costa et al., 2003).

A adequação às restrições de tempo real de um domínio é possível devido a classes específicas para a construção de comportamentos reativos, por intermédio de controladores nebulosos. Este tipo de solução garante robustez ao sistema, e especialmente, o sincronismo com o ambiente. Para a construção deste controladores, a Expert-Coop++ incorporou a biblioteca CNCL (Junius e Stepple, 1997), e adicionalmente, alguns comportamentos específicos ao domínio Soccerserver também foram implementados. Não obstante, controladores para diferentes aplicações podem ser livremente adicionados.

Quanto aos mecanismos para implementação de SBC's, a Expert-Coop++ prevê a abstração dos dois componentes principais: um método de resolução de problemas e a base de conhecimento.

O método de resolução de problemas pode ser controlado por encadeamento progressivo ou regressivo, determinado por um parâmetro de instanciação da classe. Além disso, é possível a utilização de um SBC com múltiplas bases de conhecimento, como a arquitetura descrita no item 4.2. Isto é possível pela múltipla instanciação da classe que implementa bases de regras.

Para a implementação da arquitetura que prevê a interação entre múltiplos SBC's, a Expert-Coop++ disponibiliza classes que implementam a comunicação entre sockets UDP/IP e do domínio UNIX. A partir disso, é possível a instanciação de múltiplos SBC que se comunicam por um destes métodos.

As bases de conhecimento prevêem representações em quadros, lógica e regras de produção. A representação lógica, entretanto, obedece ao padrão ( ⟨ objeto ⟩ ⟨ atributo ⟩ ⟨ valor ⟩ ). A figura 5.7 descreve a sintaxe dos formalismos de representação de conhecimento suportadas.

```
(frame 0
(object goal r)
(distance 55.10) (direction -30.00) (distchng -30.00)
(dirchng  0.00) (bodydir -30.00) (headdir  0.00)
(object flag r b)
(distance 47.90) (direction  7.00) (distchng  7.00)
(dirchng  0.00) (bodydir  7.00) (headdir  0.00))

(logic ( game state play_on )) 176
(logic ( reactive_behavior active drive_ball_fwd ))
```

**Figura 5.7:** *Formalismos de representação de conhecimento suportados pela Expert-Coop++: quadros e padrões lógicos*

As regras de produção na Expert-Coop++ têm a seguinte sintaxe:

$$(if\ \langle premissas \rangle)\ (filter\ \langle ?variaveis \rangle)\ (then\ \langle conclusoes \rangle\ \langle acoes \rangle)$$

As premissas e conclusões suportam os formalismos já citados, enquanto as ações correspondem a mensagens enviadas a outros níveis do agente. Desta forma, a Expert-Coop++ é caracterizada por uma representação híbrida de conhecimento. A figura 5.8 apresenta uma regra sob esta sintaxe.

```
(rule_003
  (if     (logic  ( reactive_behavior status reactive_enable ))
          (logic  ( ball proximity ?x1 ))
          (frame  ( ?zy1  ((ball) (distance ?x2)))))
  (filter ( != ?x1 far )
          ( > ?x2 7.0 ))
  (then   (logic  ( ball proximity far ))
          (logic  ( ball control unknown ))
          (message ((to Cognitive) (from Instintive)
                    (deadline 0) (grade 0.0) (alpha 0.0) (round 0.0)
                    (body (INFORM ((logic ( ball proximity far )))))))
          (message ((to Cognitive) (from Instintive)
                    (deadline 0) (grade 0.0) (alpha 0.0) (round 0.0)
                    (body (INFORM ((logic ( ball control unknown ))))))))
```

**Figura 5.8:** *Sintaxe das regras de produção utilizadas pela Expert-Coop++*

O campo `if` engloba o conjunto de premissas da regra. Correspondem a perguntas feitas a base de fatos, e podem utilizar tanto a representação em quadros como lógica. No exemplo, a regra *rule_003* indaga quanto a presença do padrão lógico que indica o status do nível reativo (habilitado/desabilitado) e quanto à proximidade da bola. Esta última premissa faz uso de uma variável, reconhecida pelo caracter "?". A utilização de variáveis pela Expert-Coop++ aumenta a flexibilidade na composição de regras, o que resulta em uma base de menor tamanho. A premissa restante faz referência a um quadro, onde pergunta-se pela distância da bola, também utilizando uma variável.

O campo `filter` permite a instanciação das variáveis contidas nas premissas. A Expert-Coop++ reconhece o tipo de variável utilizada, simbólica ou numérica, sem que se faça qualquer tipo de declaração a priori. Uma vez que todas as operações com variáveis sejam verdadeiras - no exemplo, se $x1$ for diferente de `far` e $x2$ for maior que 7 - então as sentenças contidas no campo `then` passam a ser verdadeiras, e as ações são executadas.

Observa-se, ainda no campo `then` a presença de uma estrutura do tipo `message`. Estas correspondem as ações dos SBC's. São mensagem contendo informações simbólicas para o nível cognitivo

- `Cognitive` na regra - ou a troca do comportamento reativo para o nível correspondente. Os demais campos correspondem ao grau de certeza da informação (`grade`) e informações sobre restrições de tempo real (`deadline`, `alpha` e `round`). No exemplo acima, os valores em zero indicam a não relevância desta informação para a regra. As mensagens estão sob o formato *Parla* (da Costa e Bittencourt, 1997), a linguagem de comunicação entre agentes utilizada pela Expert-Coop desde sua versão original.

A troca de mensagens é implementada por sockets no domínio UNIX, cujas classes específicas também estão disponíveis na Expert-Coop++. Estas classes correspondem a uma espécie de *mailbox* do nível, e distinguem as mensagens oriundas do ambiente, dos outros agentes, e dos demais níveis do agente.

A relação da Expert-Coop++ com a arquitetura do agente autônomo concorrente justifica-a como melhor abordagem de desenvolvimento para o UFSC-Team. Entretanto, sua principal lacuna está no processo de aquisição de conhecimento. O conhecimento do agente deve ser provido pela construção das bases de conhecimento de nível instintivo e cognitivo. E para tanto, não há nenhuma metodologia e/ou abordagem, a não ser a codificação segundo experiência empírica do engenheiro de conhecimento.

Dada estas restrições, os principais requisitos da metodologia e/ou abordagem de AC são:

- a inserção do agente em um SMA coordenado por metas comuns;
- a divisão da complexidade do conhecimento em múltiplos níveis de abstração;
- ferramentas de especificação que se adaptem à formalismos de representação de conhecimento híbrido.

## 5.5 Implementação da Abordagem

### 5.5.1 Descrição do Cenário

A abordagem de especificação de conhecimento proposta é utilizada com o objetivo de implementar uma jogada pré-planejada entre agentes de uma equipe, segundo as restrições da arquitetura de agente autônomo concorrente, Expert-Coop++ e do simulador Soccerserver. Esta jogada faz parte do conjunto de jogadas pré-planejadas que implementam a estratégia de uma equipe, o UFSC-Team.

A implementação a ser apresentada não abrange a especificação completa da equipe, mas apenas uma das jogadas que fazem parte do conjunto de jogadas pré-planejadas da estratégia do UFSC-Team. No nível social, será apresentada a especificação completa da equipe, porém, no nível individual, apenas a especificação da jogada do cenário da figura 5.9.

Este cenário descreve uma jogada envolvendo três agentes, cujo objetivo final é o arremate a gol por parte do agente número 9, depois de uma troca de passes entre os agentes números 8 e 7. Ao aplicar a abordagem neste cenário, é possível analisá-la quanto seu poder de atuação em SMA, estabelecendo a coordenação de papéis, a alocação destes papéis de acordo com as potencialidades de cada agente, seus mecanismos de decisão interna, tudo em benefício da meta global da equipe.

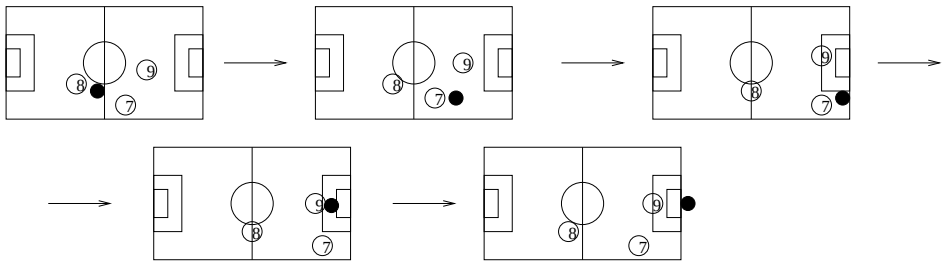

**Figura 5.9:** *Cenário para a execução de uma jogada*

### 5.5.2 Implementação

#### Nível Social

A abordagem tem início a partir da especificação do nível social do SMA, onde define-se sua estrutura organizacional. O resultado final desta etapa deve ser a geração de uma Rede de Petri (RP) que represente um nível de coordenação para o sistema, por meio de planos globais.

Uma equipe de futebol é constituída por três grupos de jogadores: defensores, meias e atacantes. Os jogadores do primeira grupo devem cumprir um papel baseado em tarefas defensivas, que visam evitar o gol do time adversário, e recuperar a posse de bola para a sua equipe. O grupo dos atacantes têm um papel baseado em tarefas ofensivas, que buscam a condução da bola até o campo adversário para que possam concluir a gol. O papel dos meias é colaborar tanto em atividades defensivas como ofensivas. No primeiro caso, auxiliando os defensores na tentativa de recuperação de bola, e no segundo caso, na escolha das jogadas de ataque que devem ser executadas.

Esta relação entre grupos de jogadores e os respectivos papéis é resumida e formalizada na tabela 5.1.

| Grupo | Papel |
|---|---|
| Defensores | Defender |
| Meias | Defender e Atacar |
| Atacantes | Atacar |

**Tabela 5.1:** *Relação entre grupos e papéis no Soccerserver*

Um bom time de futebol parte da premissa de que antes de aplicar qualquer estratégia, a equipe deve deter a posse da bola. Assim, antes de definir que jogada coletiva deve ser implementada, a equipe deve recuperar a posse bola.

Uma vez com a bola, uma jogada de ataque deve ser escolhida de acordo com as contingências do ambiente e da capacidade coletiva da equipe. As jogada de ataque são escolhidas, via de regra, pelos meio-campistas, e dividem-se em dois tipos principais: pelo centro, ou pelos lados do campo.

Sendo assim, o planejamento social da equipe é o seguinte: ao recuperar a posse de bola, esta deve ser levada até os meias, que escolherão uma das três jogadas de ataque, pelo centro, pelo lado esquerdo ou pelo lado direito, de acordo com as contingências do ambiente. Em jogadas pelos lados, a bola será levada até o lado do campo, e cruzada para a área, na espera que algum agente conclua a gol. Na jogada centralizada, será feito um lançamento para um agente, na tentativa que ele possa dominar a bola e chutar a gol.

Estabelecidos os requisitos da estratégia social , deve-se definir a estrutura organizacional. Como estrutura organizacional, poder-se-ia adotar modelos como o AGR (Ferber et al., 2004), uma evolução do modelo AALAADIN (Ferber e Gutknecht, 1998), o $\mathcal{M}$ oise$^+$ (Hübner et al., 2002), ou mesmo a Rede Contratual (Smith, 1980). Contudo, dada a multiplicidade de tarefas adotadas pelos diferentes papéis de um agente em um jogo de futebol, e a necessidade de representação explícita e direta das relações entre os múltiplos objetivos deste tipo de ambiente, adota-se como estrutura organizacional o modelo de Esquemas Sociais (Lugo et al., 2001).

Um esquema social é uma formulação que engloba um conjunto específico de tarefas utilizadas por um conjunto de agentes que compartilham objetivos sociais. Neste contexto, uma tarefa pode estar associada a um ou mais agentes, sujeita a um esquema de coordenação, que neste caso é planejamento.[4]

Formalmente, um esquema social (Lugo et al., 2001) é uma tupla $es = \langle o, T, T_f, P \rangle$, onde:

- $o$ é o objetivo a ser alcançado por $es$;

[4] c.f. item 2.2.4

- $T$ é um conjunto de tarefas;
- $T_f \subseteq T$ é um subconjunto de tarefas que uma vez finalizadas indicam que *es* foi finalizado. Na medida em que se aplica a abordagens em ambientes não-determinísticos, o sucesso de de *o* está sujeito à verificação das variáveis de estado de $T_f$ e/ou do ambiente.
- $P$ é um conjunto de papéis a serem assumidos pelos agentes responsáveis pelas tarefas $T$. Assim, um papel $p \in P$ corresponde ao conjunto de funções de uma agente dentro de *es*.

Assim, um *es* é dito finalizado com sucesso se *o* foi alcançado por intermédio da execução de $t_k \in T_f$.

Por sua vez, uma tarefa $t_k \in T$ de um *es* é uma tupla $t_k = \langle IDes_k, so_k, PC_k, P_k \rangle$, onde:

- $IDes_k$ é um identificador de tarefa único dentro de *es*;
- $so_k$ é um sub-objetivo que deve ser alcançado pela execução da tarefa;
- $PC_k \subset T$ é o conjunto de tarefas que deve anteceder a execução de $t_k$;
- $P_k \in P$ é o conjunto de papéis necessários a execução da tarefa.

Assim, para que os requisitos de um ambiente sejam atendidos, é necessário a especificação de um conjunto de esquemas sociais além da determinação das relações de precedência e causalidade entre eles.

A especificação social do UFSC-Team prevê cinco esquemas sociais, mais o esquema social inicial.

Sendo $es_0$ o esquema social de partida de uma equipe, ou seja, aquele que caracteriza a ausência de objetivo, tem-se $es_0 = \langle o_0, T_0, T_{f0}, P_0 \rangle$, tal que:

- $o_0$ = $\langle$ none $\rangle$;
- $T_0$ = $\langle\ \rangle$, $T_{f0}$ = $\langle\ \rangle$, $P_0$ = $\langle\ \rangle$.

Seja $es_1$ o esquema social cujo objetivo é recuperar a posse de bola para a equipe, $es_1 = \langle o_1, T_1, T_{f1}, P_1 \rangle$, tal que:

- $o_1$ = $\langle$ get_ball_control $\rangle$;

- $T_1$ = ⟨ marcar adversário, marcar a bola, tomar a bola ⟩;
- $T_{f1}$ = ⟨ tomar a bola ⟩;
- $P_1$ = ⟨ defender ⟩.

onde $T_1 = \langle t_1, t_2, t_2 \rangle$, tal que:

- $t_1$ = ⟨ `mark_opponent`, marcar adversário, -, ⟨ defender ⟩ ⟩;
- $t_2$ = ⟨ `mark_ball`, cercar adversário com a bola, -, ⟨ defender ⟩ ⟩;
- $t_3$ = ⟨ `take_ball`, tomar a bola do adversário, `mark_ball`, ⟨ defender ⟩ ⟩;

Seja $es_2$ o esquema social cujo objetivo é levar a bola até um dos meias da equipe, para que estes possam escolher qual jogada deve ser executada, $es_2 = \langle o_2, T_2, T_{f2}, P_2 \rangle$, tal que:

- $o_2$ = ⟨ go_to_middle_field ⟩;
- $T_2$ = ⟨ passar bola, conduzir bola, passar bola para o meia⟩;
- $T_{f2}$ = ⟨ passar bola para o meia ⟩;
- $P_2$ = ⟨ atacar ⟩.

onde $T_2 = \langle t_1, t_2, t_3 \rangle$, tal que:

- $t_1$ = ⟨ `pass_ball_x`, passar a bola para *x*, - ,⟨ defender ⟩ ⟩
- $t_2$ = ⟨ `drive_ball`, conduzir a bola para a frente, -, ⟨ defender ⟩ ⟩;
- $t_3$ = ⟨ `pass_ball_mf`, passar bola para os meias, -, ⟨ denfender ⟩ ⟩.

Seja $es_3$, o esquema social que executa uma jogada de ataque pelo centro, considerando que a bola já esteja com um dos meias da equipe, $es_3 = \langle o_3, T_3, T_{f3}, P_3 \rangle$, tal que:

- $o_3$ = ⟨ center_attack_play ⟩;
- $T_3$ = ⟨ conduzir a bola, passar a bola para atacante, chutar a gol ⟨;

- $T_{f3}$ = ⟨ chutar a gol ⟩;
- $P_3$ = ⟨ atacar ⟩.

onde $T_3 = \langle t_1, t_2, t_3, t_4 \rangle$:

- $t_1$ = ⟨ `drive_ball`, conduzir bola em direção ao gol, -, ⟨ atacar ⟩ ⟩;
- $t_2$ = ⟨ `pass_ball_9`, passar a bola para atacante, -, ⟨ atacar ⟩ ⟩;
- $t_3$ = ⟨ `kick_to_goal`, chutar a bola no gol, `pass_ball_9`, ⟨ atacar ⟩ ⟩.

Seja $es_4$ o esquema social que implementa uma jogada de ataque pelo lado direito do campo, $es_4 = \langle o_4, T_4, T_{f4}, P_4 \rangle$, tal que:

- $o_4$ = ⟨ rws_attack_play ⟩;
- $T_4$ = ⟨ passar bola, conduzir bola, cruzar bola para área, chutar a gol ⟩;
- $T_{f4}$ = ⟨ chutar a gol ⟩;
- $P_4$ = ⟨ atacar ⟩.

onde $T_4 = \langle t_1, t_2, t_3, t_4 \rangle$, tal que:

- $t_1$ = ⟨ `pass_ball_7`, passar a bola para o agente 7, -, ⟨ atacar ⟩ ⟩;
- $t_2$ = ⟨ `drive_ball`, conduzir bola até a linha de fundo, `pass_ball_7`, ⟨ atacar ⟩ ⟩;
- $t_3$ = ⟨ `pass_ball_9`, passar a bola para o agente 9, `drive_ball`, ⟨ atacar ⟩ ⟩;
- $t_4$ = ⟨ `kick_to_goal`, chutar a bola a gol, `pass_ball_9`, ⟨ atacar ⟩ ⟩.

Por fim, seja $es_5$ o esquema social que implementa uma jogada de ataque pelo lado esquerdo do campo, $es_5 = \langle o_5, T_5, T_{f5}, P_5 \rangle$, tal que:

- $o_5$ = ⟨ lws_attack_play ⟩;
- $T_5$ = ⟨ passar bola, conduzir bola, cruzar bola para área, chutar a gol ⟩;

- $T_{f5}$ = ⟨ chutar a gol ⟩;
- $P_5$ = ⟨ atacar ⟩.

onde $T_5 = \langle t_1, t_2, t_3, t_4 \rangle$, tal que:

- $t_1$ = ⟨ `pass_ball_11`, passar a bola para o agente 11, -, ⟨ atacar ⟩ ⟩;
- $t_2$ = ⟨ `drive_ball`, conduzir bola até a linha de fundo, `pass_ball_7`, ⟨ atacar ⟩ ⟩;
- $t_3$ = ⟨ `pass_ball_9`, passar a bola para o agente 9, `drive_ball`, ⟨ atacar ⟩ ⟩;
- $t_4$ = ⟨ `kick_to_goal`, chutar a bola a gol, `pass_ball_9`, ⟨ atacar ⟩ ⟩.

A geração do nível de coordenação social pela utilização de RP está sujeita a formalização de esquemas sociais como linguagem de representação de conhecimento. Sendo assim, a RP tem o seguinte conjunto $\mathcal{T}$ de termos:

- $\mathcal{C}_1 = \{es, o, T, T_f, P\}$
- $\mathcal{C}_2 = \{es_o, es_1, es_2, o_1, o_2, T_1, T_2, T_{f1}, T_{f2}, P_1, P_2\}$
- $\mathcal{C} = \mathcal{C}_1 \cup \mathcal{C}_2$
- $\mathcal{V} = \{x_1, x_2, ..., x_n\}$ tal que $n \in \mathbb{N}$
- $\mathcal{F} = \{f, c\}$
- $f(c(es, x_i), c(o, x_{i+1}), c(T, x_{i+2}), c(T_f, x_{i+3}), c(P, x_{i+4}))$

Desta forma, a ficha da RP é uma base de conhecimento de esquemas sociais, que podem ser acessadas pelos seus diferentes atributos.

Dada a descrição da jogada, a coordenação entre os esquemas sociais é modelada pela RP da figura 5.10. Esta rede descreve um planejamento multiagente que coordena as atividades dos agentes do ambiente.

A rede é interpretada da seguinte forma. Ao iniciar a operação no ambiente, não há qualquer esquema social instanciado ($es_0$). Neste estado, a transição $t_1$ invoca o esquema social $es_1$, cujo objetivo é recuperar a posse de bola, levando a ficha para o lugar $P_2$. Segundo a transição $t_2$, o sistema

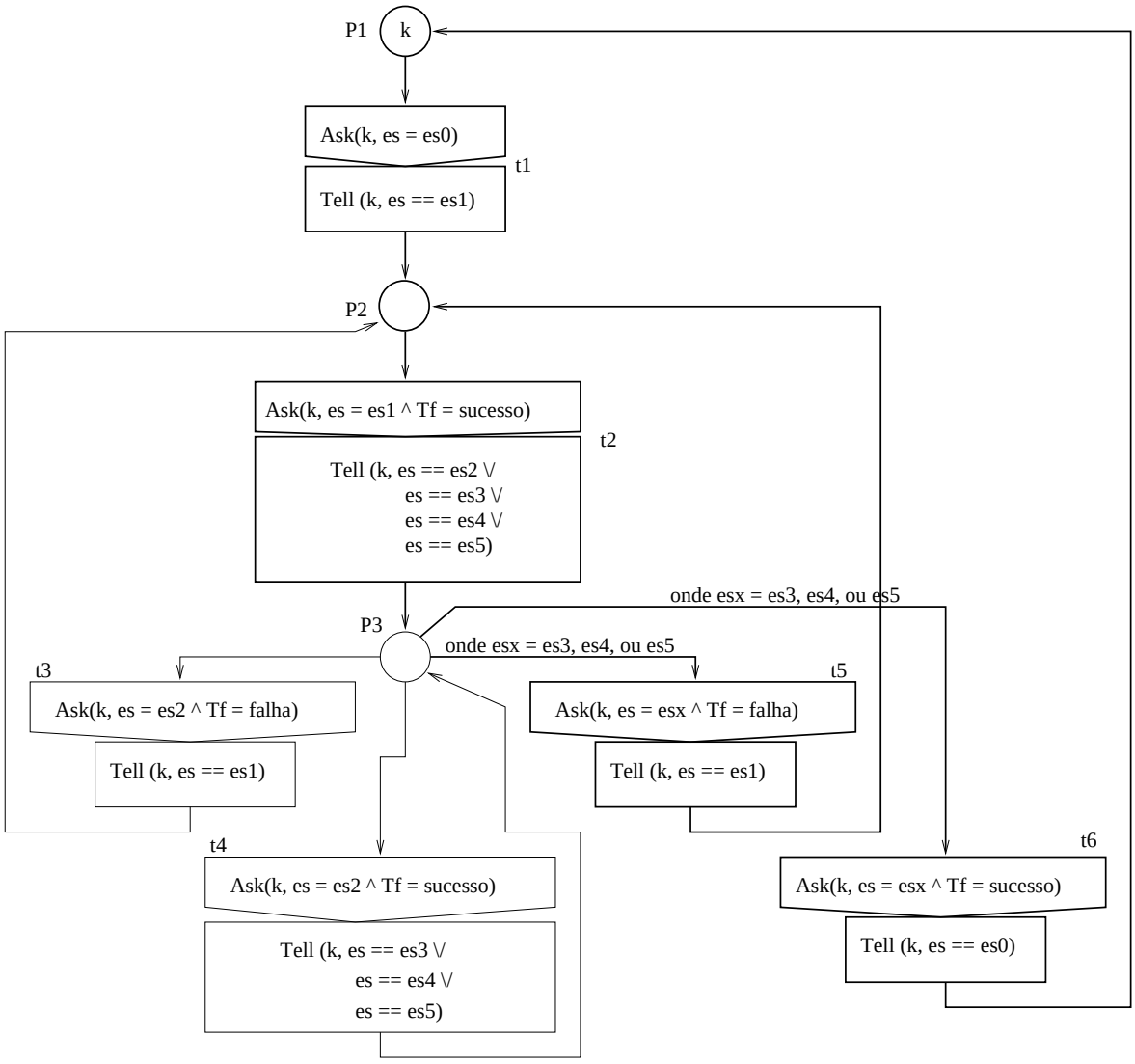


**Figura 5.10:** *Nível social para a execução da jogada*

evolui apenas se o esquema social $es_1$ é realizado com sucesso, indicado pela premissa $T_f = Sucesso$, o que significa, segundo a notação de esquemas sociais, que uma das tarefas que finaliza $es_1$ levou o sistema ao objetivo $o_1$.

Pela transição $t_2$, *es* pode assumir um dos quatro valores: $es_2$, $es_3$, $es_4$, ou $es_5$. A determinação de *es* indepedende da estrutura da rede social, em uma análise do modelo, ou independe de variáveis sociais, em uma análise do ambiente. O que vai determinar a escolha de *es* na função *Tell* em $t_2$ é a posição no campo em que a bola foi roubada. Neste caso, trata-se de uma variável determinada pelo nível individual dos agentes. Entretanto o nível social deve apontar as possibilidades de planos após a conclusão de $es_1$.

Se a bola foi recuperada em uma posição de defesa, ela deve ser levada até o meio de campo, para um dos meias ($es_2$). Ao realizar este objetivo, deve-se definir por uma das três jogadas de ataque ($es_3$, $es_4$ ou $es_5$) [5]. Segundo a rede, se a jogada escolhida falha, a equipe retorna à meta de recuperação da bola ($es_1$). Se a jogada é finalizada com sucesso, um chute a gol é realizado, o que pode implicar um gol, ou na perda da posse de bola. Assim, o sistema volta ao esquema social $es_0$.

O número de planos tratáveis pelo sistema é o número de caminhos possíveis dentro da RP, ou seja, o número de invariantes de lugar da rede, que neste caso são cinco. A execução de um destes cinco invariantes leva o sistema para um dos dois estados: para a realização de um chute a gol ($\langle [t_1,t_2,t_4,t_6],[t_1,t_2,t_6] \rangle$), ou para a retomada da meta de defesa ($\langle [t_1,t_2,t_3],[t_1,t_2,t_4,t_5],[t_1,t_2,t_5] \rangle$).

Observa-se, ao contrário das abordagens apresentadas no item 3.5.1, que a especificação não faz qualquer referência a ações individuais. Neste caso, cabe aos agentes deliberarem autonomamente suas ações no sentido das metas do sistema, resolvendo possíveis problemas de coordenação dinamicamente. Este tipo de abordagem aumenta a flexibilidade de atuação dos agentes e a robustez do sistema.

É importante ressaltar que a especificação social não contém um nível de implementação, uma vez que ela apenas descreve a estratégia e coordenação multiagente empregada.

**Nível Individual**

Neste item é apresentada apenas implementação do cenário descrito na figura 5.9, representado pelo caminho destacado na rede social da figura 5.10.

[5] Adotou-se por uma notação simplificada na rede. Neste caso, cada esquema social deve ser representado por um conjunto igual ao esquema genérico adotado na rede.

O planejamento multiagente, modelado no nível de especificação social do sistema, define a estratégia de coordenação adotada. Para que esta estratégia seja posta em prática, este planejamento deve ser instanciado na base de conhecimento dos agentes, formando seu conhecimento social. Este conhecimento representa aquilo que os agentes sabem a respeito do estado dos demais agentes do sistema, de modo que eles possam definir suas estratégias de atuação individual de acordo com os objetivos do sistema.

O nível individual deve refinar este conhecimento social dos agentes até especificar o conjunto de ações reativas, segundo a arquitetura genérica.[6] Desta forma, o planejamento global deve gerar um planejamento individual, que por sua vez gera seqüências de comportamentos coordenadas com o plano individual do agente. Cada comportamento, por sua vez, deve gerar ações específicas no ambiente.

Particularizando a arquitetura genérica na arquitetura de agente autônomo concorrente, o planejamento global é referente à base social do nível cognitivo. O planejamento individual refere-se à base local do mesmo nível, enquanto a coordenação de comportamentos dá-se pelo nível instintivo. Finalmente, os comportamentos formam o nível reativo do agente.

O problema a ser resolvido neste ponto é como mapear o nível de coordenação social no conhecimento do agente, em seus diversos níveis de abstração estruturados hierarquicamente. Este mapeamento é definido segundo a tabela abaixo.

| Conceito Social | Conceito Individual |
|---|---|
| Objetivo $o$ | Meta Global $global\_goal$ |
| Tarefas $T$ | Metas Locais $local\_goal$ |
| Papéis $P$ | Conjunto de Comportamentos $cn$ |

**Tabela 5.2:** *Mapeamento entre conceitos sociais e individuais*

As bases de conhecimento do nível cognitivo abordam o conhecimento de forma simbólica, segundo uma representação de conhecimento em lógica, de acordo com o padrão $(\langle objeto \rangle \langle atributo \rangle \langle valor \rangle)$. Assim, o conjunto de termos utilizados pelas RP que modelam as bases de conhecimento do nível cognitivo do agente é o mesmo especificado no item 4.4.1.

Seja $a_x$, tal $x = 1, \ldots, 11$, cada um dos agentes da equipe.

O conhecimento simbólico é utilizado para representar o estado dos objetos do ambiente. Desta forma, um conjunto particular destas sentenças pode representar estados objetivos do ambiente. Por

[6] c.f. item 4.2

exemplo, a sentença $(ball\ control\ me)$ representa a conclusão da tarefa $T_{f1}$ por parte de $a_8$. A sentença $(ball\ control\ my\_team)$ informa para os outros agentes do sistema que $a_8$ cumpriu seu papel no ambiente e alcançou $o_1$ de $es_1$, e que $es_2$ deve ser instanciado.

A relação entre $T$, sentenças simbólicas das bases de conhecimento e $T_f$ dá-se da seguinte maneira. $T$, dentro do agente autônomo concorrente, é o conjunto de metas locais que podem ser selecionadas para alcançar $o$. Uma vez selecionada uma meta, $t \in T$ no contexto de $es$, ou $local\_goal$ no contexto do agente, esta gera uma seqüência coordenada de comportamentos que modificam as variáveis do ambiente. Os comportamentos disponíveis para um tarefa $t \in T$ no contexto de $es$, ou para uma meta $local\_goal$ no contexto do agente, são aqueles condizentes com o papel $P$ de $es$. As modificações geradas pela execução de um comportamento reativo atualizam as sentenças simbólicas do nível instintivo, que são enviadas ao nível cognitivo. Uma vez que estas sentenças indiquem um estado que represente que algum $t_f \in T_f$ foi executado com sucesso, o objetivo $o$ de $es$ foi alcançado, ou seja, a execução de uma $local\_goal$ levou a realização de uma $global\_goal$.

Desta forma, a figura 5.11 apresenta a especificação da base social de $a_8$.

No contexto do conhecimento do agente autônomo concorrente, $es_1$ agora é a meta global get_ball_control, e $es_4$ é rws_attack_play. $es_0$ é nomeada como none. Repare que a RP tem a mesma estrutura da RP social, isto por que ela instancia o planejamento social na base de conhecimento social do agente. Por conseqüência, as duas redes apresentam o mesmo conjunto de invariantes de lugar, ou seja, o mesmo conjunto de planos.

Por se tratar da base social de $a_8$, a premissa de $t_2$, que na especificação social perguntava sobre o sucesso de $T_{f1}$, foi substituída pela sentença que indica esta condição, $(ball\ control\ me)$. Uma segunda modificação diz respeito às premissas que verificam o sucesso ou falha da meta global, respectivamente, em $t4$ e $t5$. Novamente, substitui-se a premissa baseada em $T_f$ pela sentença simbólica que representa seu sucesso ou falha, respectivamente, $(game\ state\ goal\_r)$ e $(ball\ control\ opponent)$.

Segundo o mapeamento entre RP e regras apresentado no item 4.4.3, o agente 8 apresenta a seguinte base de conhecimento social, gerada a partir da rede da figura 5.11.

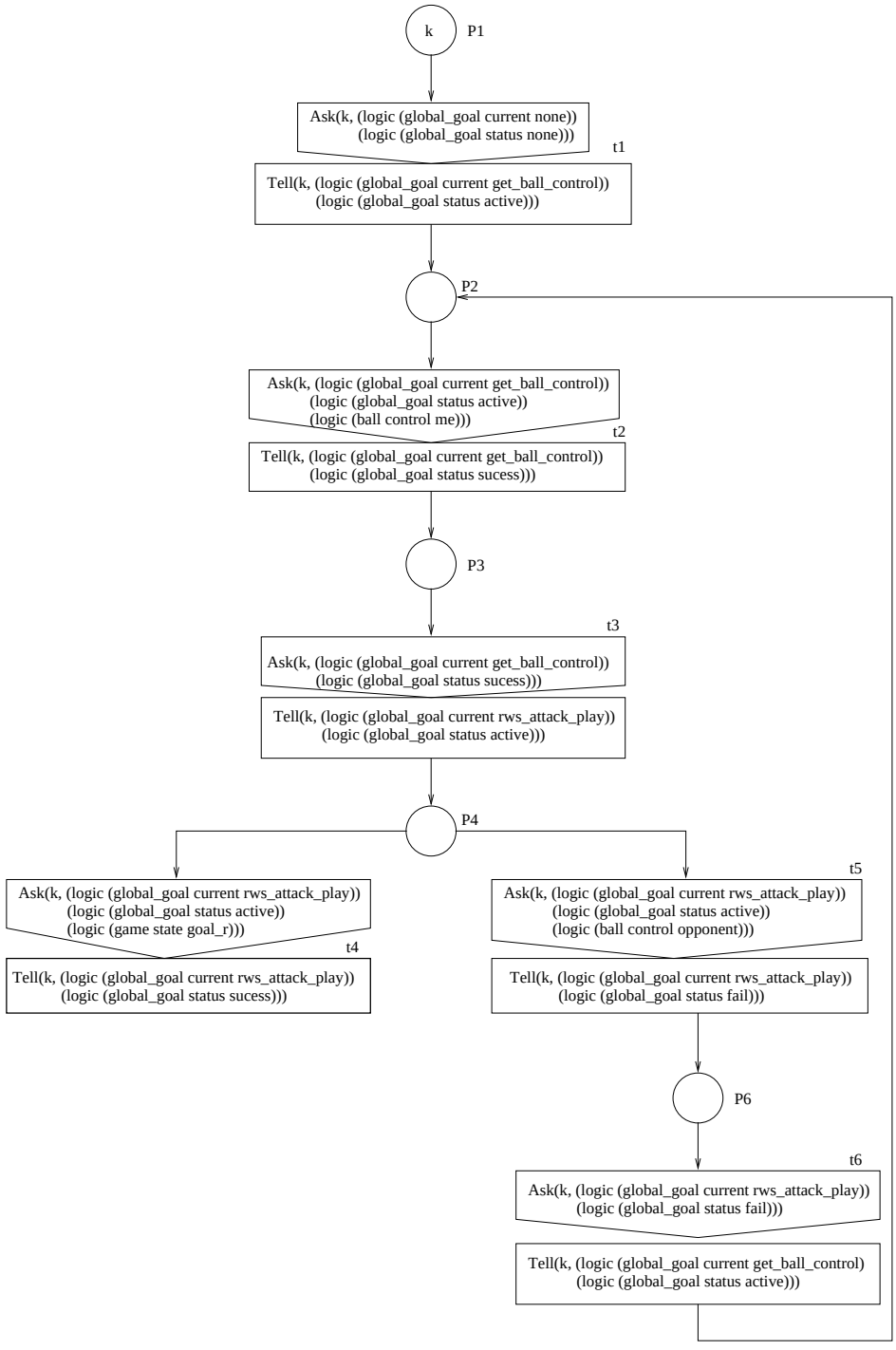


**Figura 5.11:** *Especificação da base social do agente 8*

```
((rule_001
    (if      (logic ( global_goal current none ))
             (logic ( global_goal status none )))
    (then    (logic ( global_goal current get_ball_control ))
             (logic ( global_goal status active ))))
(rule_002
    (if      (logic ( global_goal current get_ball_control ))
             (logic ( global_goal status active ))
             (logic ( ball control me )))
    (then    (logic ( global_goal current get_ball_control ))
             (logic ( global_goal status sucess ))))
(rule_003
    (if      (logic ( global_goal current get_ball_control ))
             (logic ( global_goal status sucess )))
    (then    (logic ( global_goal current rws_attack_play ))
             (logic ( global_goal status active ))))
(rule_004
    (if      (logic ( global_goal current rws_attack_play ))
             (logic ( global_goal status active ))
             (logic ( game state goal_r )))
    (then    (logic ( global_goal current rws_attack_play ))
             (logic ( global_goal status sucess ))))
(rule_005
    (if      (logic ( global_goal current rws_attack_play ))
             (logic ( global_goal status active ))
             (logic ( ball control opponent )))
    (then    (logic ( global_goal current rws_attack_play ))
             (logic ( global_goal status fail ))))
(rule_006
    (if      (logic ( global_goal current rws_attack_play ))
             (logic ( global_goal status fail )))
    (then    (logic ( global_goal current get_ball_control ))
             (logic ( global_goal status active )))))
```

O mesmo princípio é aplicado à especificação de $a_7$ e $a_9$, resultando em uma RP similar a de $a_8$. A diferença está nas sentenças que verificam as premissas associadas à realização de $T_f$, em cada um dos *es*. Assim, onde $a_8$ tem como premissa $(ball\ control\ me)$, $a_7$ e $a_9$ têm a sentença $(ball\ control\ my\_team)$.

A base de regras sociais dos agentes 7 e 9, que são iguais, é a seguinte:

```
((rule_001
    (if      (logic ( global_goal current none ))
             (logic ( global_goal status none )))
    (then    (logic ( global_goal current get_ball_control ))
             (logic ( global_goal status active ))))
(rule_002
    (if      (logic ( global_goal current get_ball_control ))
             (logic ( global_goal status active ))
             (logic ( ball control my_team )))
    (then    (logic ( global_goal current get_ball_control ))
             (logic ( global_goal status sucess ))))
(rule_003
    (if      (logic ( global_goal current get_ball_control ))
             (logic ( global_goal status sucess )))
    (then    (logic ( global_goal current rws_attack_play ))
             (logic ( global_goal status active ))))
(rule_004
    (if      (logic ( global_goal current rws_attack_play ))
             (logic ( global_goal status active ))
             (logic ( game state goal_r )))
    (then    (logic ( global_goal current rws_attack_play ))
             (logic ( global_goal status sucess ))))
(rule_005
    (if      (logic ( global_goal current rws_attack_play ))
             (logic ( global_goal status active ))
             (logic ( ball control opponent )))
    (then    (logic ( global_goal current rws_attack_play ))
             (logic ( global_goal status fail ))))
(rule_006
    (if      (logic ( global_goal current rws_attack_play ))
             (logic ( global_goal status fail )))
    (then    (logic ( global_goal current get_ball_control ))
             (logic ( global_goal status active )))))
```

Uma vez definida as bases de conhecimento sociais dos agentes, obtidas a partir da especificação do nível social, o próximo passo é a especificação da base de conhecimento local, segundo a arquitetura do agente autônomo concorrente. Esta base tem como função principal definir as metas locais dos agentes, escolhidas de acordo com a meta global do sistema. Do ponto de vista social, a base local corresponde à especificação e relação entre as tarefas que levam a realização do esquema social em execução.

A construção da base local do nível cognitivo dá-se pela explosão dos lugares $P_2$ e $P_4$ das redes que especificam a base de conhecimento social. No caso de $P_2$, trata-se da especificação das metas locais que levam a realização da meta global get_ball_control, enquanto no lugar $P_4$ é a especificação das metas locais que levam a realização da meta global rws_attack_play. A base de conhecimento local, a partir da interpretação do modelo, é constituída pelo somatório destas duas redes.

Sendo assim, seja $a_8$. Uma vez que a ficha $k$ alcança o lugar $P_2$ na rede da figura 5.11, pela construção hierárquica substituição de lugar, tem-se a instanciação da RP da figura 5.12, enquanto que uma ficha no lugar $P_4$ leva a substituição de lugar pela rede da figura 5.13. Para ambas as redes, tem-se o seguinte conjunto de termos $\mathcal{T}$[7]:

- $\mathcal{C}_1 = \{object, attribute, value\}$;
- $\mathcal{C}_2 = \{local\_goal, ball, agent\_7, current, status, proximity, control, positioning, none, mark\_ball, take\_ball, drive\_ball, pass\_ball, active, sucess, unknown, very\_close, me, ok\}$;
- $\mathcal{C}_3 = \{logic\}$;
- $\mathcal{C} = \mathcal{C}_1 \cup \mathcal{C}_2 \cup \mathcal{C}_3$;
- $\mathcal{V} = \{x_1, x_2, x_3\}$;
- $\mathcal{F} = \{f, c\}$ tal que:
  $f(logic\ (c(object, x_1)\ c(attribute, x_2)\ c(value, x_3)))$, onde:
  - $x_1 \in \{local\_goal, ball, agent\_7\}$;
  - $x_2 \in \{current, status, control, proximity, positioning\}$;
  - $x_3 \in \{none, mark\_ball, take\_ball, drive\_ball, , active, sucess, unknown, very\_close, me, ok\}$.

A execução da meta local get_ball_control dá-se pela realização de duas metas locais que devem ser executadas em seqüência: `mark_ball` e `take_ball`. Estas metas referem-se ao conjunto de tarefas $T_1$ de $es_1$. Na primeira, o agente procura pela bola até que ela se encontre dentro de seu campo de visão. A seguir, o agente vai em direção da bola até dominá-la. Uma vez inferido que o agente tem o controle da bola, $(ball\ control\ me)$, a rede transfere o fluxo de controle para a rede hierarquicamente superior, que então verifica a realização da meta global.

Já a meta global rws_attack_play, dentro do contexto de atuação de $a_8$, é realizada pela seqüencia de duas metas locais: `drive_ball` e `pass_ball`. Essas metas referem-se a um subconjunto de $T_2$ de

[7] O conjunto de termos de uma rede hierarquicamente inferior herda o conjunto de termos da rede superior. Entretanto, para fins didáticos, apresenta-se o conjunto de termos $\mathcal{T}$ da rede da base local.

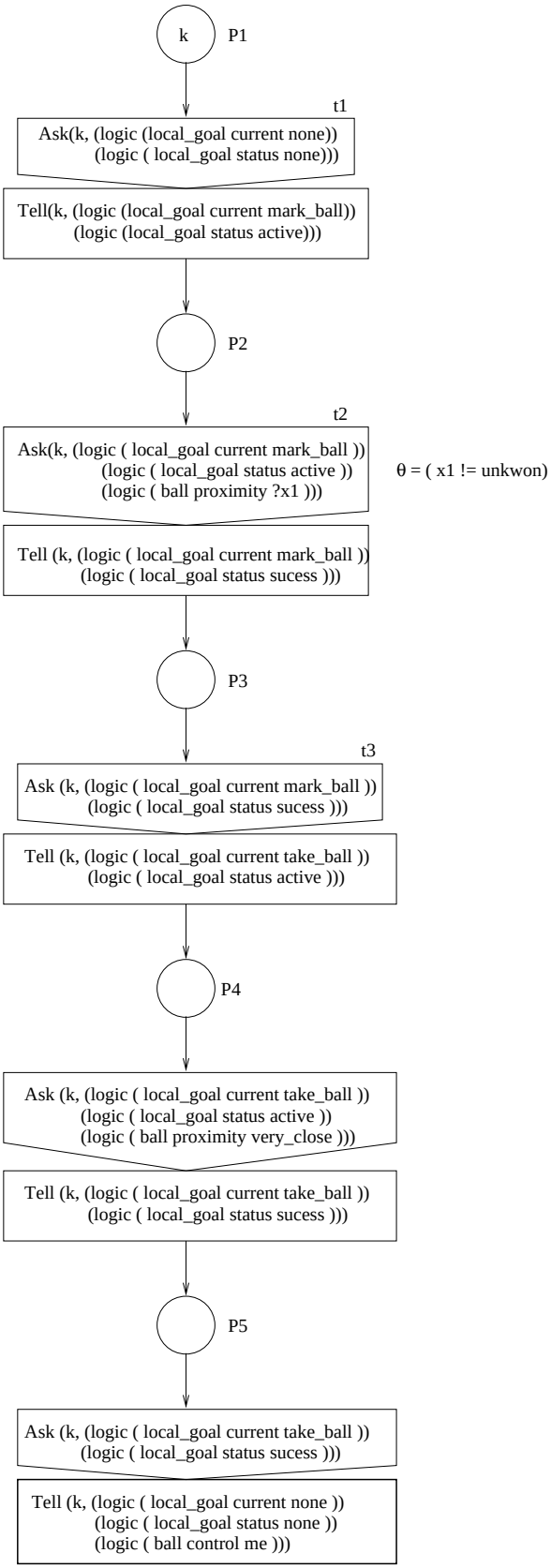


**Figura 5.12:** *Base local do agente 8*

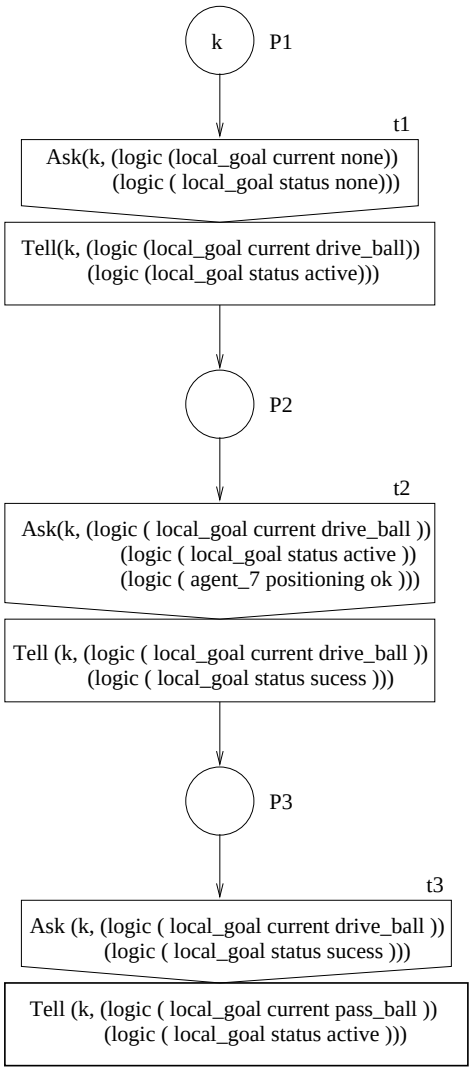


**Figura 5.13:** *Base local do agente 8*

$es_2$. Na primeira, o agente 8 conduz a bola até cruzar a linha divisória do gramado. Uma vez verificada esta condição, ele procura por $a_7$, esperando seu correto posicionamento para a continuação da jogada, $(agent\_7\ positioning\ ok)$. Uma vez verificada esta condição, sua meta local passa a ser a realização do passe para este agente.

Sendo assim, a partir das duas redes que especificam sua base local, o $a_8$ apresenta o seguinte conjunto de regras[8]:

```
((rule_001
  (if   (logic ( global_goal current get_ball_control ))
        (logic ( global_goal status active ))
        (logic ( local_goal current none ))
        (logic ( local_goal status none )))
  (then (logic ( local_goal current mark_ball ))
        (logic ( local_goal status active ))
        (message ((to Instintive) (from Cognitive)
                 (body (INFORM ((logic ( local_goal current mark_ball )))))))
        (message ((to Instintive) (from Cognitive)
                 (body (INFORM ((logic ( local_goal status active ))))))))
(rule_002
  (if   (logic ( global_goal current get_ball_control ))
        (logic ( global_goal status active ))
        (logic ( local_goal current mark_ball ))
        (logic ( local_goal status active ))
        (logic ( ball proximity ?x1 )))
  (filter ( ?x1 != unknown ))
  (then (logic ( local_goal current mark_ball ))
        (logic ( local_goal status sucess ))))
(rule_003
  (if   (logic ( global_goal current get_ball_control ))
        (logic ( global_goal status active ))
        (logic ( local_goal current mark_ball ))
        (logic ( local_goal status sucess )))
```

[8] A critério de simplificação, foram negligenciados os campos referentes á manipulação dos requisitos temporais da mensagens enviadas entre os níveis decisórios, ficando a estrutura da regra restrita ao conteúdo a ser trocado.

```
  (then (logic ( local_goal current take_ball ))
        (logic ( local_goal status active ))
        (message ((to Instintive) (from Cognitive)
                 (body (INFORM ((logic ( local_goal current take_ball )))))))
        (message ((to Instintive) (from Cognitive)
                 (body (INFORM ((logic ( local_goal status active ))))))))
(rule_004
  (if   (logic ( global_goal current get_ball_control ))
        (logic ( global_goal status active ))
        (logic ( local_goal current take_ball ))
        (logic ( local_goal status active ))
        (logic ( ball proximity very_close )))
  (then (logic ( local_goal current take_ball ))
        (logic ( local_goal status sucess ))))
(rule_005
  (if   (logic ( global_goal current get_ball_control ))
        (logic ( global_goal status active ))
        (logic ( local_goal current take_ball ))
        (logic ( local_goal status sucess )))
  (then (logic ( ball control me ))
        (logic ( local_goal current none ))
        (logic ( local_goal status none ))))
(rule_006
  (if   (logic ( global_goal current rws_attack_play ))
        (logic ( global_goal status active ))
        (logic ( local_goal active none ))
        (logic ( local_goal status none )))
  (then (logic ( local_goal current drive_ball ))
        (logic ( local_goal status active ))
        (message ((to Instintive) (from Cognitive)
                 (body (INFORM ((logic ( local_goal current drive_ball )))))))
        (message ((to Instintive) (from Cognitive)
                 (body (INFORM ((logic ( local_goal status active ))))))))
(rule_007
  (if   (logic ( global_goal current rws_attack_play ))
        (logic ( global_goal status active ))
        (logic ( local_goal active drive_ball ))
        (logic ( local_goal status active ))
        (logic ( agent_7 positioning ok )))
  (then (logic ( local_goal current drive_ball ))
        (logic ( local_goal status sucess ))))
```

```
(rule_008
  (if   (logic ( global_goal current rws_attack_play ))
        (logic ( global_goal status active ))
        (logic ( local_goal active drive_ball ))
        (logic ( local_goal status sucess )))
  (then (logic ( local_goal current pass_ball ))
        (logic ( local_goal status active )))))
```

Relembrando a sintaxe da Expert-Coop++, as sentenças do tipo *message* correspondem a troca de informações entre os níveis decisórios do agente. Neste caso, estas mensagens enviam a meta local inferida pelo nível cognitivo ao nível instintivo.

Dado o dinamismo de uma partida de futebol, um jogador, e neste caso, um agente, não deve ficar sem um papel, mesmo em um cenário em que ele não seja um participante ativo. Dado o cenário simplificado deste SMA em desenvolvimento, quando o sistema encontra-se em atividade defensiva, os agentes que possuem papéis apenas ofensivos devem adotar um comportamento padrão, de simples posicionamento em um local específico do campo. Assim, quando $es_1$ estiver em execução, $a_7$ deve se localizar no lado extremo direito, na linha divisória do campo, enquanto $a_9$ deve posicionar-se na junção esquerda das linhas que formam a grande área. Para isso, os comportamentos utilizados pelos agentes, devem ser os mesmos que implementam o papel defensivo de $a_8$.

Agora será possível observar como dá-se a coordenação de atividades entre os agentes sujeitos as mesmas metas do sistema. Na rede da figura 5.14 é modelada a base de conhecimento local de $a_7$[9]. Trata-se também de uma chamada ao procedimento *Run*, referente à base de conhecimento social de $a_7$. Entretanto, dada a diferença de funções entre os agentes, é possível constatar a diferença de atuação do agente com relação a meta global do sistema. Neste caso, enquanto o agente 8 busca dominar a bola, o agente 7 procura o melhor posicionamento para a recepção do futuro passe.

A meta local `mark_opponent` procura simular um atividade defensiva de um jogador de ataque. Entretanto, neste cenário, o agente apenas coloca-se em um posição mais recuada do campo, avançando logo em seguida, a medida que a metal global de tomada de bola for cumprida. Dada a similaridade de papéis entre $a_7$ e $a_9$ dentro da meta defensiva, esta também é a especificação de $a_7$ para a explosão do lugar $P_2$ de sua base de conhecimento social.

Diferentemente da base local de $a_8$, as regras geradas a partir da rede da figura 5.14 não inferem a sentença que ativaria e dispararia a transição imediatamente seguinte da rede hierarquicamente

[9]Daqui para adiante será negligenciada a demonstração do conjunto de termos de cada rede, permitindo que o leitor foque sua atenção do processo de modelagem das bases de conhecimento.

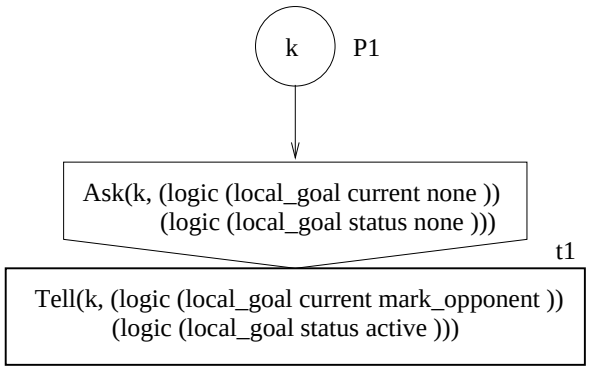


**Figura 5.14:** *Base Local do Agente 7*

superior, $t_2$, ou seja, $(ball\ control\ my\_team)$. Isto por que esta sentença deve ter origem no processo de cooperação, que uma vez recebida, é diretamente lançada a base de fatos do nível cognitivo. Neste caso, a seqüência de estados da base de conhecimento de $a_7$, segundo a dinâmica da RP, é a seguinte. Uma vez que a ficha alcance o lugar $P_2$ da rede referente a base de conhecimento social, a rede da figura 5.14 é instanciada. A seguir, ela infere a meta local `mark_opponent` e imediatamente retorna o fluxo a rede superior. Esta, por sua vez, aguarda a sentença $(ball\ control\ my\_team)$, cuja origem é o processo de cooperação, seguindo sua seqüencia normal de execução.

A substituição do lugar $P_4$ da rede referente a base social de $a_7$ dá origem a rede da figura 5.15. Nela é descrita a base local do agente no que tange a meta global rws_attack_play.

Esta rede possui dois planos possíveis, dependendo das contingências do ambiente. Considerando uma seqüência de disparo $s = [t_1\ t_2\ t_3]$, levando o agente a meta local `drive_ball`, onde deve conduzir a bola até a linha de fundo do campo adversário, ele pode eventualmente deixar escapar a bola de seu domínio. Neste caso, dispara-se a transição $t_4$, levando-o de volta a meta `domain_ball`. Uma vez de sua posse, novamente, ele pode reiniciar a condução da bola até a lateral da área, de onde ele deverá efetuar um cruzamento para área, reconhecido pela meta local `pass_to_area`.

As regras que formam a base de conhecimento local do agente 7, a partir das redes das figuras 5.14 e 5.15, são as seguintes:

```
((rule_001
  (if    (logic ( global_goal current get_ball_control ))
         (logic ( global_goal status active ))
         (logic ( local_goal status none ))
         (logic ( local_goal status none )))
  (then  (logic ( local_goal current mark_opponent ))
         (logic ( local_goal status active ))
         (message ((to Instintive) (from Cognitive)
                  (body (INFORM ((logic ( local_goal current mark_opponent )))))))
         (message ((to Instintive) (from Cognitive)
                  (body (INFORM ((logic ( local_goal status active ))))))))
(rule_002
  (if    (logic ( global_goal current rws_attack_play ))
         (logic ( global_goal status active ))
         (logic ( local_goal current none ))
         (logic ( local_goal status none )))
  (then  (logic ( local_goal current domain_ball ))
         (logic ( local_goal status active ))
         (message ((to Instintive) (from Cognitive)
                  (body (INFORM ((logic ( local_goal current domain_ball )))))))
         (message ((to Instintive) (from Cognitive)
                  (body (INFORM ((logic ( local_goal status active ))))))))
(rule_003
  (if    (logic ( global_goal current rws_attack_play ))
         (logic ( global_goal status active ))
         (logic ( local_goal current domain_ball ))
         (logic ( local_goal status active ))
         (logic ( ball proximity very_close )))
  (then  (logic ( local_goal current domain_ball ))
         (logic ( local_goal status sucess ))))
(rule_004
  (if    (logic ( global_goal current rws_attack_play ))
         (logic ( global_goal status active ))
         (logic ( local_goal current domain_ball ))
         (logic ( local_goal status sucess )))
  (then  (logic ( local_goal current drive_ball ))
         (logic ( local_goal status active ))
         (message ((to Instintive) (from Cognitive)
                  (body (INFORM ((logic ( local_goal current drive_ball )))))))
         (message ((to Instintive) (from Cognitive)
                  (body (INFORM ((logic ( local_goal status active ))))))))
```

```
(rule_005
  (if    (logic ( global_goal current rws_attack_play ))
         (logic ( global_goal status active ))
         (logic ( local_goal current drive_ball ))
         (logic ( local_goal status active ))
         (logic ( ball proximity far )))
  (then  (logic ( local_goal current domain_ball ))
         (logic ( local_goal status active ))
         (message ((to Instintive) (from Cognitive)
                 (body (INFORM ((logic ( local_goal current domain_ball )))))))
         (message ((to Instintive) (from Cognitive)
                 (body (INFORM ((logic ( local_goal status active ))))))))
(rule_006
  (if    (logic ( global_goal current rws_attack_play ))
         (logic ( global_goal current active ))
         (logic ( local_goal current drive_ball ))
         (logic ( local_goal status sucess )))
  (then  (logic ( local_goal current pass_to_area ))
         (logic ( local_goal status active ))
         (message ((to Instintive) (from Cognitive)
                 (body (INFORM ((logic ( local_goal current pass_to_area )))))))
         (message ((to Instintive) (from Cognitive)
                 (body (INFORM ((logic ( local_goal status active ))))))))
(rule_007
  (if    (logic ( global_goal current rws_attack_play ))
         (logic ( global_goal status active ))
         (logic ( local_goal current pass_to_area ))
         (logic ( local_goal status active )))
  (then  (logic ( local_goal current none ))
         (logic ( local_goal status none ))
         (message ((to Instintive) (from Cognitive)
                 (body (INFORM ((logic ( local_goal current none )))))))
         (message ((to Instintive) (from Cognitive)
                 (body (INFORM ((logic ( local_goal status none ))))))))
```

Para finalizar, tem-se a especificação da base local de $a_9$. Este agente tem o mesmo planejamento defensivo individual apresentado por $a_7$, o que não é verdade, contudo, para a meta ofensiva. Neste caso, $a_9$ tem a função de finalizar a jogada, interceptando o cruzamento de $a_7$ e concluindo a bola em gol. Esta tarefa é modelada pela RP da figura 5.16.

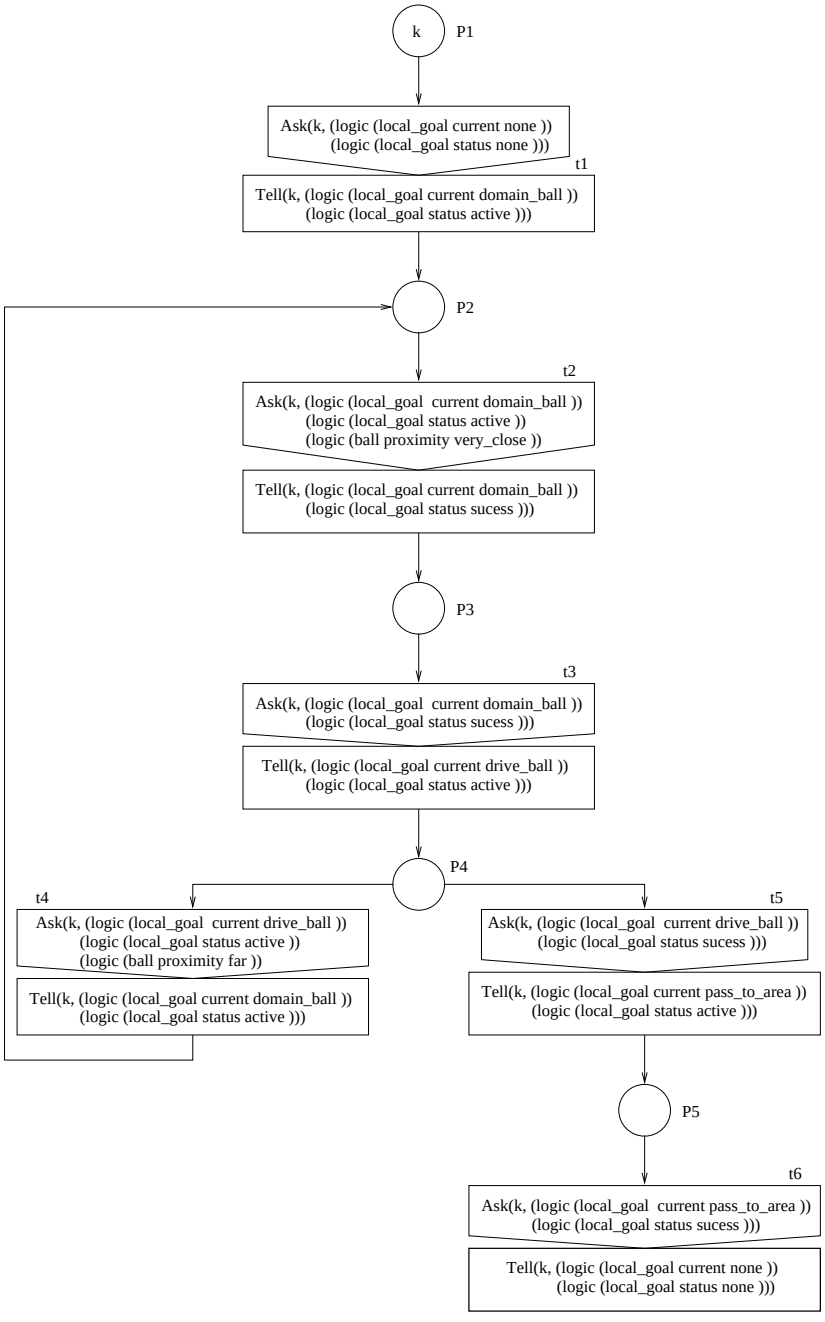


**Figura 5.15:** *Base Local do Agente 7*

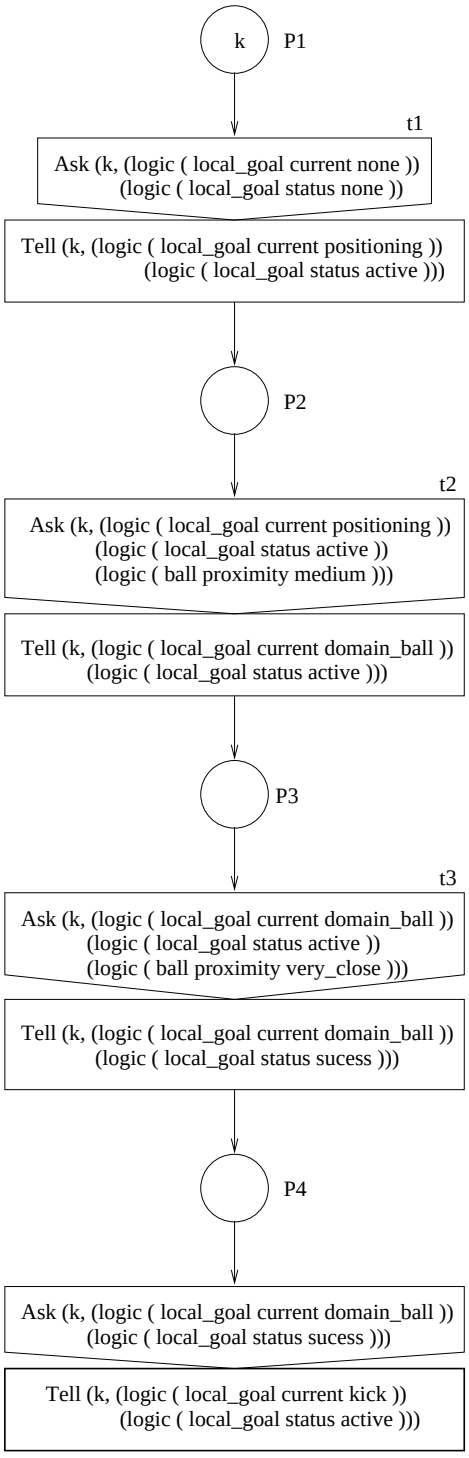


**Figura 5.16:** *Base Local do Agente 9*

Ao ser selecionada a meta global rws_attack_play $a_9$ posiciona-se a entrada da grande área, segundo a meta local `positioning`. No momento em que o agente verifica que a bola está a seu alcance, após o cruzamento, (*ball proximity medium*), ele vai em sua direção com o objetivo de dominá-la. Quando a bola está dentro da área de chute, (*ball proximity very_close*), o agente a chuta em gol.

A partir da especificação apresentada na figura 5.16, a base local de nível cognitivo de $a_9$ é formada pelo seguinte conjunto de regras:

```
((rule_001
  (if    (logic ( global_goal current get_ball_control ))
         (logic ( global_goal status active ))
         (logic ( local_goal status none ))
         (logic ( local_goal status none )))
  (then  (logic ( local_goal current mark_opponent ))
         (logic ( local_goal status active ))
         (message ((to Instintive) (from Cognitive)
                 (body (INFORM ((logic ( local_goal current mark_opponent )))))))
         (message ((to Instintive) (from Cognitive)
                 (body (INFORM ((logic ( local_goal status active ))))))))
(rule_002
  (if    (logic ( global_goal current rws_attack_play ))
         (logic ( global_goal status active ))
         (logic ( local_goal status none ))
         (logic ( local_goal status none )))
  (then  (logic ( local_goal current positioning ))
         (logic ( local_goal status active ))
         (message ((to Instintive) (from Cognitive)
                 (body (INFORM ((logic ( local_goal current positioning )))))))
         (message ((to Instintive) (from Cognitive)
                 (body (INFORM ((logic ( local_goal status active ))))))))
```

```
(rule_003
  (if    (logic ( global_goal current rws_attack_play ))
         (logic ( global_goal status active ))
         (logic ( local_goal status positioning ))
         (logic ( local_goal status active ))
         (logic ( ball proximity medium )))
  (then  (logic ( local_goal current domain_ball ))
         (logic ( local_goal status active ))
         (message ((to Instintive) (from Cognitive)
                 (body (INFORM ((logic ( local_goal current domain_ball )))))))
         (message ((to Instintive) (from Cognitive)
                 (body (INFORM ((logic ( local_goal status active ))))))))
(rule_004
  (if    (logic ( global_goal current rws_attack_play ))
         (logic ( global_goal status active ))
         (logic ( local_goal status domain_ball ))
         (logic ( local_goal status active ))
         (logic ( ball proximity very_close )))
  (then  (logic ( local_goal current domain_ball ))
         (logic ( local_goal status sucess ))))
(rule_005
  (if    (logic ( global_goal current rws_attack_play ))
         (logic ( global_goal status active ))
         (logic ( local_goal status domain_ball ))
         (logic ( local_goal status sucess )))
  (then  (logic ( local_goal current kick ))
         (logic ( local_goal status active ))
         (message ((to Instintive) (from Cognitive)
                 (body (INFORM ((logic ( local_goal current kick )))))))
         (message ((to Instintive) (from Cognitive)
                 (body (INFORM ((logic ( local_goal status active ))))))))
```

Definidas as bases de conhecimento sociais e locais de $a_7$, $a_8$ e $a_9$, que referem-se ao nível cognitivo, o próximo passo é a construção das bases de conhecimento dos respectivos níveis instintivos.

Recordando a arquitetura do agente autônomo concorrente, o nível instintivo é responsável por selecionar o comportamento reativo do agente, segundo a sua meta local. Além disso, ele deve prover o nível cognitivo com informações simbólicas, que referem-se à classificação dos estados do ambiente. Portanto, a base de regras do nível instintivo contém dois tipos de regras. O primeiro tipo infere informações simbólicas a partir das informações sensorias, classificando o estado atual do ambiente. Já o segundo tipo, define uma seqüência de ações a serem executadas no ambiente,

segundo a meta local do agente. Sendo assim, indica-se a utilização da abordagem por RP apenas para o segundo caso. De fato, não há nada que impeça a utilização também para o primeiro grupo, mas dada a natureza do ambiente, e o tipo de regra a ser modelada, não faz sentido especificá-las por intermédio desta abordagem, uma vez que tratam-se de regras isoladas, sem qualquer tipo de encadeamento ou relação dinâmica.

A escolha dos comportamentos para o nível reativo deve estar de acordo com o papel que o agente está executando. Uma vez que a especificação social aponta dois esquemas sociais, tem-se dois conjuntos de papeis: $P_1$ de $es_1$ e $P_2$ de $es_2$. $P_1$ inclui um papel de defesa e $P_2$ um papel de ataque. Para poder executar $P_1$ um agente possui os seguintes comportamentos disponíveis: [10] *Watch-Ball*, *Move-To-Ball*, *Move-to-Direction*, ou seja, todos os comportamentos que não incluem a manipulação da bola. Para executar $P_2$, além dos comportamentos de $P_1$, um agente pode utilizar os seguintes comportamentos: *Pass-Ball*, *Drive-Ball-Fwd* e *Kick-To-Goal*.

As RP que modelam a base de conhecimento instintiva surgem a partir da substituição de lugares da redes que modelam a base local do nível cognitivo do agente. Mais especificamente, substituem lugares para os quais a base de conhecimento infere regras em que há o envio de mensagens para o nível instintivo, como os lugares $P_2$ e $P_4$ para a rede da figura 5.12, ou o lugar $P_2$ para a rede da figura 5.13. [11] Sendo assim, os lugares $P_2$ e $P_4$ da rede da figura 5.12 geram, respectivamente os modelos apresentados pelas figuras 5.17 e 5.18. Já o lugar $P_2$ da rede da figura 5.13 gera a rede da figura 5.19.

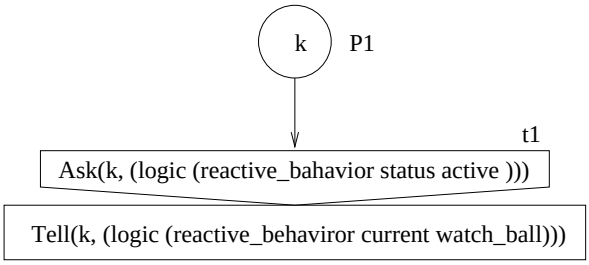


**Figura 5.17:** *Base Instintiva referente a meta local* `mark_ball` *do agente 8*

Os modelos que referem-se as metas locais `mark_ball` e `take_ball` são redes simples com apenas um lugar e uma transição. Elas determinam o comportamento reativo mais adequado a meta local selecionada pelo nível instintivo.

Já a rede que modela a base instintiva para a meta local `drive_ball` apresenta duas possibilidades de ações, permitindo um trabalho de correção da condução de bola, uma vez que o agente a

[10] c.f. item 5.3

[11] Será apresentado apenas a construção de algumas regras do nível instintivo de $a_8$, uma vez que o objetivo aqui é descrever o processo de construção da base de conhecimento de um agente.

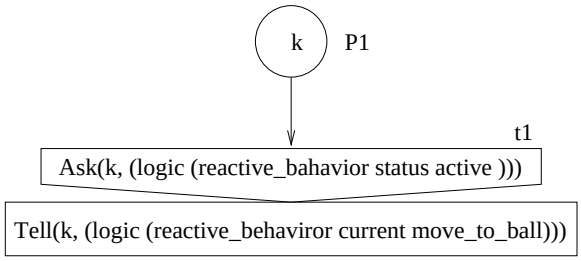


**Figura 5.18:** *Base Instintiva referente a meta local* `take_ball` *do agente 8*

deixe escapar, em curtas distâncias.

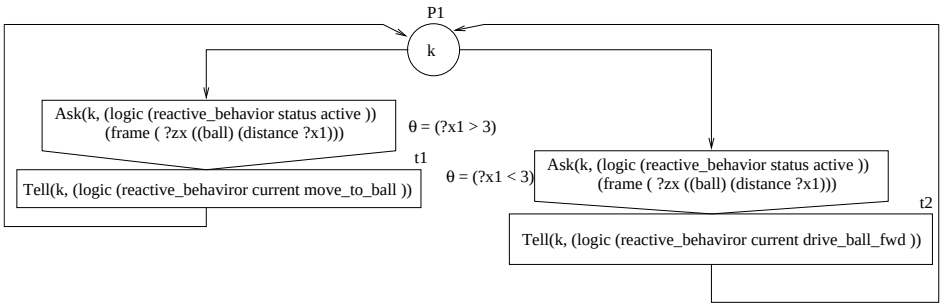


**Figura 5.19:** *Base Instintiva referente a meta local* `drive_ball` *do agente 8*

Assim como para a base de conhecimento local do nível cognitivo, a base de conhecimento do nível instintivo é formada pelo somatório das regras geradas a partir das diferentes instâncias hierárquicas inferiores a base local. Sendo assim, as três redes que modelam o comportamento instintivo de $a_8$ geram o seguinte grupo de regras:

```
((rule__001
  (if   (logic ( local_goal current mark_ball ))
        (logic ( local_goal status active ))
        (logic ( reactive_behavior status active )))
  (then (logic ( reactive_behavior current watch_ball ))
        (message ((to Cognitive) (from Instintive)
          (body (INFORM ((logic ( reactive_behavior current watch_ball )))))))
         (message ((to Reactive) (from Instintive)
          (body (INFORM ((logic ( new_controller Watch_Ball ))))))))
```

```
(rule__002
  (if   (logic ( local_goal current take_ball ))
        (logic ( local_goal status active ))
        (logic ( reactive_behavior status active )))
  (then (logic ( reactive_behavior current move_to_ball ))
        (message ((to Cognitive) (from Instintive)
          (body (INFORM ((logic ( reactive_behavior current move_to_ball )))))))
        (message ((to Reactive) (from Instintive)
          (body (INFORM ((logic ( new_controller Move_To_Ball ))))))))
(rule__003
  (if   (logic ( local_goal current drive_ball ))
        (logic ( local_goal status active ))
        (logic ( reactive_behavior status active )))
  (then (logic ( reactive_behavior current drive_ball_fwd ))
        (message ((to Cognitive) (from Instintive)
          (body (INFORM ((logic ( reactive_behavior current drive_ball_fwd )))))))
        (message ((to Reactive) (from Instintive)
          (body (INFORM ((logic ( new_controller Drive_Ball_Fwd ))))))))
```

O nível reativo do agente autônomo concorrente é formado por um conjunto de controladores nebulosos, onde cada controlador descreve um comportamento reativo do agente no ambiente. No Soccerserver, cada comportamento refere-se às habilidades básicas para a prática do futebol, como passar, chutar, correr, conduzir a bola, etc. Desta forma, a abordagem de especificação de conhecimento, a partir da especificação em RP, é utilizada até o nível instintivo. Dada a particularidade do nível reativo, outros métodos devem ser empregados para o seu desenvolvimento.

### Análise das RP

Após a modelagem das RP nos diversos níveis de abstração do sistema, estas estão sujeitas a uma análise de suas propriedades, visando um certo nível de correção sintática.

O processo de análise dá-se em cima do modelo adjacente de cada RP. Este modelo adjancente corresponde a estrutura da RP, abstraindo sua estrutura de dados, e considerando um distribuição de fichas binárias correspodente a marcação inicial da rede, segundo o modelo de análise apresentado em Sibertin-Blan (1985).

Deste modo, as RP adjacentes que implementam as jogadas descritas no item anterior foram analisadas com o auxílio da ferramenta ARP (ARP, 1989). No quadro abaixo, apresenta-se a RP adjacente referente ao nível social do SMA (figura 5.10).

```
Net Nivel_Social;
{
Esta RP corresponde ao modelo adjacente à RP que especifica
o nível social do SMA.
}

Nodes
t1, t2, t3, t4, t5, t6 : transition;
P1                     : place(1);
P2, P3                 : place;

Structure
t1 : (P1), (P2);
t2 : (P2), (P3);
t3 : (P3), (P2);
t4 : (P3), (P3);
t5 : (P3), (P2);
t6 : (P3), (P1);
endNet.
```

Dada as características do domínio, o que se busca é uma rede reiniciável, viva e limitada. Neste caso, uma rede viva implica que todos os estados descritos pela rede são alcançáveis a partir de qualquer outro estado da mesma, e uma rede limitada significa que os recursos disponíveis pelo ambiente são suficientes para levar o sistema de seu estado atual a qualquer outro estado. De maneira geral, estas são propriedades desejáveis a qualquer sistema modelado por uma RP. Já a propriedade que garante da reinicialização da rede é desejada na medida em que quem determina o final do ciclo de execução do SMA é o ambiente e não o próprio SMA.

As redes que instanciam o nível social nas bases sociais dos agentes possuem a mesma estrutura daquela, e portanto apresentam as mesmas propriedades.

Com relação as redes adjacentes de nível hierárquico inferior, como por exemplo a rede da figura 5.12, a propriedade de reiniciabilidade já não é necessária, na medida em que o fluxo de controle da rede deverá ser devolvido a rede de nível superior. Sendo assim, apenas a RP de maior nível hierárquico é que deve apresentar esta propriedade.

Após a simulação de todas as redes na ferramenta ARP, foram observadas as propriedades necessárias que garantam a sua correção com relação aos requisitos de cada um dos níveis de abstração tratados. Dentre os motivos observados para a rápida convergência das redes aos modelos ideias em

relação as suas propriedades estão a simplicidade observada nos modelos adjacentes, que deve-se essencialmente a possibilidade de hierarquização entre as redes.

**Refinamento do Conhecimento**

Este item tem o objetivo de demonstrar como dá-se o processo de incorporação de novo conhecimento àquele já exsitente no SMA em desenvolvimento. Para isso será apresentada a especificação de uma nova jogada dentro do Soccerserver.

A RP da figura 5.10 descreve o conjunto de planos manipulados pelo SMA por intermédio de um conjunto de esquemas sociais. Entretanto, a jogada apresentada refere-se apenas à seqüência de disparo $s = [t_1\ t_2\ t_5\ t_6]$ da RP, onde $es_x = es_4$. Para a implementação de $es_5$ tem-se o mesmo conjunto de RP apresentadas para a $es_4$, entretanto com os agentes $a_{10}$, e $a_{11}$ executando os mesmos papéis de $a_8$ e $a_7$, respectivamente. O agente $a_9$ mantém o mesmo comportamento descrito, modificando apenas seu comportamento reativo, que, neste caso, é independente da especificação dos níveis instintivo e cognitivo. A conseqüência desta simetria de especificação é que as redes que modelam $a_8$ são replicadas para o agente $a_{10}$, apenas modificando as sentenças que designam a meta tratada. Deste modo $(logic\ (global\_goal\ current\ rws\_attack\_play))$ deve ser substituído por $(logic\ (global\_goal\ current\ lws\_attack\_play))$. A figura 5.20 apresenta a RP que modela a base social de $a_{10}$.

A RP da base local de defesa de $a_8$, por sua vez, é igual a de $a_{10}$. Entretanto, a base local de ataque de $a_{10}$ deve considerar um passe para $a_{11}$, e não para $a_7$ como é o caso da base local de ataque de $a_8$, apresentada na figura 5.13. A base local de ataque de $a_{10}$ é apresentada na figura 5.21.

Observa-se que toda a incorporação de conhecimento ao SMA dá-se pela inclusão de novos elementos a RP existentes (lugares e transições) e/ou pela hierarquização de novas RP. A incorporação de conhecimento pode ser compreendida pela modificação ou refinamento de conhecimento considerando os agentes já existentes, ou pela especificação de novos agentes, com suas respectivas RP.

### 5.5.3 Análise da Aplicação da Abordagem

A análise dos resultados obtidos a partir da utilização da abordagem proposta nesta tese, independente da aplicação utilizada, foi o ponto mais sensível deste trabalho. Isto por que, foi difícil encontrar uma forma de isolar o problema dentro de qualquer aplicação, o que permitiria uma análise específica da questão.

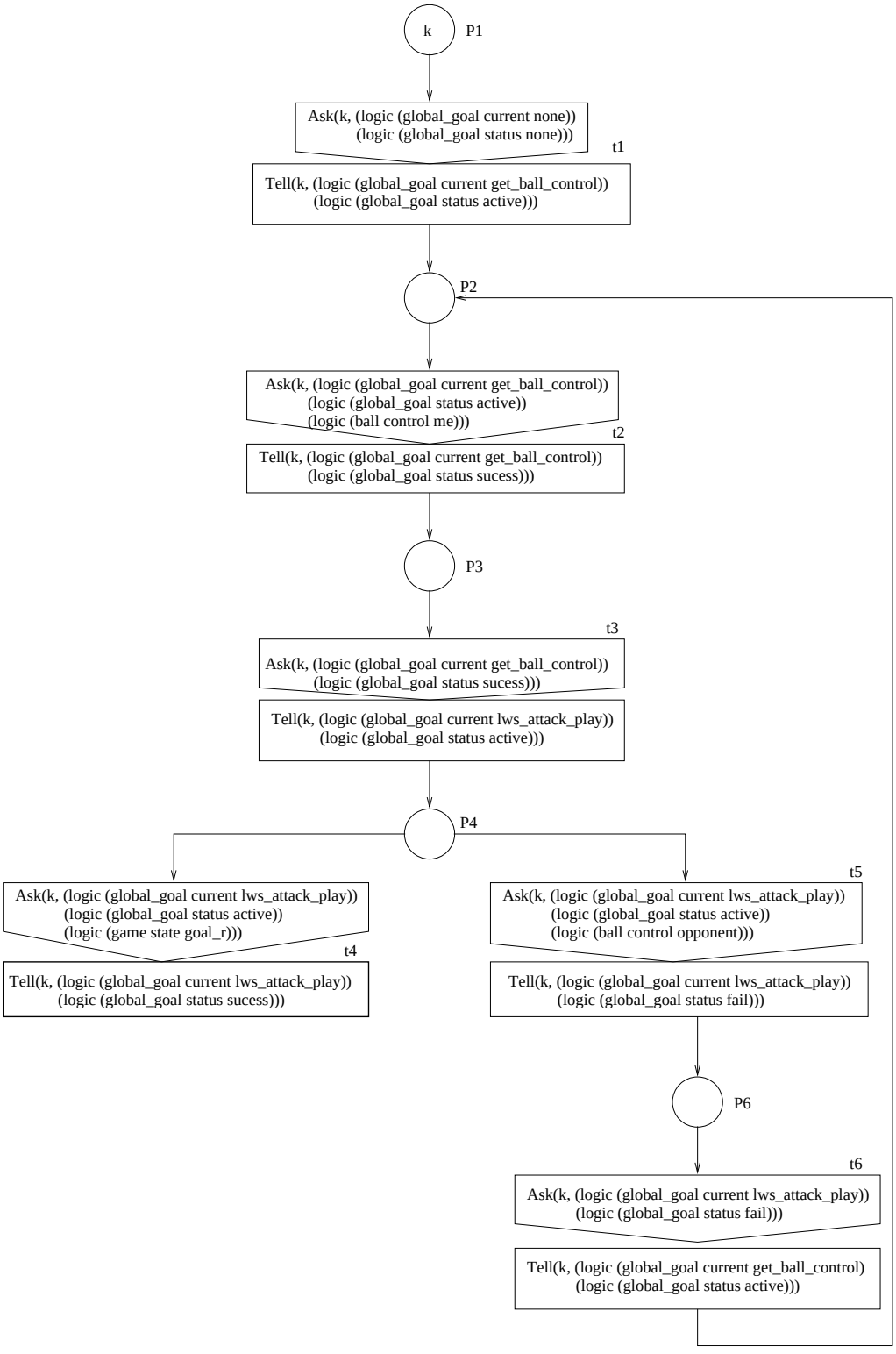


**Figura 5.20:** *Base Social do agente 10*

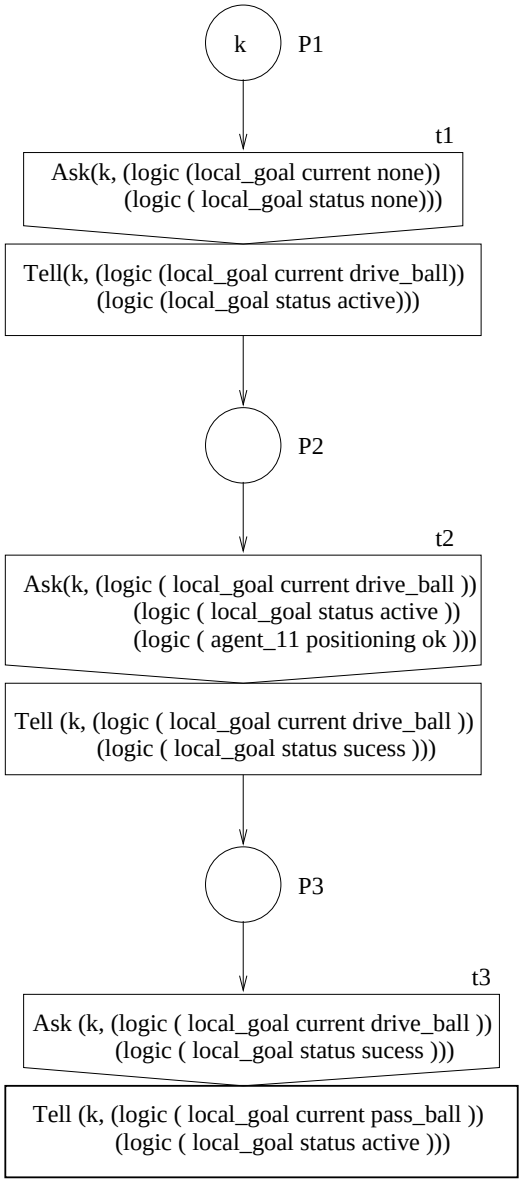


**Figura 5.21:** *Base local do agente 10*

O método de avaliação que se mostraria mais evidente consideraria a execução de múltiplas simulações, onde determinar-se-ia a porcentagem de vezes em que um gol foi marcado, dado o cenário especificado. Entretanto, por este método, se avalia o desempenho do agente como um todo, incluindo sua arquitetura, seu nível de conhecimento e de implementação, cujo processo de aquisição de conhecimento é apenas um dos seus aspectos.

Por outro lado, a utilização de outras abordagens se mostraria um contra senso, uma vez que nos capítulos anteriores provou-se que as metodologias existentes não se adaptavam ao problema.

O problema está no fato de que a análise de uma metodologia e/ou abordagens tem um caráter mais qualitativo do que quantitativo, quando dentro da ciências exatas se está habituado justamente com este último tipo. Desta forma buscou-se uma análise qualitativa do método, ressaltando suas principais características dentro do domínio em questão.

Em uma primeira tentativa de implementação deste cenário (da Costa e Bittencourt, 2002), o processo de construção das regras dos agentes era puramente empírico e intuitivo. A medida que aumentava-se a base de regras, aumentava-se a dificuldade de incluir novas regras, uma vez que estas, além de serem verificadas quanto a sua correção semântica, deveriam ainda ser compatíveis com as regras já existentes. A dificuldade advinha da complexidade em gerenciar as variáveis envolvidas, sem que se gerasse algum tipo de inconsistência entre o conhecimento já codificado e aquele que deveria ser inserido.

A vantagem mais evidente, a partir da utilização de RP, é a modularização do processo de aquisição de conhecimento em torno dos aspectos necessários à articulação de ações no ambiente. As RP permitem uma visualização explícita entre as estruturas de conhecimento, permitindo que, uma vez estabelecida a correta relação entre estas estruturas, dedicar-se ao desenvolvimento de cada uma delas. A conseqüência imediata é a aceleração do processo, cuja garantia de correção, agora, não dependia apenas da simulação do sistema, mas também, da simulação do modelo, no caso, a RP.

Esta modularização permite ainda a divisão do processo de estruturação do conhecimento em grupos de trabalho, onde cada grupo fica encarregado de um módulo. No caso do agente autônomo concorrente, poderia-se atribuir um grupo de desenvolvimento específico para o nível cognitivo, um para o instintivo e outro para o reativo. As relações entre os grupos dar-se-ia pela correta hierarquização entre as RP desenvolvidas.

## 5.6 Considerações Finais

Neste capítulo, buscou-se descrever um processo de aplicação da abordagem de especificação de conhecimento para SMA Cognitivos em um simulador de partidas de futebol entre agentes computacionais.

O simulador utilizado, Soccerserver, faz parte de uma das ligas que compõe a Robocup. A Robocup é uma iniciativa de escala mundial que estabelece um problema padrão a ser resolvido, visando o desenvolvimento de tecnologias baseada em IA. Este problema é justamente a construção de equipes de robôs para a prática do futebol.

Dentro da proposta estabelecida pela Robocup, o Soccerserver tem a finalidade de abranger os assuntos relacionados ao desenvolvimento de softwares baseados em IA, especialmente em IA Distribuída. Torna-se, portanto, uma excelente plataforma para o desenvolvimento de SMA Cognitivos, dada a dimensão e complexidade da tarefa.

Uma vez definido o domínio, fez-se necessário a definição de uma arquitetura que suportasse a construção da equipe. Para tanto, foi escolhido o agente autônomo concorrente, cuja arquitetura baseia-se na divisão da complexidade de um domínio em níveis de decisão. O nível reativo implementa o conjunto de comportamentos do agente no ambiente. O nível instintivo classifica os estados do ambiente e seleciona o comportamento do agente segundo sua meta local. O nível cognitivo estabelece o planejamento do agente segundo seu papel na equipe.

Para implementar o agente autônomo concorrente foi utilizada a Expert-Coop++, um arcabouço para o desenvolvimento de SMA Cognitivos sob restrições de tempo real do tipo melhor esforço. Sua escolha é justifica-se uma vez que sua versão utilizada foi devidamente construída para atender os requisitos do agente autônomo concorrente. Outrossim, a Expert-Coop++ pode ser empregada também no desenvolvimento de outras arquiteturas.

Na medida em que se busca uma comprovação da eficácia da abordagem, foi definido um cenário simplificado do domínio para implementação. Trata-se de uma jogada pré-planejada envolvendo três agentes, cujo objetivo final é a marcação de um gol. Tal cenário mostrou-se suficiente para a demonstração das principais características do emprego da abordagem em um cenário multiagente.

Se comparado aos métodos empíricos utilizados para a implementação do mesmo cenário, a abordagem mostrou-se eficaz. Sua principal contribuição é a aceleração da aquisição de conhecimento do SMA em desenvolvimento. A abordagem permite a modularização do processo, de modo a organizar a participação de mais desenvolvedores na implementação do sistema.

# Capítulo 6

# Conclusão

O problema tratado nesta tese tem como tema fundamental a aquisição de conhecimento em Sistemas Multiagentes (SMA) Cognitivos, ou seja, a definição de métodos e mecanismos que permitam a especificação em nível de conhecimento de cada agente de uma sociedade, de modo que ele possa atuar segundo os objetivos do sistema e seus próprios objetivos.

A partir da inclusão do agente em um ambiente com outros agentes que também possuem suas próprias metas, planos e habilidades, o problema da aquisição de conhecimento torna-se um gargalo ainda mais evidente. Neste novo cenário, o conhecimento do agente deve considerar não apenas suas próprias atividades, mas deve considerar também o fato de que outros agentes podem alterar o estado do ambiente. A descrição do estado global do sistema é composta pela combinação dos estados parciais, e estratégias de resolução de problemas devem considerar diferentes níveis de abstração, desde mecanismos de coordenação global até a integração das atividades motoras individuais.

Dada a descrição do problema e o conjunto de restrições estabelecidas pela literatura, partiu-se para a definição de uma abordagem para especificação de conhecimento aplicada a SMA Cognitivos baseada em Redes de Petri (RP) Hierárquicas. Em outras palavras, trata-se de um conjunto de etapas e procedimentos voltados para especificação em nível de conhecimento de um agente, considerando sua participação em uma sociedade de agentes, utilizando RP Hierárquicas como linguagem de especificação, desde o nível social até o nível individual.

Esta abordagem está alicerçada sobre uma arquitetura genérica de cognição. Entre as hipóteses desta arquitetura está a idéia de que a inteligência possui necessariamente uma contrapartida social, ou seja, um ser só pode ser considerado inteligente se inserido dentro de uma sociedade de entidades similares.

Esta arquitetura divide a inteligência em aspectos, segundo um modelo das funções executivas do cérebro. Assim, toda e qualquer atividade cognitiva envolve os seguintes aspectos: *planejamento*, *coordenação de atividades motoras* e *atividades motoras*. Em termos de uma arquitetura computacional compatível, cada aspecto deve estar associado a um ou mais Sistemas Baseado em Conhecimento (SBC), de acordo com o nível de abstração do sistema. O nível de planejamento tem seu contexto definido a partir da inserção do agente em uma sociedade de agentes, ou seja, seu planejamento de ações individuais depende diretamente do planejamento global estabelecido pela sociedade e do papel do agente dentro desta sociedade.

A partir desta arquitetura genérica, o SMA é especificado segundo duas visões do sistema: a visão externa, em que o sistema é estruturado em um nível social, que corresponde a organização do SMA, e um nível individual, que corresponde a especificação e implementação dos agentes segundo a arquitetura genérica de cognição; e uma visão interna, em que são aplicados os modelos de organização de SMA para o nível social, e os princípios de aquisição de conhecimento para o nível individual segundo uma linguagem de especificação baseada em um modelo específico de RP Hierárquicas. Esta linguagem tem a propriedade de integrar os aspectos sociais à implementação do agente em seu nível individual.

Além de integrar o nível social e individual, este modelo de RP permite também:

- uma especificação em nível de conhecimento, adequando seu conjunto de termos a qualquer formalismo de representação de conceitos sociais e a qualquer formalismo de representação de conhecimento, estabelecendo uma relação formal entre o nível social e individual;
- permitir a verificação estrutural do conhecimento modelado, pela utilização das ferramentas de análise de RP;
- estruturar o sistema em múltiplos níveis de abstração, definido segundo a complexidade do conhecimento do sistema, por intermédio dos mecanismos de hierarquização;
- definir um mecanismo de representação para a coordenação social do sistema, por intermédio de um planejamento global de ações;
- um mapeamento automático entre os diferentes formalismos de representação de conhecimento utilizados, na base de conhecimento dos agentes da sociedade.

Ao se comparar a abordagem proposta com uma síntese das metodologias para o desenvolvimento de SMA é possível constatar uma série de avanços no estado da arte. De fato, a literatura

sobre metodologias para a construção de SMA é bem ampla. Contudo, as metodologias existentes não abordam todas as possibilidades de soluções que a teoria de SMA propicia.

A maioria das metodologias derivam de ferramentas utilizadas em Engenharia de Software para abordagens orientadas a objetos. Esta adequação é possível a partir da extensão do conceito de objeto, de modo a abranger propriedades fundamentais de agentes, como autonomia e interação com um subconjunto de agentes. Contudo, esta adequação é feita de modo específico para cada metodologia, o que gera a ausência de um padrão para a definição de termos específicos para um SMA, como papéis, tarefas, relações entre papéis, etc. Em alguns casos, este termos são definidos de maneira implícita, utilizando outros conceitos derivados da orientação a objeto.

Um aspecto que pode ser problemático na construção de um SMA é a especificação das interações entre os agentes. As metodologias tratam destas especificações no nível de comunicação do agente, de modo que toda a troca de mensagem deve ser especificada de acordo com o protocolo utilizado. Dependendo do número de agentes do sistema, e do número de interações, o nível de comunicação deve ser especificado por diagramas de seqüências extremamente complexos e de difícil implementação.

Quanto à arquitetura interna dos agentes, as metodologias para SMA utilizam uma das seguintes abordagens: ou consideram uma arquitetura interna genérica, definida apenas pelas propriedades dos agentes que devem ser observadas, que geram diretrizes para a implementação dos agentes; ou partem para uma especificação na arquitetura BDI, cujo resultado final é uma implementação, ou parte da implementação do agente, segundo uma linguagem baseada em BDI.

Primeiramente, a abordagem proposta adequa-se ao paradigma de orientação a agentes de forma natural, permitindo a representação explícita dos mecanismos e conceitos que geram as propriedades desejadas neste tipo de abordagem.

Por meio de um mecanismo de coordenação baseado em planejamento global, a abordagem simplifica a especificação das interações entre os agentes, uma vez que ela é definida por planos. Os agentes coordenam suas ações pela definição de planos comuns, em que cada um possui atribuições específicas segundo seus papéis no ambiente. Não se trata, evidentemente, da eliminação de interações entre agentes, mas sim, de sua otimização em torno de mecanismos que independem da especificação explícita deste aspecto da comunicação. O resultado final é uma especificação mais limpa e simplificada dos meios de interação entre os agentes.

Ainda em comparação com as metodologias para a construção de SMA, a abordagem proposta diferencia-se pela consideração de uma arquitetura interna genérica do agente baseada em

conhecimento. Assim, permite-se uma alternativa com relação à arquitetura BDI, quando parte-se para uma solução que busca um resultado mais próximo da implementação final, ou uma alternativa à consideração genérica de arquitetura, mas que nem por isso, distancia o resultado obtido da implementação final do agente.

Se comparada as metodologias específicas para a construção de SBC, a abordagem proposta se diferencia pela utilização de diferentes mecanismos de representação de conhecimento, pela estruturação do conhecimento em diferentes níveis de abstração, e pela utilização de uma linguagem de especificação que permite uma melhor aproximação entre engenheiro de conhecimento e especialista.

Devido às inúmeras possibilidades de aplicação de SBC's, existe um grande número de ferramentas utilizadas para sua implementação. Estas ferramentas podem ser divididas segundo a abordagem empregada na construção do SBC: baseada em transferência ou baseada em modelos. A primeira abordagem considera que todo o trabalho de construção de um SBC dá-se pela transferência do conhecimento existente a respeito de um domínio para uma base de conhecimento. As principais ferramentas baseadas neste modelo são o EMYCIN/TEIRESIAS e o ONCOCIN/OPAL. Em uma abordagem por modelos o objetivo é a construção de modelos computacionais dos métodos de resolução de problemas utilizados no domínio. Entre as principais ferramentas deste tipo estão o KADS/CommonKADS e o Protégé.

Contudo, estas ferramentas apresentam algumas limitações se inseridas em uma abordagem baseada em conhecimento para a construção de SMA. As principais são a ausência de uma linguagem de especificação capaz de integrar os aspectos sociais ao conhecimento interno do agente, a impossibilidade de tratar com múltiplos formalismos de representação de conhecimento, e a incapacidade de estruturar o conhecimento em diferentes níveis de abstração.

Ao comparar a abordagem proposta com outras metodologias que também utilizam RP para o desenvolvimento de SMA, em qualquer dos seus níveis, também observa-se uma série de evoluções.

Em SMA, as RP descrevem especificações sociais por intermédio das relações de sincronizações entre os agentes de um ambiente. Estes pontos de sincronizações são ações que os agentes devem realizar em conjunto para que as tarefas do ambiente sejam executadas. Assim, estes modelos geram uma especificação global do sistema a partir do somatório das especificações individuais pautadas em pontos de sincronização entre os agentes. Como conseqüência deste tipo de abordagem, os agentes possuem baixa flexibilidade de comportamento no ambiente.

Quanto às especificações individuais de agentes, ou de SBC's, os modelos que utilizam RP o fazem segundo um, ou mais de um dos critérios a seguir:

- como método de representação de conhecimento, capaz de representar apenas formalismos baseado em lógica, seja proposicional, de primeira ordem, ou nebulosa;
- como mecanismo de verificação de anomalias em estruturas baseadas em regras;
- incorporado a metodologias e/ou abordagens específicas de desenvolvimento de soluções baseadas em sistemas inteligentes, permitindo uma análise semântica a partir dos requisitos do sistema.

No nível social, a metodologia abordagem representa o conhecimento social pelos mecanismos de coordenação utilizados, ao invés do somatório de conhecimentos individuais dos modelos existentes. No nível individual, a abordagem proposta prevê a estruturação do conhecimento, quando necessário, por intermédio de hierarquizações, e permite a utilização de diferentes formalismos de representação de conhecimento. Enquanto isso, os modelos existentes prevêem limitados mecanismos de hierarquização, e são limitados a representações em lógica.

Com o objetivo de demonstrar o potencial da abordagem proposta, foi implementada uma equipe de agentes virtuais para a execução de uma jogada ensaiada no ambiente Soccerserver. O Soccerserver é um simulador de partidas de futebol envolvendo agentes virtuais.

Comparado a outras tentativas de se implementar o mesmo cenário, a aplicação da abordagem gerou a aceleração do processo desenvolvimento, por meio da modularização do conhecimento envolvido na jogada. Além disso, a abordagem provê meios para a integração de novo conhecimento, sem gerar inconsistência ou redundância com o conhecimento já adquirido.

Finalizando, em linhas gerais, as contribuições desta tese são as seguintes:

- uma abordagem para a construção de SMA Cognitivos, em que trabalha-se no nível de conhecimento do sistema, segundo uma abordagem orientada a agentes, baseada em um modelo de cognição genérica adequada à hipótese de inteligência social;
- um modelo de coordenação que permite simplificar a especificação das interações entre os agentes;
- uma abordagem de arquitetura interna de agente baseada em conhecimento, segundo um modelo de cognição genérica, estruturado em duas visões distintas do sistema;
- uma linguagem de especificação única para todos os níveis de abstração do sistema, baseada em RP Hierárquicas;

- adequação desta linguagem a diferentes formalismos de representação de conhecimento, seja social ou individual;
- aplicação da abordagem em uma equipe de futebol de agentes virtuais em um cenário simplificado do ambiente de simulação Soccerserver.

## 6.1 Trabalhos Futuros

Dentro de um conceito de metodologia completa para o desenvolvimento automático de SMA cognitivos, a partir de uma especificação social e do conhecimento das funções de cada agente do ambiente, é possível vislumbrar uma série de outros tópicos que devem ser explorados no sentido de maximizar as potencialidades do modelo proposto, e principalmente, superar as limitações do modelo atual:

- Inclusão de conceitos específicos sobre *Planejamento* nos níveis que operam com este tipo de conhecimento. Neste caso, a hipótese seria a anexação de condições e pós-condições determinísticas, quando possível, a um subconjunto de ações do agente.
- Inclusão no nível de especificação social do modelo de uma discussão sobre outras formas de coordenação multiagente, além de planos.
- Utilização de um processo de engenharia reversa para a análise de uma base de conhecimento. Neste caso, geraria-se um conjunto de RP relacionadas hierarquicamente a partir de um conjunto de bases de conhecimento com o objetivo de analisá-las estruturalmente quanto à presença de anomalias.
- Análise do estado-da-arte sobre RP e incorporação destas novas abordagens á linguagem de especificação.
- Análise e implementação do nível de interpretação das RP que garantiriam relação direta bilateral entre o nível de conhecimento e de implementação do sistema, incluindo a arquitetura subjacente necessária. Neste caso, uma vez especificado o conjunto de ações disponíveis por cada agente e seu papel no ambiente, as bases de conhecimento individuais seriam deduzidas automaticamente a partir do nível de especificação social.
- Inclusão de um mecanismo que garanta correção entre o conjunto de termos especificados para as Redes de Petri e os métodos de representação de conhecimento utilizados pelos agentes.

- Inclusão de um mecanismo de verificação dos requisitos do sistema, dentro de uma análise semântica.
- Implementação de uma ferramenta computacional baseada na abordagem.

# Apêndice A

# Soccerserver

Este apêndice tem por objetivo descrever os mecanismos que possibilitam a troca de informações entre os clientes desenvolvidos e o simulador Soccerserver.

## A.1 Sensores

Dentro do Soccer Server um agente possui três sensores. Um sensor auditivo que detecta as mensagens difundidas pelos demais agentes, pelo árbrito e pelo treinador. Um sensor visual fornece a informação relativa as distâncias e direções de objetos dentro do campo de visão do agente. Um terceiro sensor, chamado sensor corporal, informa o estado físico atual do agente dentro da partida [1].

Mensagens auditivas são enviadas quando um cliente faz a chamada de um comando `say`, da mesma forma que o árbrito. Todas as mensagens são recebidas imediatamente, e apresentam este formato:

(*hear Time Sender "Message"*)

*Time* indica o tempo de simulação do envio.

*Sender* é a direção relativa do agente que enviou a mensagem. Caso contrário este campo pode assumir os seguintes valores:

- *self*: quando trata-se de uma mensagem enviada pelo próprio agente.

[1] Nos experimentos realizados para este trabalho de tese foram desconsideradas as informações do sensor corporal. Deste modo, para um melhor entendimento deste sensor sugere-se a leitura do manual (Chen et al., 2001)

- *referee*: quando trata-se de uma mensagem enviada pelo árbrito.

*Message* é a mensagem enviada, cujo tamanho máximo é um parâmetro de simulação.

O sensor visual transmite ao cliente os objetos contidos dentro de sua faixa de visão, em intervalos regulares de 150ms. Esta informação chega do servidor no seguinte formato:

*(ObjName Distance Direction DistChng DirChng BodyDir HeadDir)*
*ObjName* ::= *(player Teamname UniformNumber)*
| *(goal [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:

$$p_{rx} = p_{xt} - p_{xo} \quad \text{(A.1)}$$

$$p_{ry} = p_{yt} - p_{yo} \quad \text{(A.2)}$$

$$v_{rx} = v_{xt} - v_{xo} \quad \text{(A.3)}$$

$$v_{ry} = v_{yt} - v_{yo} \quad \text{(A.4)}$$

$$Distance = \sqrt{p_{rx}^2 + p_{ry}^2} \quad \text{(A.5)}$$

$$Direction = \arctan(p_{ry}/p_{rx}) - a_o \quad \text{(A.6)}$$

$$e_{rx} = p_{rx}/Distance \quad \text{(A.7)}$$

$$e_{ry} = p_{ry}/Distance \quad \text{(A.8)}$$

$$DistChng = (v_{rx} * e_{rx}) + (v_{ry} * e_{ry}) \quad \text{(A.9)}$$

$$DirChng = [(-(v_{rx} * e_{ry}) + (v_{ry} * e_{rx}))/Distance] * (180/\pi) \quad \text{(A.10)}$$

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 a direção absoluta para a qual o observador está 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. Os campos *BodyDir* e *HeadDir* são somente inclusos na informação caso o objeto observado seja um jogador, e são as direções do corpo e cabeça do jogador observado em relação as direções do corpo e cabeça do jogador que está observando. Assim, se os jogadores têm seus corpos apontados na mesma direção, então *BodyDir* seria 0. O mesmo acontece para *HeadDir*.

O objeto (*goal r*) é interpretado como o ponto central sob o gol posicionado do lado direito do campo. (*flag c*) é um flag virtual que situa-se no centro do campo. (*flag l b*) é o flag que se localiza no canto inferior esquerdo do campo. (*flag p l b*) é um flag virtual localizado no canto inferior direito da grande área posicionada do lado esquerdo do campo. (*flag g l b*) é um flag virtual que marca o poste direito localizado no gol do lado esquerdo do campo. Os tipos restantes de flags estão todos localizados a 5 metros fora do campo de jogo.

No caso de (*line ...*), *Distance* é a distância do ponto onde a linha de centro da visão do jogador cruza a dada linha, e *Direction* é a direção da linha.

O modelo numérico do campo prevê 55 flags (os gols contam como flags) e 4 linhas possíveis de serem visualizadas. Todos os flags e linhas são mostradas na figura A.1.

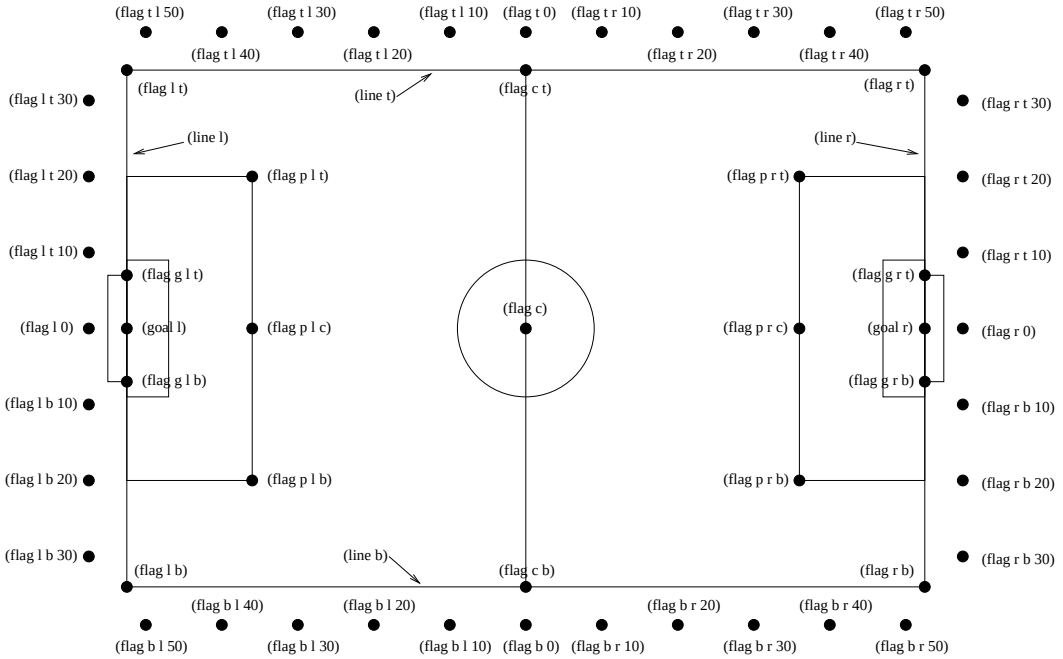


**Figura A.1:** *Localização dos flags e linhas de simulação*

O setor visível de um jogador é dependente de vários fatores. Primeiramente temos os parâmetros do servidor *sense_step* e *visible_angle* que determinam o passo de tempo básico entre a informação visual e quantos graus possui o cone de visão do jogador. Os valores default são 150ms e 90 graus. O jogador pode também influenciar na freqüência e qualidade da informação modificando os parâmetros *ViewWidth* e *ViewQuality*. Estes serão melhores descritos na subseção referente aos comandos. Para calcular os atuais valores de *view_frequency* e *view_angle* do agentes usa-se as seguintes equações:

$$view\_frequency = sense\_step * view\_quality\_factor * view\_width\_factor \quad \text{(A.11)}$$

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

$$view\_angle = visible\_angle * view\_width\_factor \quad \text{(A.12)}$$

onde *view_width_factor* é igual a 0.5 para *ViewWidth* igual a *narrow*, 1 para *ViewWidth* igual a *normal*, e 2 para *ViewWidth* igual a *wide*.

O jogador pode também "ver" um objeto se este está dentro da distância estabelecida em *visible_distance*. Se o objeto está dentro desta distância, mas não no cone de visão, então o jogador pode saber somente o tipo do objeto (bola, jogador, gol, flag), mas não o nome exato do objeto.

O significado do parâmetro *view_angle* é ilustrado na figura A.2.

## A.2 Comandos

A ação do agente no ambiente é viabilizada por meio de um conjunto específico de comandos. O Soccer Server aceita um comando a cada 20ms, respeitando uma série de restrições. Os comandos disponíveis e suas restrições são:

- (`turn` *mi*) - Permite que o jogador execute um giro de [-180, 180] graus em torno do seu próprio eixo. A este comando é associado um argumento *mi*, ou seja, o valor do ângulo em graus.

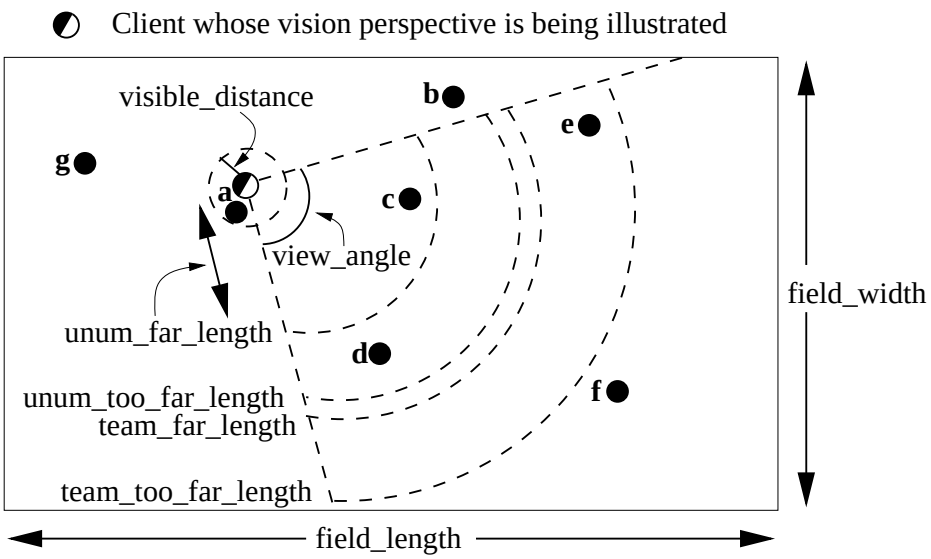


**Figura A.2:** *Campo visual do jogador*

- (`turn neck` *mi*) - Permite que o jogador gire [-180, 180] graus em torno de seu próprio eixo, a parte superior do robô onde se encontra a câmera.

- (`dash` *p*) - Incrementa a velocidade do jogador na direção atual com a potência *p* especificada no argumento. A potência pode assumir um valor inteiro no intervalo [-30, 100].

- (`kick` *p d*) - Chuta a bola com a potência *p* [-30, 100] na direção *d* [-180, 180]. Este comando só será executado se a bola estiver dentro da distância estabelecida pelos parâmetros do simulador (a distância deve ser menor que *kickable_margin* + *ball_size* + *player_size*).

- (`catch` *d*) - Tenta agarrar a bola na direção *d*. A possibilidade de sucesso é definida no parâmetro *catch_possibility*. A bola deve encontrar-se em uma área definida pelo parâmetro *goalie_catchable_area*, um retângulo que por default é de 2m X 1m. Este comando deve ser utilizado apenas pelo goleiro.

- (`say` *msg*) - Difunde uma mensagem para todos os jogadores em um raio de 50m, com o centro no jogador que a enviou.

- (`change_view` *a q*) - Modifica o ângulo do setor visível para *a* ([-45, 45], [-22.5, 22.5], [-90, 90]) e a qualidade visual para *q* (*low*, *high*).

Os valores limites dos intervalos apresentados acima são todos parâmetros ajustáveis do simulador. O soccerserver aceita um novo comando a cada 20ms, entretanto apenas um comando `turn`, `kick` ou `dash`, é executado a cada ciclo de simulação (100ms).

## A.3 Temporização

É importante ressaltar que o servidor é um sistema de tempo real trabalhando com intervalos de tempo discretos (ou ciclos). Cada ciclo tem uma duração específica, e ações que precisam ser executadas em um dado ciclo, devem chegar ao servidor durante o intervalo de tempo certo, sob pena de prejudicar a sincronização entre cliente e servidor.

A comunicação entre o soccerserver e os agentes envolve mensagens síncronas e assíncronas (da Costa e Bittencourt, 1999a), como demonstrado na figura A.3.

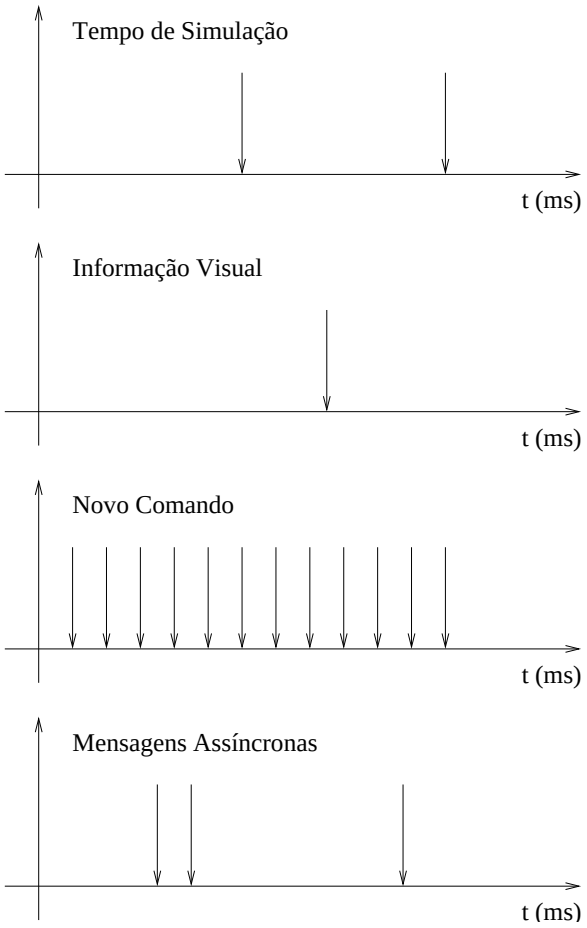


**Figura A.3:** *Temporização no Soccer Server*

- **Tempo de simulação** - Cada ciclo de simulação tem a duração de 100ms.
- **Aceitação de comandos** - O soccerserver aceita um novo comando a cada 20ms.
- **Informação visual** - Depende do modo de visão utilizado (*normal, narrow, wide*) 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 150ms.

- **Informações Auditivas** - São trocadas assincronamente pelo árbitro e pelos demais agentes.

# Referências Bibliográficas


Alonso, E. (2002). Ai and agents. *AI Magazine*, 23(3):25–29. ISSN:0738-4602.

ANSI (1992). Life cycle development of knowledge-based systems using dod-std 2167a. Relatório Técnico G-031-1992, ANSI/AIAA.

ARP (1989). *ARP - Analisador/Simulador de Redes de Petri*. http://www.ppgia.pucpr.br/ maziero/diversos/petri/manual-pt.html. Manual da Versão 2.3.

Asada, M. (2002). An overview of robocup-2002 fukuoka/busan. *AI Magazine*, 24(2):21–40. ISSN:0738-4602.

Austin, J. L. (1962). *How to Do Things with Words*. Claredon Press.

Barkley, R. (1997). *ADHD and The Nature of Self-Control*. The Guilford Press.

Barr, A. e Feigenbaum, E. A. (1981). *The Handbook of Artificial Intelligence*, volume 1. Willian Kaufman.

Bastide, R. (1995). Approaches in unifying petri nets and the object-oriented approach. Em *1st Workshop on Object-Oriented Programming and Models of Concurrency*.

Bittencourt, G. (1997). In the quest of the missing link. Em *International Joint Conference on Artificial Intelligence (IJCAI'97)*, páginas 310 – 315, Nagoya, Japan.

Bittencourt, G. (1998). *Inteligência Artificial: Ferramentas e Teorias*. Editora da UFSC, Florianópolis.

Blum, A. L. e Furst, M. L. (1995). Fast planning through planning graph analysis. Em *IJCAI-95*, páginas 1636–1642. Morgan Kaufmann.

Bordini, R. H., Vieira, R., e Álvaro Freitas Moreira (2001). Fundamentos de sistemas multiagentes. Em Ferreira, C. E., editor, *Jornada de Atualização em Informática (JAI'01)*, volume 2, páginas 3–44, Fortaleza. Sociedade Brasileira de Computação (SBC).

Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., e Mylopoulos, J. (2001). A knowledge level software enginnering methodology for agent oriented programming. Em *Proceedings of the Fifth International Conference on Autonomous Agents*, Montreal, Canada.

Brooks, R. (1986). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*.

Bruchanan, B. G. e Shortliffe, E. H. (1984). *Rule-Based Expert Systems*. Addison-Wesley. ISBN 0-201-10172-6.

Buchanan, B. G., Barstow, D., Bechtel, R., Bennet, J., Clancey, W., Kulikowski, C., Mitchell, T., e Waterman, D. A. (1983). *Building Expert Systems*, chapter Constructing an expert system. Addison-Wesley. Chapter 5.

Cardoso, J., Bittencourt, G., Frigo, L. B., e Pozzebon, E. (2004). Petri nets for authoring mechanism. Em *XV Simpósio Brasileiro de Informática na Educação (SBIE'2004)*, Manaus.

Cardoso, J. e Pradin-Chézalviel, B. (1997). Logic and fuzzy petri nets. Em *Proceedings of 2nd International Workshop on Manufacturing and Petri Nets*, Toulouse, France.

Cardoso, J. e Valette, R. (1997). *Redes de Petri*. Editora da UFSC.

Chang, M. K. e Woo, C. (1991). Sanp: A communication level protocol for negotiations. Em Werner, E. e Demazeau, Y., editors, *Third European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, páginas 31–54.

Chen, M., Foroughi, E., Heintz, F., Huang, Z. X., Kapetanakis, S., Kostiadis, K., Johan Kummeneje, I. N., Obst, O., Riley, P., Timo Steffens, Y. W., e Yin, X. (2001). *RoboCup Soccer Server: for Soccer Server Version 7.07 and later*. URL `www.robocup.org`.

Chen, S.-M., Ke, J.-S., e Chang, J.-F. (1990). Knowledge representation using fuzzy petri nets. *IEEE Transactions on Knowledge and Data Engineering*, 2(3):311–319.

Conte, R. e Castelfranchi, C. (1992). Mind is not enough: Precognitive bases of social interaction. Em *Proceedings of 1992 Symposium on Simulating Societies*, páginas 93 – 110, Guildford, UK.

Coutinho, L. R., Sichman, J. S., e Boissier, O. (2005). Modeling organization in mas: A comparison of models. Em *Proceedings of the 1st Workshop of Software Engineering for Agent-Oriented Systems (SEAS'05*, Uberlândia - MG.

da Costa, A. C. P. L. e Bittencourt, G. (1997). Parla: A cooperation language for cognitive multi-agent systems. Em *8th Portuguese Conference of Artificial Inteligence*, páginas 492–497.

da Costa, A. C. P. L. e Bittencourt, G. (1999a). Soccer server: Um simulador para o futebol de robôs da robocup federation tutorial-3. Em *Encontro Nacional de Inteligência Artificial*.

da Costa, A. L. e Bittencourt, G. (1999b). From a concurrent architecture to a concurrent autonomous agents architecture. Em *International Joint Conference on Artificial Intelligence (IJCAI'99)*.

da Costa, A. L. e Bittencourt, G. (2002). Dynamic social knowledge: The timming evaluation. Em *XVI Brazilian Symposium on Artificial Intelligence - SBIA'02*, páginas 175–184. Springer.

da Costa, A. L., Bittencourt, G., da Silva, L. R., e Gonçalves, E. M. N. (2003). Expert-coop++: Ambiente para desenvolvimento de sistemas multiagentes. Em *ENIA - Encontro Nacional de Inteligência Artificial*.

Dastani, M., Hulstijn, J., e Dignum, F. (2004). Issues in multiagent system development. Em *Third International Joint Conference on Autonomous Agents and Multiagent System (AAMAS'04)*, volume 2, páginas 922–929.

David, R. e Alla, H. (1994). Petri nets for modeling of dynamic systems - a survey. *Automatica*, 30(2):175–202.

de Almeida, H. O., da Silva, L. D., Perkusich, A., e de Barros Costa, E. (2005). A formal approach for the modelling and verification of multiagent plans based on model checking and petri nets. Em Choren, R., Garcia, A., Lucena, C., e Romanovsky, A., editors, *Software Engineering for Multi-Agent Systems III: Research Issues and Pratical Applications*, volume 3390/2005. Springer-Verlag GmbH. ISSN: 0302-9743.

de Barros Costa, E. e Perkusich, A. (1996). Modeling the cooperative interactions in a teaching/learning situation. Em *Proceedings of The Intelligent Tutoring Systems (ITS-96)*, Montreal, Canada.

de Brito, S. R., Gava, T. B. S., de Lira Tavares, O., e de Menezes, C. S. (2001). Metodologias para desenvolvimento de sistemas multiagentes: Visão geral e comparação. Em *Anais do Encontro Nacional de Inteligência Artificial (ENIA'01)*, Fortaleza. Sociedade Brasileira de Computação (SBC).

de Freitas, F. L. G. e Bittencourt, G. (2002). Comunicação entre agentes em ambientes distribuídos abertos: o modelo "peer-to-peer". *Revista Eletrônica de Iniciação Científica (REIC)*, II(II).

Decker, K. S. (1996). Tæms: A framework for environment centered analysis and design of coordination mechanisms. Em O'Hare, G. M. P. e Jennings, N. R., editors, *Fundations of Distributed Artificial Intelligence*, chapter 16, páginas 429–447. John Wiley, New York.

Dignum, V. e Dignum, F. (2001). Modelling agent societies: Co-ordination frameworks and institutions. Em *Proceedings of the 10th Portuguese Conference on Artificial Intelligence (EPIA'01)*, páginas 191–204, Berlin. Springer. LNAI 2258.

Durfee, E. H. e Rosenschein, J. S. (1994). Distributed problem solving and multi-agent systems: Comparisons and examples. Em *International Workshop on Distributed Artificial Intelligence*.

Federation, T. R. (2003). Robocup. www.robocup.org. Home Page, Acessado em: 28/08/2003.

Feigenbaum, E. (1977). The art of artificial intelligence: Themes and case studies in knowledge engineering. Em *Proceedings of IJCAI-5*, páginas 1014–1029.

Ferber, J. (1999). *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*. Addison-Wesley Pub Co. ISBN:0201360489.

Ferber, J. e Gasser, L. (1991). Intelligence artificielle distribuée. Em *Tutorial Notes of the 11th Conference on Expert Systems and their Aplications*, France.

Ferber, J., Gutknecht, J., e Michel, F. (2004). From agents to organizations: an organizational view of multi-agent systems. Em Giorgini, P., Müller, J. P., e Odell, J., editors, *Agent-Oriented Software Engineering IV: 4th International Workshop (AOSE 2003) - Revised Papers*, volume 2935 of *Lecture Notes in Computer Science*, páginas 214–230. Springer, Melbourne, Australia.

Ferber, J. e Gutknecht, O. (1998). A meta-model for the analysis and design of organizations in multi-agent systems. Em Demazeau, Y., editor, *Proceedings of the 3rd International Conference on Multi-Agens Systems (ICMAS'98)*, páginas 128–135. IEEE Press.

Finin, T., Labrou, Y., e Mayfield, J. (1995). *KQML as an Agent Comunication Language*. MIT Press, Cambridge.

Frozza, R. e Alvares, L. O. C. (2001). Análise de coordenação em ambientes multiagentes. Em *Encontro Nacional de Inteligência Artificial (ENIA'01)*. Sociedade Brasileira de Computação (SBC).

Garcia, A. C. B. e Sichman, J. S. (2003). Agentes e sistemas multiagentes. Em Rezende, S. O., editor, *Sistemas Inteligentes: Fundamentos e Aplicações*, chapter 11, páginas 269–306. Manole.

Garcia, A. C. B., Varejão, F. M., e Ferraz, I. N. (2003). Aquisição de conhecimento. Em Rezende, S. O., editor, *Sistemas Inteligentes: Fundamentos e Aplicações*, páginas 51–85. Manole, Barueri-SP, 1o edição.

Genrich, H. J. (1991). Predicate/transition nets. Em *High-Level Petri Nets: Theory and Application*, páginas 3–43. Springer-Verlag, Berlin. ISBN 3-540-54125-X.

Goc, M. L., Frydman, C., e Torres, L. (2002). Verification and validation of the sachem conceptual model. *International Journal Human-Computer Studies*, 56:199–223.

Goldberg, E. (2002). *The Executive Brain: Frontal Lobes and the Civilized Mind*. Oxford University Press.

Gonçalves, E. M. N. (2001). Otimização de controladores nebulosos e sistemas especialistas reativos utilizando algoritmos genéticos. Dissertação de Mestrado, UFSC - Universidade Federal de Santa Catarina.

Guarino, N. e Giaretti, P. (1995). *Towards Very Large Knowledge Bases*, chapter Ontologies and Knowledge Bases: Towards a Terminological Clarification, páginas 25–32. IOS Press.

Holvoet, T. (1995). Agents and petri nets. *Petri Net Newsletters*, páginas 3–8.

Holvoet, T. e Verbaeten, P. (1996). Synchronization specifications for agents with net-based behavior description. Em *Proceedings of Conference, Symposium on Discrete Events and Manufacturing Systems*, páginas 613–619, Lille, France.

Hong, J. e Bae, D. (1998). Hoonets: Hierarchical object-oriented petri nets for system modeling and analysis. URL `citeseer.csail.mit.edu/hong98hoonets.html`.

Huber, P., Jensen, K., e Shapiro, R. (1990). Hierarchies in coloured petri nets. Em Rozemberg, G., editor, *Advances in Petri Nets 1990*. Springer-Verlag.

Hübner, J. F., Bordini, R. H., e Vieira, R. (2004). Introdução ao desenvolvimento de sistemas multiagentes com jason. Em *Anais a XII Escola Regional de Informática - SBC*.

Hübner, J. F. e Sichman, J. S. (2003). Aplicação de organização de sistemas multiagentes em futebol de robôs. Em *XI Escola de Informática do SBC*, volume 1, páginas 119–147, Lages-SC. Angelo Augusto Frozza.

Hübner, J. F., Sichman, J. S., e Boissier, O. (2002). Moise$^+$: Towards a structural, functional, and deontic model for mas organization. Em *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'02)*, páginas 501–502. ACM Press.

JACK (2005). *JACK$^{TM}$ Intelligent Agents: Agent Manual*. Agent Oriented Software Pty. Ltd., http://www.jackagents.com/shared/demosNdocs/Agent_Manual.pdf. Acessado em 30/10/2005.

Jensen, K. (1996). *Coloured Petri Nets: Basic concepts, analysis methods and practical use*, volume 1. Springer-Verlag, second edição.

Junius, M. e Stepple, M. (1997). *CNCL Reference Manual*. Universität Aachen. http://www.comnets.rwth-aachen.de/cnroot_engl.html.

Kidd, A. L. (1987). *Knowledge Acquisition for Expert Systems*, chapter Knowledge Acquisition - An Introductory Framework, páginas 1–16. Plenum Press, New York and London. ISBN 0-306-42454-1.

Kitano, H., Asada, M., anda Itsuki Noda, Y. K., e Osawa, E. (1995). Robocup: The robot world cup initiative. Em *IJCAI-95 Workshop on Entertainment and AI/Alife*.

Kitano, H., Asada, M., Kuniyoshi, Y., Noda, I., e Osawa, E. (1997). Robocup: The robot world cup initiave. Em *The First International Conference on Autonomous Agent (Agents-97)*.

Koriem, S. M. (2000). A fuzzy petri net tool for modeling and verification of knowledge-based systems. *The Computer Jornal*, 43(3).

Lakos, C. A. (1995). The object orientation of object petri nets. Em *Workshop on Object Oriented Programming and Models of Concurrency*.

Lakos, C. A. (1997). Object Oriented Modelling with Object Petri nets. Em *Advances in Petri nets*, Lecture Notes in Computer Science. Springer, Berlin.

Lee, J. e Lai, L. F. (2002). A high-level petri nets-based approach to verifying task structures. *IEEE Transactions on Knowledge and Data Engineering*, 14(2):316–335.

Li, X. e Yu, W. (2001). Object oriented fuzzy petri net for complex knowledge system modeling. Em *Proceedings of the 2001 IEEE International Conference on Control Applications*, páginas 476–481, Mexico City, Mexico.

Li, X., Yu, W., e Lara-Rosano, F. (2000). Dynamic knowledge inference and learning under adaptive fuzzy petri net framework. *IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews*, 30(4):442–450.

Lind, J. (2001). Issues in agent-oriented software engineering. Em *Agent-Oriented Software Engineering, First International Worshop (AOSE 2000) - Revised Papers*, volume 1957 of *Lecture Notes in Computer Science*. Springer-Verlag. ISBN: 3-540-41594-7.

Looney, C. G. (1988). Fuzzy petri nets for rule-based decisionmaking. *IEEE Transactions on Systems, Man and Cybernetics*, 18(1).

Lugo, G. G., Hübner, J. F., e Sichman, J. S. (2001). Representação e evolução de esquemas sociais em sma: um enfoque funcional. Em *Encontro Nacional de Inteligência Artificial (ENIA'01)*. Sociedade Brasileira de Computação (SBC).

Malone, T. W. e Crowston, K. (1994). The interdisciplinary study of coordination. *ACM Computing Survey*, 26(1):87–119.

Martins, P. (2003). Aprendizado de máquina para otimizaçãao de parâmetros em sistemas baseados em conhecimento. Dissertação de Mestrado, Universidade Federal de Santa Catarina, Florinópolis.

Miller, L. (1989). A realistic industrial strengh life cycle model for knowledge-based systems development and testing. Em *AAAI Workshop Notes: Validation and Verification*.

Minsky, M. (1975). A framework to represent knowledge. *The Psychology of Computer Vision*, páginas 211–277.

Miranda, M. V. C. e Perkusich, A. (1999). Modeling and analysis of a multi-agent system using colored petri nets. Em Portinale, L., Valette, R., e Zhang, D., editors, *Proceedings of the Workshop on Application of Petri Nets to Intelligent System Development*, páginas 59–70.

Moldt, D. e Wienberg, F. (1997). Multi-agent-systems based on coloured petri nets. Em *Proceedings of the 18th International Conference on Application and Theory of Petri Nets*, Toulouse, France.

Murata, T. (1989). Petri nets: Properties, analysis and applications. *IEEE*, 77(4):481–497.

Musen, M. A., Fagan, L. M., Combs, D., e Shortlife, E. H. (1987). Use of a domain model to drive an interactive knowledge-edit tool. *International Journal of Man-Machines Studies*, 12:347–375.

Nazareth, D. L. (1993). Investigating the applicability of petri nets for rule-based system verification. *IEEE Transactions on Knowledge and Data Engineering*, 4(3):402–415.

Newell, A. (1982). The knowledge level. *Artificial Intelligence*, 18(1):87–127.

Noda, I. e Matsubara, H. (1996). Soccer server and researches on multi-agent systems. Em *Proceedings of the IROS-96 Workshop on RoboCup*. URL `citeseer.ist.psu.edu/noda96soccer.html`.

Nwana, H. S. (1995). Software agents: An overview. *Knowledge Engineering Review*, 11(2):205–244. URL `citeseer.nj.nec.com/nwana96software.html`.

Nwana, H. S., Ndumu, D. T., Lee, L. C., e Collis, J. C. (1999). Zeus: A toolkit for building distributed multi-agent systems. *Applied Artificial Intelligence Journal*, 13(1):129–186.

Odell, J., Parunak, H. V. D., e Bauer, B. (2000). Extending uml for agents. Em *Proceedings of the Agent-Oriented Information Systems Workshop at the 17th National Conference on Artificial Intelligence*.

Padgham, L. e Winikoff, M. (2002). Prometheus: A methodology for developing intelligent agents. Em *Agent-Oriented Software Enginnering III, Third International Workshop (AOSE 2002)*, volume 2585/2003, páginas 174–185. Lecture Notes in Computer Science, Springer-Verlag. ISSN: 0302-9743.

Petri, C. A. (1962). *Kommunikation mit Automaten*. Tese de Doutorado, Institut für Instrumentelle Mathematik.

Plant, R. e Gamble, R. (2003). Methodologies for the development of knowledge-based systems, 1982-2002. *The Knowledge Engineering Review*, 18(1):47–81.

Post, E. (1943). Formal reductions of the general combinatorial problem. *American Journal of Mathematics*, páginas 65:197–268.

Pressman, R. S. (1995). *Engenharia de Software*. Makron Books.

Quillian, M. R. (1968). Semantic memory. *Semantic Information Processing*, páginas 216–270.

Rezende, S. O., Pugliesi, J. B., e Varejão, F. M. (2003). Sistemas baseado em conhecimento. Em Rezende, S. O., editor, *Sistemas Inteligentes: Fundamentos e Aplicações*, chapter 2, páginas 13–49. Manoele. ISBN 85-204-1683-7.

Russel, S. e Norvig, P. (1995). *Artificial Intelligence, A Modern Approach*. Alan Apt.

Scalabrin, E. E., Vandenberghe, L., Azevedo, H., e Barths, J.-P. A. (1996). A generic model of cognitive agent to develop open systems. *Lecture Notes in Artificial Intelligence - Advances in Artificial Intelligence*, 1159:61–70. Proceedings of the 13th Brazilian Symposium on Artificial Intelligence.

Schreiber, G., Akkermans, H., Anjewierden, A., de Hoog, R., Shabdbolt, N., de Velde, W. V., e Wielinga, B. (1999). *Knowledge Engineering and Management, The CommomKADS Methodology*. MIT Press, Cambridge.

Shoham, Y. (1993). Agent-oriented programming. *AI*, 60:51–92.

Shortliffe, E. H. (1976). *Computer-Based Medical Consultations: MYCIN*. Americam Elsevier, New York.

Shortliffe, E. H., Scott, A. C., Bischoff, M. B., Campbell, A. B., van Melle, W., e Jacobs, . C. D. (1981). Oncocin: An expert system for oncology protocol management. Em *International Joint Conference on Artificial Intelligence (IJCAI '81)*, páginas 876–881, Vancouver, CA.

Sibertin-Blan, C. (1985). High-level Petri nets with data structures. Em *European Workshop on Application and Theory of Petri net*, páginas 141–170, Helsinki, Finland.

Smith, R. G. (1980). The contract net protocol:hifh-level communication and control in a distributed problem solving. *IEEE Transactions on Computers*.

Stokes, M., editor (2001). *Managing Engineering Knowledge, MOKA: Methodology for Knowledge Based Engineering Applications*. ASME Press.

Studer, R. e Benjamins, R. V. (1998). Knowledge engineering: Principles and methods. *Data Knowledge Engineering*, 25(1-2):161–197.

Sun Microsystems (1990). *Networking Programming Guide*. Sun Microsystems.

Sutton, R. S. e Barto, A. (1998). *Reinforcement Learning: An Introduction*. MIT Press.

Tuthill, G. S. (1989). *Knowledge Engineering - Concepts and Practices for Knowledge-Based Systems*. TAB Books. ISBN 0-8306-9297-5.

Weitzel, J. R. e Kershberg, L. (1989). Developing knowledge-based systems: reorganizing the system development life cycle. *Communications of the ACM*, 32(4):482–490.

Wood, M. F. e DeLoach, S. A. (2001). An overview of the multiagent systems enginnering methodology. Em Ciancarini, P. e Wooldridge, M., editors, *Agent-Oriented Software Engineering - Proceedings of the First International Workshop on Agent-Oriented Software Engineering*, volume 1957, páginas 207–221, Berlin. Lecture Notes in Computer Science, Springer-Verlag.

Wooldridge, M., Jennings, N. R., e Kinny, D. (2000). The gaia methodology for agent-oriented analysis and design. *Autonomous Agents and Multi-Agent Systems*, 3:285–312.

Wu, C.-H. e Lee, S.-J. (1997). Enhanced high-level petri nets with multiple colors for knowledge verification/validation of rule-based expert systems. *IEEE Transactions on Systems, Man and Cybernetics - Part B: Cybernetics*, 27(5):760–773.

Xu, D., Volz, R. A., e Ioerger, T. R. (2002). Generating parallel based on planning graph analysis of predicate/transition nets. Em *Proceedings of the 2002 International Conference on Artificial Intelligence (IC-AI'02)*, páginas 440–446.

Xu, D., Volz, R. A., e Yen, J. (2003). Modeling and analysing multi-agent behaviors using predicate/transitions nets. *International Journal of Software Engineering and Knowledge Engineering*, 13(1):103–124.

Zhang, D. e Nguyen, D. (1994). Prepare: A tool for knowledge base verification. *IEEE Transactions on Knowledge and Data Engineering*, 6(6).