UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

FABIO YOSHIMITSU OKUYAMA

Modelo MAS-SOC: Integrando Ambientes e Organizações para Simulações Baseadas em Sistemas Multiagentes Situados

Tese apresentada como requisito parcial
para a obtenção do grau de
Doutor em Ciência da Computação

Prof. Dr. Antônio Carlos da Rocha Costa
Orientador

Prof. Dr. Rafael Heitor Bordini
Co-orientador

Porto Alegre, dezembro de 2008

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Okuyama, Fabio Yoshimitsu

Modelo MAS-SOC: Integrando Ambientes e Organizações para Simulações Baseadas em Sistemas Multiagentes Situados / Fabio Yoshimitsu Okuyama. – Porto Alegre: PPGC da UFRGS, 2008.

102 f.: il.

Tese (doutorado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2008. Orientador: Antônio Carlos da Rocha Costa; Co-orientador: Rafael Heitor Bordini.

1. Sistemas multiagentes. 2. Modelo para simulação social. 3. Infraestrutura normativa. 4. Organização de sistemas multiagentes. 5. Sistemas multiagente situados. I. Rocha Costa, Antônio Carlos da. II. Bordini, Rafael Heitor. III. Título.

AGRADECIMENTOS

Gostaria de agradecer a orientação Prof. Antônio Carlos da Rocha Costa por todo o apoio oferecido nesta longa empreitada. Ao co-orientador Prof. Rafael H. Bordini, que acompanha este trabalho deste o mestrado.

Aos membros da banca, Profª. Ana Lúcia Cetertich Bazzan, Profª. Renata Vieira, Profª. Diana Francisca Adamatti pelas valiosas correções e sugestões a este trabalho.

Agradeço ao CNPq (doutorado) e à CAPES (sanduíche) pelo suporte financeiro, sem o qual este trabalho não seria possível.

Aos colegas de laboratório Pablo Souza Grigoletti, Juliana Kaizer Vizzotto, Raquel de Miranda Barbosa e Márcia Häfele Islabão Franco, pela amizade e companherismo.

Agradeço a meus familiares (Okuyama e Machado) que me ajudaram e incentivaram na realização do curso de doutorado. À Familia Aires (Alfredo, Adelaide e Hannah) pela amizade e apoio. Agradeço à minha esposa Flavia, por todo o carinho e apoio que precisei.

Agradeço a todas as pessoas com quem convivi nesta longa empreitada do curso de doutorado, que me ajudaram direta ou indiretamente na sua realização. Apesar de ser uma lista de agradecimentos extensa, sou muito agradecido a todos.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

BDI	Belief Desire Intention
IAD	Inteligência Artificial Distribuída
IE	Instituições Eletrônicas
JNI	Java Native Interface
MABS	Multi-Agent Based Simulation
SMA	Sistemas Multiagente

LISTA DE FIGURAS

RESUMO

Este trabalho encontra-se situado na área de Inteligência Artificial, mais especificamente na modelagem de Sistemas Multiagentes destinados à simulação social. A área de pesquisa de simulação social baseada em agentes é uma área recente e bastante promissora. Por tratar de problemas extremamente complexos, existe a necessidade de criar modelos e abstrações para possibilitar sua realização. Dando continuidade a trabalhos anteriores, esta tese pretende consolidar uma nova versão do modelo MAS-SOC, destinado à definição de simulações sociais baseada em Sistemas Multiagentes Situados, com organizações que funcionam em ambientes determinados (*organizações situadas*). É proposta uma abordagem integrada, que conecta de maneira forte as entidades agentes, organizações e ambiente, sendo os agentes, o ambiente e as estruturas da organização tratadas como entidades de primeira ordem. A definição do ambiente é feita através da linguagem ELMS, estendida com uma infraestrutura normativa. Esta infraestrutura normativa é composta basicamente por *objetos normativos* e *espaços normativos* que permitem a distribuição espacial da informação normativa no ambiente, possibilitando a contextualização das normas que os agentes devem seguir em um escopo espacial determinado. A contextualização das normas facilita a sua operacionalização e a verificação de conformidade, reduzindo também a possibilidade de interpretações errôneas das normas. Com isso, a infraestrutura normativa conecta ambiente físico à estrutura da organização do sistema multiagente. Além disso, o modelo proposto usa um esquema mínimo de definição das organizações para armazenar informações e restrições que o projetista da simulação prefira representar desta forma. Esta maneira integrada de modelar os sistemas multiagentes, associando a organização ao espaço em que a organização deve funcionar, pode facilitar a modelagem de grandes sistemas, pois o conceito de espaço normativo permite que a modelagem seja feita através da partição do ambiente físico em módulos onde as atividades da organização situada são realizadas. Assim, o modelo trata de forma bastante prática o uso das normas organizacionais que podem ser representadas através de objetos normativos. Esta forma de representação possibilita aos agentes decidirem sobre a aderência as normas, já que estas não estão embutidas no mecanismo de raciocínio dos agentes. Além disso, a forma *contextualizada* das normas facilita raciocínio sobre elas e possibilita a aderência a normas previamente desconhecidas pelos agentes.

Palavras-chave: Sistemas multiagentes, modelo para simulação social, infraestrutura normativa, organização de sistemas multiagentes, sistemas multiagente situados.

MAS-SOC Model: Integrating Environments and Organisations to Simulations based on Situated Multiagent Systems

ABSTRACT

This work is situated in the research area of Artificial Intelligence, specifically the modelling of Multi-agent systems for social simulation. The research area of agent based social simulation is a recent and interesting area. In order to handle with its very complex problems, it requires the development of models and abstractions to make possible its realisation. Continuing previous works, this thesis aims to consolidate extensions to the MAS-SOC model, in order to turn it into a suitable model for the social simulation based on situated multi-agent systems with organizations that operate in determined environments (*situated organisations*). It is proposed an integrated approach in which multiagent entities such as agents, organisations and environments are strongly connected to each other, and the environment and the organisational structures being treated as first order entities. The definition of the environment is made with the use of the ELMS language, which have been extended with a normative infrastructure. The normative infrastructure is composed essentially by *normative objects* and *normative places*, which are means for the spatial distribution of the normative information over the environment, allowing the contextualisation of the norms in a bounded spatial scope. The norms being bounded in a specific spatial scope facilitates its operationalisation and conformity checks, also reducing the possibilities of norms misinterpretations. Thus, the normative infrastructure connects the physical environment to the organisational structures of the multiagent system. The thesis proposes that this integrated approach to model multi-agent systems may ease the modelling of large scale systems, since it allows the partition of the environment in a modular way, facilitating the operationalisation and verification of the adequacy of the structure of an organisation to the physical space where it is located, and also reducing the possibility of the misinterpretations of norms by the agents, through the contextualisation of norms. Also, the proposed scheme uses a minimal structure for the definition of the organisations in order to store information that the simulation designer prefers to represent in such way. Further, the proposed approach allows a very practical way to use of norms in a physical environment, by allowing the agents to reason about following a norm abiding behaviour or not, since the norms are not *hard-wired* in their reasoning mechanisms, and its *contextualized* form facilitates that agents reason about adhering to norms that were previously unknown to them.

Keywords: multi-agent systems, platforms for social simulation, normative infrastructure, multi-agent systems organizations, situated multiagent systems.

1 INTRODUÇÃO

Um Sistema Multiagente (SMA) é tipicamente composto de múltiplos agentes, que compartilham um ambiente comum, estruturas organizacionais e ferramentas para interação entre o ambiente, estruturas organizacionais e agentes. Neste contexto, *SMA Situados* são definidos pela presença de uma estrutura espacial explícita onde os agentes existem, sendo geralmente caracterizados pelos mecanismos específicos de percepção e interação baseados em propriedades contextualizadas que os agentes devem dispor para acesso ao ambiente, tal como o posicionamento relativo dos agentes (WEYNS; VIZZARI; HOLVOET, 2006).

Entretanto, nas abordagens de SMA que dão enfoque central às formas de organização dos SMA, as estruturas organizacionais são usualmente definidas de forma *não-situada*, ou seja, independente do ambiente onde o sistema opera, seja este físico, virtual ou espacial. De maneira similar, a maioria das abordagens para SMA normativos, tais como as mencionadas em (BOELLA; TORRE; VERHAGEN, 2007), abordam várias questões sobre como normas podem ser formalmente definidas, verificadas e impostas, porém não têm estabelecido ainda, de modo claro, a forma pela qual elas podem ser usadas no desenvolvimento de sistemas situados em um ambiente determinado.

Assim, na maioria das abordagens existentes para o projeto orientado à organização de sistemas multiagentes, sejam estas focadas em estruturas organizacionais ou em estruturas normativas, existe uma lacuna conceitual entre o ambiente, estruturas organizacionais e estruturas normativas. Isto é mais claro nos casos em que o SMA é situado em um ambiente espacial (simulado ou não), devido à inexistência de uma conexão explícita entre os elementos das estruturas organizacionais e normativas e os espaços físicos onde tais elementos devem existir e operar.

À primeira vista, a conexão entre o ambiente e organizações pode parecer supérflua para o projeto e entendimento dos sistemas multiagentes em geral, mas as *organizações situadas* constituem os casos em que o ambiente (posicionamento de espaços organizacionais, distâncias, conjuntos de objetos presentes) influencia a forma e o funcionamento da organização e dos agentes, fazendo com que adquira importância a conexão entre as estruturas organizacionais e normativas e as estruturas do ambiente para a concreta existência de tais organizações.

De maneira específica, tal conexão entre estruturas do ambiente e estruturas organizacionais é importante quando se está lidando com organizações cuja estrutura normativa não opera de maneira homogênea através do ambiente físico. Ou seja, quando é importante para as organizações que regras que regulam o comportamento dos agentes variem de acordo com os diferentes espaços onde os agentes se encontram e operam. Por exemplo, pode haver uma fábrica onde os trabalhadores que estão em um local excessivamente barulhento não poderem trabalhar mais que duas horas consecutivas sem um intervalo,

enquanto outros trabalhadores em um local com menos barulho poderiam trabalhar até três horas, com intervalo menor.

Além disso, organizações também fazem uso de normas que são espacialmente e temporalmente limitadas, ou seja, normas que se referem a algum espaço e tempo específico. Algumas regras de comportamento são definidas desta forma, tal como "Proibido fumar neste local" ou "Não entre após as 18h".

Porém existem regras mais complexas e mesmo indiretas que são espacialmente ou temporalmente limitadas. Por exemplo, uma placa indicando "Diretor" em uma mesa em um escritório, indica, entre outras coisas, que os papéis existentes nesta mesa deveriam ser lidos apenas pelo diretor ou alguém com permissão explícita, significando neste caso uma norma tal como "Não leia os papéis sobre esta mesa". Porém, um documento que voou da mesa para algum canto do escritório não está mais sujeito a tal norma.

Igualmente, o posicionamento de objetos físicos podem ser usados pelas organizações para indicar o *poder* de seus agentes. Para um porteiro é dada a autoridade para verificar e controlar o acesso de entrada e saída de um recinto, sendo o sinal para tal autoridade é o local onde o agente está posicionado e o uniforme que está usando. Os outros agentes entendem que aquele agente específico tem a autoridade de um porteiro através da percepção do local que tal agente ocupa, não através de uma mensagem explícita sobre tal fato. Um agente policial tem o poder para questionar as pessoas na rua devido à posse de um determinado tipo de insígnia, que lhe identifica como sendo um agente com autoridade para realizar tal tarefa. Além disso, pode ocorrer que esta autoridade seja limitada a um determinado município e não necessariamente em outro.

Resumidamente, em diversas situações as operações de uma organização fazem uso de referências espaciais enquanto recursos do ambiente propagam e instanciam normas e poderes dos agentes, fazendo com que a organização tenha as características de uma *organização situada*. A falta de uma conexão explícita entre estruturas organizacionais e o ambiente representa, desta forma, uma lacuna conceitual nos modelos de sistemas multiagentes. Esta lacuna pode ser ainda maior se considerada a forma como sistemas normativos podem ser incorporados nesses modelos, onde é necessário encontrar uma forma prática para implementar normas que regulam o comportamento de agentes nas organizações.

Esta é especificamente a lacuna que este trabalho pretende abordar, de maneira que se possa dispor de um modelo de simulação social de organizações situadas que contemple a articulação adequada entre agentes, organizações e ambientes. O modelo proposto combina um modelo de ambiente, um modelo básico de estruturas organizacionais e um modelo de estruturas normativas.

De maneira mais específica, este trabalho visa desenvolver um modelo de SMA associado à plataforma para simulação multiagente chamada MAS-SOC (BORDINI et al., 2005). Em trabalhos anteriores, foi apresentada a linguagem para descrição de ambientes ELMS (OKUYAMA; BORDINI; ROCHA COSTA, 2005; OKUYAMA, 2003). A descrição explícita do ambiente é baseada nos conceitos de recursos, representação espacial e de corpos de agentes, descrevendo também seus sensores e efetores. A descrição do ambiente foi estendida para abranger a definição de partes da estrutura normativa de uma organização através da noção de *infraestrutura normativa*, a qual permite a ligação das estruturas ambientais com as estruturas organizacionais do SMA a ser simulado, conforme apresentado em (OKUYAMA; BORDINI; ROCHA COSTA, 2007a).

Desta forma este modelo possibilita a especificação de simulações baseadas em SMA tendo como características a forte ligação entre as entidades que compõem o SMA, além

da abordagem prática das normas, o que permite o uso efetivo de normas na simulação, incluindo a possibilidade de simular sistemas onde agentes tem a capacidade de racionar sobre aderir a normas previamente desconhecidas.

1.1 Motivação

De acordo com Castelfranchi (1998), *"a maior preocupação da simulação social é o estudo experimental e a modelagem de conjuntos de efeitos imprevisíveis de uma população de agentes em um ambiente comum, o que seria o estudo da complexidade, emergência, auto-organização e dinâmica originada do comportamento dos agentes."* Neste contexto, a simulação social é um problema extremamente complexo e naturalmente distribuído, ao qual a solução baseada em SMA se aplicaria de forma bastante adequada. Assim, a simulação social baseada em SMA seria uma ferramenta importante para teorização de questões sociais. Da mesma forma, as ciências sociais são muito importantes para a área de SMA no que diz respeito a obtenção de cooperação, coordenação e interação de agente autônomos de maneira geral. Assim, acredita-se haver uma troca bastante produtiva entre as ciências sociais e SMA, de maneira mais específica entre a simulação social e SMA. Ferber em (FERBER, 1999), afirma que *"mesmo quando não simulam um mundo real ou não permitam a resolução de problemas específicos, a construção de "mundos sintéticos" tem grande importância na pesquisa de sistemas multiagente pois torna possível analisar certos mecanismos de interação de maneira mais detalhada que em uma aplicação real"*. Estendendo este conceito, seria possível afirmar que além dos mecanismos de interação, também seria possível analisar as entidades e modelos que interagem de maneira mais detalhada.

Em Dignum (2002), o autor afirma que várias tentativas de formalizar normas, tais como as lógicas deônticas vem sendo realizadas. Dignum afirma que a pesar de ser possível capturar as normas desta forma e associar a um determinado tipo de semântica, este tipo de formalização não indica como a norma deve ser interpretada em um determinado contexto.

Tendo esses conceitos como referência, o objetivo principal do MAS-SOC (**M**ulti-**A**gent **S**imulations for the **SOC**ial Sciences) é fornecer uma plataforma para criação de simulações baseadas em agentes que não exija grande experiência em programação dos seus usuários, permitindo o uso de tecnologias do estado-da-arte na área de SMA. Mais especificamente, esta plataforma deve permitir o projeto e implementação de agentes *cognitivos*, com uma interface gráfica para facilitar as especificações de ambientes multiagentes, planos, agentes e simulações completas, e também o gerenciamento de bibliotecas destes componentes. A partir de informações dadas pelos usuários, o sistema gera códigos fontes para os interpretadores das linguagens de programação de agentes cognitivos e para a especificação de ambientes multiagentes.

Assim, um dos objetivos a longo prazo do MAS-SOC, é possibilitar o uso de mecanismos de simulação que conciliem cognição e emergência. Este objetivo especificamente se inspira nas idéias de Castelfranchi (CASTELFRANCHI, 2001, 1998) que somente a simulação social com agentes cognitivos irá permitir o estudo de *mentes* de agentes individualmente e as ações coletivas emergentes, que interagem determinando-se mutuamente. Em outras palavras, o objetivo (a longo prazo) é fornecer as condições básicas para que a plataforma MAS-SOC ajude no estudo de um problema fundamental nas ciências sociais, que também é de grande relevância nos sistemas multiagentes, o problema da relação *micro-macro* (CONTE; CASTELFRANCHI, 1995).

1.2 Objetivos

Este trabalho tem como objetivo propor uma abordagem integrada para projeto de SMA destinados à simulação social baseada em agentes, centralizada em uma descrição explícita do ambiente integrando agentes e estruturas organizacionais e normativas, integração possibilitada pela infraestrutura de objetos normativos associados ao ambiente. Além disso, tem como objetivo organizar este modelo de forma a garantir sua implementabilidade e seu uso prático no desenvolvimento de simulações sociais de baseadas em SMA situados, integrando ambiente e organizações possibilitando assim a definição de *organizações situadas*.

O modelo proposto está associado também ao desenvolvimento da plataforma MAS-SOC de modo a prover uma ferramenta para desenvolvimento de simulações sociais sem a necessidade de grandes conhecimentos de programação. Tal objetivo, poderá ser viabilizado através do uso do modelo a ser construído na tese, pois acredita-se que tal modelo torna menos abstrata a tarefa de modelar SMA espacialmente e temporalmente situados.

1.3 Histórico

No ano de 1.996, Anand Rao apresentou a linguagem abstrata AgentSpeak(L), para a programação de agentes BDI (RAO, 1996). Baseada nessa definição da linguagem, Rodrigo Machado sob orientação do professor Rafael H. Bordini desenvolveu em seu trabalho de iniciação científica o primeiro interpretador para a linguagem AgentSpeak(L), denominado SIM_SPEAK, apresentado em (MACHADO; BORDINI, 2002).

A partir disso, dentro do projeto CUCLA(Combining Cognitive and Utilitarian Coordination in a Layered Agent Architecture) sob orientação das professoras Ana Lúcia Cetertich Bazzan, Rosa Maria Vicari e o professor Rafael H. Bordini, foi desenvolvido um novo interpretador desenvolvido em C++ por Rafael Jannone e Daniel M. Basso. Este novo interpretador, chamado AgentSpeak(XL), apresentava algumas extensões à linguagem, como a possibilidade de definir bibliotecas de ações, além da possibilidade de fazer uso de um escalonador de utilidade integrado. Alguns dos resultados obtidos podem ser encontrados em (BORDINI et al., 2002).

No desenvolvimento do projeto MAS-SOC (BORDINI et al., 2004) (ver Capítulo 5), com a disponibilidade de um interpretador para a linguagem AgentSpeak(L), tornou se necessário encontrar uma forma prática para definir os ambientes para as simulações. Como a definição de ambientes para simulações era um assunto pouco abordado, sob a orientação do professor Rafael H. Bordini, foi desenvolvida uma linguagem para definição de ambientes chamada ELMS (Environment Language for Multiagent Simulation) (OKUYAMA, 2003; OKUYAMA; BORDINI; ROCHA COSTA, 2005) e seu interpretador que funcionaria fazendo uso do interpretador AgentSpeak(XL). A partir deste protótipo foi possível realizar algumas simulações, entre as quais uma simulação de crescimento urbano, desenvolvida por Denise de Oliveira sob a orientação de Rafael H. Bordini e Rômulo Krafta, professor do departamento de arquitetura da UFRGS. Alguns destes resultados podem ser vistos em (KRAFTA; OLIVEIRA; BORDINI, 2002) e (BORDINI et al., 2004). Outra simulação realizada foi a simulação de um processo de compra em um mercado virtual, apresentada em (PACHECO; OKUYAMA; DIAS, 2004) por Oscar G. Calcin, Fabio Y. Okuyama e Aurélio M. Dias.

Com a descontinuação do AgentSpeak(XL), foi desenvolvido um novo interpretador para a linguagem AgentSpeak(L) chamado ***Jason*** (BORDINI; HÜBNER; WOOL-

DRIDGE, 2007). O ***Jason*** desenvolvido em JAVA por Jomi F. Hübner e Rafael H. Bordini, possui várias funcionalidades que estendem a linguagem AgentSpeak(L), esta linguagem estendida é referida pelos autores simplesmente como AgentSpeak. Atualmente na plataforma MAS-SOC, faz-se uso do interpretador ***Jason***, conforme descrito em (BORDINI et al., 2005).

O passo seguinte no desenvolvimento da plataforma MAS-SOC foi a inserção de estruturas para distribuição espacial de normas, apresentado em (OKUYAMA; BORDINI; ROCHA COSTA, 2007a). A idéia de distribuir espacialmente as normas foi aprofundado em uma infraestrutura normativa em (OKUYAMA; BORDINI; ROCHA COSTA, 2007b). Posteriormente, esta infraestrutura normativa foi apresentada como um meio de conectar as especificações de ambientes e organizações, conforme apresentado em (OKUYAMA; BORDINI; ROCHA COSTA, 2007c).

1.3.1 Trabalhos Complementares

O trabalho desenvolvido por Maíra Rodrigues durante seu mestrado sob orientação do professor Antônio Carlos da Rocha Costa, apresentou uma abordagem na qual os valores de troca propostos por Piaget, poderiam ser utilizados para fomentar as interações em um SMA. Em (RODRIGUES; ROCHA COSTA; BORDINI, 2003; RODRIGUES, 2003) é apresentado um modelo de valores de troca para fomentar interações qualitativas entre agentes. Este consiste de um mecanismo de deliberação social e da especificação de estruturas para o armazenamento e manipulação de valores de troca. No sistema proposto de valores, o agente deve, a cada interação em que este participa, armazenar quatro tipos de valores, que seriam satisfação, renúncia, reconhecimento e recompensa. Os valores resultantes destas trocas podem ser vistos como *valores morais*, compreendendo *débitos morais* (obrigação em realizar um serviço em contrapartida a um serviço recebido) e *créditos morais* (direito de requisitar a execução serviços em contrapartida a serviços já prestados). Então no decorrer de várias interações o agente irá acumular estes valores, para cada agente com o qual ele realizou interações utilizando estes valores na decisão de suas próximas interações. Desta forma, os débitos e créditos dos agentes irão fazer com que os agentes realizem interações, prevenindo que as interações cessem e que a organização entre em colapso.

Dando continuidade ao trabalho de Maíra Rodrigues, a tese desenvolvida por Márcia H. Islabão Franco, sob orientação do professor Antônio Carlos da Rocha Costa e coorientação do professor Helder Coelho apresenta um mecanismo de interação de alto nível que concentra diversas áreas, entre as quais teorias utilitaristas, psicologia e ciências sociais. O mecanismo possibilita aos agentes avaliarem suas interações e assim escolherem seus futuros parceiros, conforme a teoria dos valores de troca de Piaget. Neste mecanismo, são definidos e agregados protocolos de argumentação e meios para avaliação de trocas. Assim, a avaliação de trocas permite a escolha de estratégias para interação que auxiliam o agente na tomada de decisão. Além disso, os objetivos do agente são considerados para a seleção de parceiros e futuros parceiros para a realização de novas interações. Desta forma, o uso deste mecanismo, conduziria à obtenção de uma sociedade favorável à trocas equilibradas. Este mecanismo de interação, se agregado a plataforma de simulação social poderia produzir situações muito interessantes, já que aborda aspectos interessantes da tomada de decisão dos agentes, que não são abordados nesta tese. Mais informações podem ser encontradas em (FRANCO; COSTA, 2007).

Ambos os trabalhos acima mencionados tratam da questão da interação entre agentes, aspecto não abordado neste trabalho. Desta forma acredita-se que estes trabalhos com-

plementariam esta tese, que tem como uso básico as simulações sociais, ao tratarem do importante aspecto das interações multiagente. Ao incentivar as trocas entre os agentes e propiciar um ambiente social onde os agentes são *motivados* a interagir entre si, estes mecanismos de interação favorecem a ocorrência da *emergência* e outros fenômenos sociais que são objetos de interesse para a simulação social.

Em outro trabalho complementar, desenvolvido em conjunto com a professora Renata Vieira, sobre a possibilidade de definir ambientes multiagentes através de ontologias. Para tanto, foi desenvolvido um protótipo de ontologia de topo da qual poderiam ser derivadas criando ontologias especificas para cada ambiente. Nesta ontologia estariam presentes os dados necessários que definem um ambiente ELMS, permitindo que este ambiente pudesse ser simulado, além da possibilidade de se consultar a ontologia como tal, a fim de se obter informações sobre o ambiente. Mais informações podem ser encontradas em (OKUYAMA et al., 2006)

1.4 Estrutura do Texto

No Capítulo 2, é apresentado brevemente uma revisão bibliográfica com os conceitos de SMA que guiaram o desenvolvimento deste trabalho. São apresentados conceitos relacionados aos SMA, simulação social e organizações multiagente.

No Capítulo 3, é revisada a descrição de ambientes em ELMS, descrita inicialmente na dissertação de mestrado (OKUYAMA, 2003). Os conceitos relacionados à modelagem de ambientes e construções da linguagem são brevemente apresentados, juntamente com as extensões adicionadas na descrição do ambiente, que são explicitadas na seção 3.6.

No Capítulo 4, é apresentada a infraestrutura normativa desenvolvida para a modelagem de simulações baseadas em SMA, são abordados os *objetos normativos*, *espaços normativos*, a contextualização espacial de normas, *supervisores de normas*, a linguagem para especificação das normas, a biblioteca de planos relacionados a normas e algumas implicações do uso desta infraestrutura.

No Capítulo 5, é apresentado o novo modelo MAS-SOC, abordando a forma como os componentes interagem possibilitando a criação de simulações baseadas em SMA.

No Capítulo 6, são apresentados cenários modelados com o uso do modelo MAS-SOC.

Finalmente, no Capítulo 7, são destacadas algumas considerações sobre o trabalho.

2 SISTEMAS MULTIAGENTES

De acordo com (WOOLDRIDGE, 2002, pág. 3), um Sistema Multiagente (SMA), é composto por diversos agentes que interagem entre si, onde cada agente pode possuir objetivos e motivações distintas. Para que estes agentes, possam interagir satisfatoriamente, estes devem ser capazes de se coordenar, cooperar e negociar, possibilitando que tanto a *sociedade* de agentes quantos os agentes individualmente possam atingir seus objetivos. Assim, os SMA baseiam-se na idéia de que um comportamento *inteligente* pode ser alcançado através das interações de seus múltiplos agentes.

Apesar da possibilidade de um sistema de agentes distribuídos ter um equivalente centralizado, que otimizado pode ser mais eficiente que um distribuído, os sistemas distribuídos são, na maioria das vezes, muito mais simples para a compreensão e fáceis de desenvolver (HUHNS; LARRY; STEPHENS, 1999), especialmente quando a resolução do problema é de natureza distribuída.

Ao contrário das abordagens da IA tradicional, onde a metáfora da inteligência é baseada em um *comportamento individual humano* com ênfase na representação de conhecimentos e métodos de inferência, a metáfora usada na IAD é baseada em *comportamento social* com ênfase nas ações e interações. Um comportamento social "inteligente" pode surgir de membros "inteligentes" da sociedade, chamados agentes cognitivos, ou de membros "não-inteligentes" da sociedade, chamados agentes reativos (SICHMAN; DEMAZEAU; BOISSIER, 1992).

Os problemas tratados em SMA, diferentemente da Resolução Distribuída de Problemas (RDP), tratam de agentes que trabalham de forma autônoma e cada qual em seus objetivos particulares. Estes objetivos podem interagir, complementando-se ou contrapondo-se. Porém mesmo que não haja interação direta entre seus objetivos, existirá alguma forma de interação entre os agentes, visto que estes encontram-se em um ambiente compartilhado.

Algumas das razões para o grande interesse em SMA seriam, segundo (JENNINGS; SYCARA; WOOLDRIDGE, 1998):

- capacidade de prover robustez e eficiência;
- possibilitar inter operação com sistemas legados (antigos);
- capacidade de resolução de problemas onde dados, conhecimento e controle podem estar distribuídos.

Na área de SMA, busca-se criar sistemas onde diversos agentes possam interagir para que cada agente atinja seus objetivos. Em (BORDINI; VIEIRA; MOREIRA, 2001), os autores afirmam que o enfoque principal dos SMA é prover mecanismos para criação

de sistemas computacionais a partir de entidade de software autônomas, denominadas agentes, que interagem através de um ambiente compartilhado por todos os agentes de uma sociedade e sobre o qual atuam, alterando seu estado. Como os agentes possuem um conjunto específico e limitado de capacidades, freqüentemente os agentes precisam interagir para atingirem seus objetivos. Assim, é possível para os projetistas de sistemas computacionais a criação de sistemas computacionais de forma naturalmente distribuída e *bottom-up*.

Assim, é comum classificar um SMA como *reativo* ou *cognitivo*, de acordo com as características dos agentes que os compõem.

Os SMA reativos baseiam-se na idéia de que o comportamento inteligente deve emergir da interação de diversos agentes simples. Normalmente, um sistema reativo apresenta um grande número de agentes, podendo ser baseados no comportamento de colônias de insetos. Algumas das principais características dos SMA reativos, citados em (ÁLVARES; SICHMAN, 1997) são:

- Não há representação explícita de conhecimento: o conhecimento do agente é implícito e se manifesta através de seu comportamento;
- Não há representação do ambiente: o seu comportamento se baseia no que é percebido a cada instante no ambiente mas sem uma representação explícita dele;
- Não há memória das ações: os agentes reativos não mantém um histórico de suas ações, de forma que uma ação passada não exerce nenhuma influência sobre suas ações futuras, excetuando as mudanças percebidas no ambiente causadas por ações passadas sobre o mesmo;
- Organização etológica: a forma de organização dos agentes reativos é similar a sociedades de insetos, mais simples se comparada à organização social dos sistemas cognitivos;
- Grande número de membros: os SMA reativos têm, em geral, um grande número de agentes, da ordem de dezenas, centenas ou mesmo milhões de agentes.

Nos SMA cognitivos o comportamento inteligente também deve ter origem nas interações dos agentes, porém devido a maior complexidade destes agentes e o fato de que os SMA cognitivos agrupam usualmente um menor número de agentes, há uma maior responsabilidade do agente em originar o comportamento inteligente. No desenvolvimento destes, utiliza-se uma noção mais forte de agência, onde os agentes apresentam de maneira mais clara características como autonomia e capacidades de deliberação. As características dos SMA apresentadas em (JENNINGS; SYCARA; WOOLDRIDGE, 1998) são:

- cada agente tem capacidades e informações incompletas para solução do problema, de forma que cada agente tem um ponto de vista limitado do problema;
- não há um sistema de controle global;
- os dados são descentralizados;
- a computação é assíncrona.

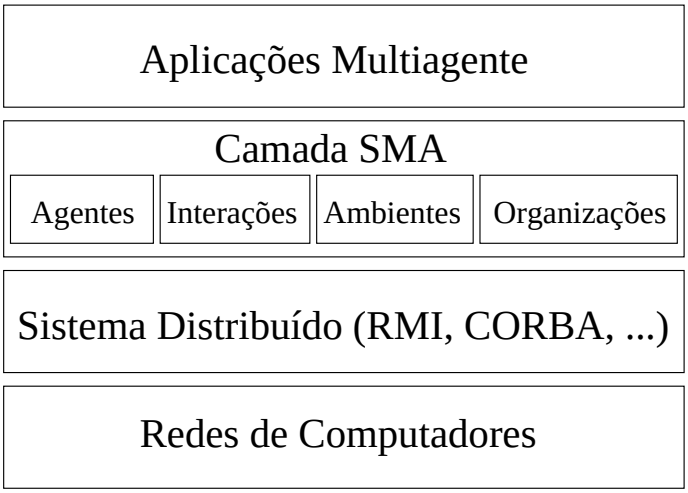

Figura 2.1: Estrutura de um SMA

Segundo Demazeau (1995), os componentes básicos de um SMA seriam os agentes, as interações, o ambiente e as organizações. Neste modelo os agentes poderiam ser desde simples autômatos a complexos sistemas baseados em conhecimento. O ambiente seria dependente do domínio da aplicação, porém na maioria dos casos seriam ambientes espaciais. As estruturas de interação compreenderiam desde interações físicas até diálogos com atos de fala. Ainda nesta visão, as organizações poderiam ser baseadas em estudos biológicos ou guiadas pelas leis sociais baseadas nos estudos de ciências sociais. Desta forma, a construção de um SMA deve ser baseada nos seguintes pressupostos:

Equação declarativa: um SMA é composto de vários agentes, um conjunto de interações possíveis e pelo menos uma organização (*SMA = Agentes + Ambiente + Interações + Organização*);

Equação funcional: a funcionalidade de um SMA é a soma das funções individuais de cada um dos agentes e a função coletiva, obtida através das interações, ou seja *Funcionalidade(MAS)* $= \sum$ *Funcionalidade(Agentes)*$+$ *Funcionalidade Coletiva*;

Princípio da recursão: um SMA pode ser considerado, em um nível de abstração mais elevado, como sendo um agente ($Agente = Agente|SMA$).

Seguindo estas idéias, dado um problema ou uma simulação a realizar, o usuário selecionaria os agentes, um modelo de ambiente, modelos de interações e organizações a serem instanciados, gerando desta forma um sistema multiagente para a resolução de seu problema. A figura 2.1, adaptada de Demazeau (1995), mostra esta concepção de SMA onde temos as entidades SMA sendo instanciadas em uma aplicação para atender o usuário.

2.1 Simulação Social e os SMA

Os SMA têm grande influência das ciências sociais, da mesma forma, simulações baseadas em SMA têm crescente importância para simulações sociais. A simulação social é uma das áreas de pesquisa onde SMA é a abordagem mais adequada para sua realização, por ser um problema de resolução descentralizada e os agentes envolvidos serem basicamente autônomos.

No campo da simulação social, objetivos variam em função das disciplinas de interesse. Objetivos mais práticos prevalecem em ciências econômicas, gerenciamento e política e a ciência das organizações. Outras disciplinas como arqueologia e antropologia são claramente direcionadas ao crescente conhecimento científico através de formulação e testes de modelos interpretativos de fenômenos existentes através da reconstrução computacional (CONTE; GILBERT; SICHMAN, 1998). A maior razão para o crescimento de interesse dos cientistas sociais na simulação computacional é o seu potencial para descoberta e formalização. Cientistas sociais podem construir modelos simples com foco em algum aspecto específico do mundo social e descobrir as conseqüências de suas teorias na "sociedade artificial" que construíram (GILBERT; TROITZSCH, 1999).

A maior preocupação da simulação social é o estudo experimental e a modelagem de conjuntos de efeitos imprevisíveis de uma população de agentes em um ambiente comum (CASTELFRANCHI, 1998). Enquanto a simulação computacional consiste da análise de propriedades de modelos teóricos com o objetivo de explicar ou prever fenômenos naturais (FERBER, 1999, pág. 35). A simulação computacional de fenômenos social é um campo de pesquisa promissor, que se encontra na intersecção entre ciências sociais, matemática e ciência da computação. Estas simulações computacionais têm aplicação nas diversas ciências sociais, abrangendo da sociologia à economia, da psicologia social às teorias de organização política, e da demografia à antropologia e arqueologia (CONTE; GILBERT; SICHMAN, 1998). Atualmente, a simulação computacional de processos e fenômenos sociais pode ser considerada uma área de pesquisa bem estabelecida.

Nas últimas décadas a simulação computacional foi capaz de se beneficiar de um grande número de facilidades como linguagens de alto nível e desenvolvimento de sistemas inteligentes. Neste contexto, a IAD, e em particular os SMA, forneceram as arquiteturas para desenvolvimento de agentes relativamente autônomos, fundamentais neste tipo de simulação. Os SMA trouxeram uma nova solução ao conceito de modelagem e simulação de ciências ambientais, oferecendo a possibilidade de representar diretamente indivíduos, seu comportamento e suas interações. A simulação com SMA se baseia na idéia que é possível representar de maneira computadorizada o comportamento de entidades que são ativas no mundo, e através disto representar um fenômeno como o produto de interações de um conjunto de agentes que possuem autonomia operacional (FERBER, 1999).

A abordagem baseada em agentes aumentou as potencialidades da simulação computacional como uma ferramenta para teorização sobre questões científicas sociais. Se o campo de SMA pode ser caracterizado pelo estudo de sociedades de agentes autônomos artificiais, a simulação social baseada em agentes pode ser definida como o estudo das sociedades artificiais de agentes autônomos (CONTE; GILBERT; SICHMAN, 1998).

Os SMA e a simulação social diferem em termos de formalismos utilizados, os SMA baseiam-se em lógica e IA enquanto a simulação social baseia-se na matemática (CONTE; GILBERT; SICHMAN, 1998). Apesar da teoria das decisões e teoria dos jogos terem influenciado ambos, existem diferenças teóricas entre os dois campos. Assim, ainda segundo (CONTE; GILBERT; SICHMAN, 1998) os SMA herdaram da IA e das ciências cognitivas um grande embasamento teórico o que resultou em:

- longa experiência com projeto e implementação de arquiteturas integradas em oposição a autômatos elementares;

- forte ênfase no agente como um todo em oposição a apenas ações;

- grande atenção ao processo de construção de planos, não apenas a tomada de decisão e escolha;

- familiaridade com a normalização e implementação de estados mentais e comportamentais de agentes;

- tendência de prover o agente social com capacidades específicas para responder pedidos e tarefas sociais ao invés de modelar processos sociais como meras propriedades emergentes da interação dos agentes.

A área de simulação social tem maior influência das ciências sociais em relação aos SMA. Os fatores que contribuíram para o desenvolvimento da área foram:

- tendência de uso da simulação computacional para a testes de hipóteses teóricas ao invés de eficiência computacional do sistema;

- maior familiaridade com a interpretação de fenômenos sociais da vida real;

- produção de grandes quantidades de dados relativos à populações artificiais de grande escala.

Todas estas características resultaram na consolidação da reputação da metodologia científica da simulação computacional, reduzindo a característica *toy-world* de suas aplicações.

2.2 Organizações

Tanto para a simulação social e simulação baseada em SMA, as organizações e o estudo das organizações são aspectos importantes, seja para o estudo científico das organizações ou indiretamente na busca das melhores estruturas organizacionais ou na busca de melhores resultados ou performance.

Uma organização pode ser definida como um conjunto de relacionamentos entre componentes ou indivíduos que compõem uma unidade, dotada com qualidades não apresentadas no nível de componentes ou indivíduos (FERBER, 1999, pág. 88).

Segundo Carley e Gasser, não existe um grande consenso sobre o conceito de organizações; estes autores listam em (CARLEY; GASSER, 1999) algumas características que normalmente estão associadas a este conceito:

- constituídas de múltiplos agentes (humanos ou artificiais);

- estão associados à tecnologias de resolução de problemas de grande escala;

- estão comprometidas com uma ou mais tarefas; organizações são sistemas altamente ativos;

- capaz de afetar e ser afetada pelo ambiente;

- possui conhecimento, cultura, memórias, história e capacidades distintas a um agente simples.

As características que tornam o estudo das organizações uma tarefa complexa e bastante pesquisada é que as organizações são sistemas complexos, dinâmicos, adaptativos que evoluem.

O estudo de organizações dentro dos SMA pode ser classificado, de acordo com Ferber (1999), da seguinte maneira:

- **nível micro-social:** onde há interesse essencialmente nas interações entre agentes e nas várias formas de relacionamento que existem entre pequenas quantidades de agentes;
- **nível de grupos:** onde há interesse nas estruturas intermediárias que compõem organizações completas. Enfocando nas diferenciações de papéis e atividades dos agentes, emergência de estruturas organizacionais entre agentes e problemas gerais de agregação de múltiplos agentes na constituição de organizações;
- **nível de sociedades globais:** onde o interesse é concentrado na dinâmica de um grande número de agentes, junto à estrutura geral do sistema e sua evolução.

Alguns conceitos ou fenômenos importantes abordados no estudo de organizações seriam:

Emergência: fenômeno no qual estruturas não-existentes no nível micro "emergem" no nível macro, através da interação das estruturas micro.

Emergência cognitiva: fenômeno que ocorre quando os agentes tomam consciência através de uma "conceitualização" do fenômeno da emergência, influenciando o comportamento destes agentes (CASTELFRANCHI, 1998).

Imergência: processo através do qual a estrutura emergente no nível macro ocasiona mudanças no nível micro, remodelando o comportamento das estruturas micro (agentes) (CASTELFRANCHI, 1998).

Emergência de segunda ordem: processo através do qual as mudanças realizadas no nível micro através da imergência ocasionam alteram as estruturas emergentes já existentes.

O comportamento organizacional é resultado das interações entre uma variedade de agentes adaptativos (humanos ou artificiais), estruturação emergente em resposta a processos não-lineares e interações entre um grande número de outros fatores (PRIETULA; CARLEY; GASSER, 1998, pág. xiv). Por esta razão, a análise computacional torna-se uma importante ferramenta para construção de teorias pois permite gerar um conjunto de proposições teóricas mesmo quando existem interações complexas de diversos fatores.

Em (FERBER; MICHEL; BÁEZ-BARRANCO, 2005), os autores afirmam que o nível organizacional descreve os relacionamentos experados e padrões de atividade que podem ocorrer no nível dos agentes e assim definir os limites e potencialidade que constituem o espaço onde os agentes agem. Além disso, (COUTINHO; SICHMAN; BOISSIER, 2006) afirma que uma organização ou coletividade apresenta padrões e estruturas estáveis de atividade conjunta sendo o inverso também verdadeiro, especialmente quando os padrões e estruturas de atividade conjunta são projetados, algo bastante comum em diversas abordagens para modelagem de SMA.

Em (COUTINHO; SICHMAN; BOISSIER, 2006), os autores citam quatro dimensões que são utilizadas na modelagem de organizações, baseados na análises de diversas abordagens e ferramentas para modelagem de organizações. Estas dimensões seriam:

- **Estrutural:** grupos, papéis e suas relação sociais;
- **Funcional:** objetivos, decomposição de tarefas e a relação entre tarefas;
- **Normativa:** normas (deônticas) da organização;
- **Dialógica:** diálogos, cenas e protocolos.

Estas dimensões podem ser utilizadas para classificar as abordagens de organizações e suas características.

2.2.1 Metodologia Gaia

Uma das primeiras abordagens para modelagem do nível organizacional de sistemas multiagente é a metodologia Gaia (WOOLDRIDGE; JENNINGS; KINNY, 2000). O Gaia é também baseado em modelos organizacionais que objetivam o projeto de sistemas multiagentes que podem ser compostos de modelos e teorias de agentes heterogêneos. Todavia, o Gaia é uma metodologia para o projeto e análise de sistemas orientados à agente, não sendo uma plataforma para o desenvolvimento de sistemas.

A metodologia Gaia consiste da construção de cinco modelos a partir da "especificação de requisitos" (*requirement statement*). Estes modelos definem respectivamente, os papéis, interações, agentes, serviços e "conhecidos" (*acquaintances*). O primeiro modelo define os possíveis papéis no sistema, onde um papel é definido por um conjunto de responsabilidades, permissões, atividades e protocolos. O modelo de interação define as dependências e relacionamentos entre os vários papéis na organização. O modelo de "conhecidos" define os possíveis elos de comunicação entre os agentes. É uma metodologia bastante completa e genérica, porém não contempla aspectos internos de agentes e a modelagem de sistemas abertos.

2.2.2 $\mathcal{M}$OISE$^+$ – Modelo de Organização para Sistemas Multiagentes

Em (HÜBNER; SICHMAN; BOISSIER, 2002), é apresentado o $\mathcal{M}$OISE$^+$ (Model of Oganization for multI-agent SystEms) que é um modelo de organização bastante completo para SMA, sendo uma de suas características principais a possibilidade de reorganizar dinamicamente a organização. Conforme a classificação de (COUTINHO; SICHMAN; BOISSIER, 2006), este modelo abrangeria as dimensões estrutural, funcional e normativa. No $\mathcal{M}$OISE$^+$ a organização é descrita em três modelos independentes: especificação estrutural, especificação funcional da organização e especificação deôntica (obrigações dos agentes que participam da organização). Nesta seção são apresentadas resumidamente as principais características do $\mathcal{M}$OISE$^+$ como modelo para definição de organizações; características e funcionalidades de reorganização não são abordadas.

Especificação Estrutural

No $\mathcal{M}$OISE$^+$, a especificação estrutural da organização é feita através da definição dos papéis, relacionamentos entre os papéis e os grupos, correspondendo respectivamente aos níveis individual, social e coletivo. Além disso, a especificação da estrutura é enriquecida com o uso de conceitos como herança, compatibilidade de papéis, cardinalidade e subgrupos. A partir da especificação dos papéis, que relacionam os agentes à organização, são definidos os relacionamentos entre os papéis quais podem ser de três tipos:

- Conhecimento (acq): indica que um agente tem permissão de conhecer outro agente;

- Comunicação (com): indica que um agente tem permissão para se comunicar com outro;

- Autoridade (aut): indica que um agente tem autoridade sobre outro.

Além dos relacionamentos entre os papéis, é possível também especificar a compatibilidade entre papéis, ou seja quais papéis podem ser desempenhados por um agente que já possui outro papel em uma mesma organização. Finalmente no nível coletivo (grupos) são especificados para cada grupo os papéis que podem ser assumidos no grupo, tipos de sub-grupos que podem ser criados, ligações válidas dentro do grupo, compatibilidade intra-grupo e cardinalidade dos papéis do grupo.

Especificação Funcional

A especificação funcional é feita através da especificação de esquemas sociais que são compostos de *missões* (que seriam conjuntos de objetivos coerentes com os quais um agente pode se comprometer) que visam atingir uma meta global. Desta forma, um esquema social é formado a partir da decomposição de um objetivo em uma árvore onde a raiz é a meta global do esquema e as responsabilidades para os sub-objetivos são distribuídos em missões. Uma meta global é um estado de mundo desejado pelo SMA, na definição das metas, cada meta possui três atributos que indicam a *satisfabilidade* (satisfied/unsatisfied/impossible), nível de *alocação* (committed/uncommitted) e nível de *ativação* (permitted/forbidden); além disso, para cada meta pode ser associada uma probabilidade de sucesso. Neste contexto, uma missão é o conjunto de metas compatíveis entre si, que podem ser atribuídas a um agente, de acordo com seu papel na organização. Na organização de metas e sub-metas, é possível três tipos de relacionamentos:

- seqüência: significa que a meta será satisfeita quando o conjunto de sub-metas for realizado;

- escolha: a meta será satisfeita quando uma das sub-metas for satisfeita;

- paralelismo: significa que a meta será satisfeita quando o conjunto de sub-metas for realizado, porém sua sub-metas podem ser trabalhadas paralelamente.

Através destes relacionamentos e do uso dos atributos, a ordem de execução das metas é controlada a fim de se obter o estado de mundo desejado. Nos casos onde exista mais de uma opção, pode ser explicitada a preferência das missões, relacionando uma missão como tendo preferência sobre outra.

Especificação Deôntica

No $\mathcal{M}$OISE$^+$ a especificação estrutural e funcional de uma organização podem ser feitas de maneira independente. Porém estas são relacionadas através do modelo deôntico que especifica as permissões e obrigações que um papel deve se comprometer. Tanto as permissões quanto obrigações são compostas por um papel, uma missão e uma restrição temporal. Uma permissão determina que um agente que assume determinado papel pode se comprometer com a missão, dada uma determinada restrição temporal. Da mesma forma, uma obrigação determina que todo agente que assumiu determinado papel tem a obrigação de cumprir a missão nos períodos de tempo definidos na restrição temporal.

2.2.3 OMNI – Organizational Model for Normative Institutions

Esta seção é apresenta resumidamente as principais características do OMNI, a baseados em (VÁZQUEZ-SALCEDA; DIGNUM; DIGNUM, 2005; DIGNUM; VÁZQUEZ-SALCEDA; DIGNUM, 2005). O OMNI é uma abordagem para modelagem de SMA, que possui características para especificar diversos tipos de sistemas. A estrutura do OMNI se baseia em três dimensões: normativa, organizacional e ontológica, sendo cada destas dimensões divididas em três níveis:

Nível Abstrato: onde os estatutos da organização a ser modelado são definidos em um alto nível de abstração;

Nível Concreto: os valores do nível anterior são definidos através de normas, regras, papéis, marcos (*landmarks*) e conceitos ontológicos concretos;

Nível de Implementação: onde o projeto das dimensões normativa e organizacional são implementados em uma determinada arquitetura multiagente, incluindo mecanismos para implementação dos papéis e execução das normas.

Dimensão da Organização

Na dimensão da organização, no nível abstrato, são descritos os objetivos da organização como sistema social, definidos como uma lista de objetivos externamente observáveis da organização, ou seja, os estados de mundo desejados na sociedade.

No nível concreto são definidos a estrutura social e estrutura de interações. A estrutura social é definido basicamente por papéis, grupos, uma hierarquia de papéis e as relações de dependência de papéis. Estes elementos são compostos por:

Papel: identificador, conjunto de objetivos e sub-objetivos, direitos, normas e tipo do papel;

Grupos: identificador, conjunto de papéis, regras do grupo;

Hierarquia de papéis: conjunto de relacionamentos formados cada um por um papel de origem, destino e o tipo de hierarquia existente entre estes papéis;

Dependência entre papéis: conjunto de relacionamentos formados por um papel de origem, destino e o tipo de dependência existente.

A estrutura de interação é definida por cenas relevantes que seguem scripts abstratos pré-definidos. Um *script de cena* descreve a cena através dos papéis envolvidos, os resultados esperados e as normas que regulam a interação. Na definição de cenas, visando atingir os objetivos da cena podem ser utilizados *landmarks*, que podem indicar estados intermediários ou sub-objetivos a serem alcançado a fim de atingir o objetivo da cena. O conjunto de cenas é estruturado a fim de indicar a seqüência de cenas que levam a organização a atingir seu objetivo.

No nível de implementação, a modelagem é feita em duas partes; o modelo social e o modelo de interação. O modelo social é descrito em termos de compromisso relacionados à execução dos papéis por instâncias de agentes. Este compromisso é dado por *contratos sociais*, que identificam o agente, o papel e um conjunto de cláusulas contratuais que podem incluir o período temporal ou condições, condições envolvendo a execução do papel, além das sanções que serão tomadas no caso de violação das normas.

No modelo de interação são instanciados os *scripts de cena* abstratos definidos no nível concreto.

Dimensão Normativa

Assim como a dimensão organizacional, a dimensão normativa também é definida em três níveis. O nível abstrato é composto por um conjunto de *valores* que determinam os conceitos que serão utilizados para determinar o valor ou utilidade das situações. Ainda no nível abstrato, cada um dos valores é traduzido em um conjunto de normas deônticas abstratas.

No nível concreto, as normas abstratas são traduzidas em ações e conceitos que podem ser manipulados por tais organizações. Para isso, as normas abstratas são refinadas em normas concretas iterativamente até serem transformadas em regras, violações e sanções que as implementam. No nível das normas, para cada norma abstrata são definidas normas concretas que cumprem a norma abstrata. Ainda no nível concreto, as normas são transformadas em regras, compostas por expressões que referenciam ações e indicam possíveis violações das regras. Além disso, são definidas sanções relacionadas a cada violações. Cada sanção referência uma violação, pré-condições, a sanção efetiva, efeitos colaterais e os agentes que devem executá-la.

No nível de implementação, é possível implementar as regras do nível de regras. A primeira é criar um interpretador de regras que todos os agentes que entrem na organização devem incorporar. Outra opção é traduzir as regras em protocolos a serem incluídos nos contratos de interação.

Dimensão Ontológica

A dimensão ontológica, visa tratar o problema da compreensão mútua a fim de atingir coordenação e colaboração em ambientes abertos. No OMNI, a dimensão ontológica, descreve tanto o conteúdo quanto a linguagem de comunicação, em três diferentes níveis de abstração. No nível abstrato, a ontologia modelo pode ser vista como uma meta-ontologia que define todos os conceitos da abordagem, tais como normas, regras, papéis, grupos, violações, sanções e marcos (*landmarks*). Os aspectos de conteúdo das comunicações e conhecimento de domínio são especificados em *ontologias de domínio*, onde conceitos abstratos são iterativamente refinados em conceitos mais concretos. A *ontologia de domínio concreto*, inclui todos os predicados e elementos que aparecem durante o projeto da estrutura organizacional e normativa. Enquanto na *ontologia procedural de domínio* contém os termos do domínio que serão utilizados na implementação do sistema.

2.2.4 Instituições Eletrônicas

Outra abordagem para modelagem de organizações que tem recebido grande atenção são as Instituições Eletrônicas (*Electronic Institutions*) (GARCIA-CAMINO; NORIEGA; RODRÍGUEZ-AGUILAR, 2005). De maneira geral, o funcionamento interno de uma instituição eletrônica é similar ao de uma máquina de estados onde cada estado é chamado de “cena”. Em cada cena são especificados o conjunto de papéis que podem participar e um *protocolo de conversação* que os agentes devem seguir quando interagir na cena. Para percorrer a série de cenas que constituem a operação de uma instituição eletrônica, os agentes devem realizar uma seqüência de ações em cada cena e também se comprometer a realizar ações específicas em determinadas cenas.

As Instituições Eletrônicas (IE) fornecem abstrações baseadas em organizações humanas onde humanos e agentes inteligentes com diferentes papéis organizacionais interagem para cumprir metas individuais e organizacionais. A seguir, são apresentados os principais conceitos relacionados a instituições eletrônicas, com base em (SIERRA et al., 2004).

As IE são voltadas para regular sistemas abertos onde agentes auto-interessados necessitam ser regulados. Os principais conceitos envolvidos são:

Agentes e papéis: os agentes interagem através da IE, através de atos de fala. Todos os agentes registrados em uma IE devem adotar pelo menos um papel. São definidos agentes e papéis institucionais para atingir e garantir as regras da instituição. Enquanto os papéis e agentes não-institucionais são os agentes que estão sujeitos às regras da IE.

Modelo Dialógico: Alguns aspectos de uma IE, tais como objetos do ambiente e a linguagem utilizada para comunicações são fixas, constituindo o contexto ou modelo de interação entre agentes. Cada IE estabelece os atos de fala aceitáveis através da definição da ontologia e da linguagem comum para comunicação e representação do conhecimento que são incluídas no modelo dialógico (*dialogical framework*).

Cena: Interações entre os papéis são articuladas através de reuniões de grupos de agentes e uso de protocolos definidos, referidas como cenas. As cenas definem as interações (diálogos) possíveis entre os agentes.

Estrutura Performativa: As cenas podem ser conectadas, compondo um fluxograma, referido como estrutura performativa, que define as condições para entrada e saída das cenas.

Regras Normativas: Os agentes e papéis devem assumir compromissos na IE que impõem obrigações, restrições ou ações dialógicas dos agentes nas cenas onde estão interagindo ou vão interagir posteriormente. De maneira geral, são obrigações e proibições que o agente deve cumprir em determinada cena.

Em (SIERRA et al., 2004) afirma-se *"ao contrário das abordagens tradicionais que permitem aos agentes interagir livremente com seus pares via camada de comunicação, nossa realização computacional de uma IE precisa ser encarada como uma camada social que fica entre os participantes externos e a camada de comunicação validando ou rejeitando suas ações."*

Para a modelagem de uma IE é fornecida um conjunto de ferramentas, referido como IDE-eli (*Integrated Development Environment for Electronic Institutions*). Estas ferramentas suportam uma abordagem *top-down* para modelagem e são compostas por:

- Islander: Ferramenta gráfica para especificação de regras e protocolos em uma IE;
- AMELI: plataforma de software para execução de Instituições eletrônicas;
- aBUILDER: ferramenta para desenvolvimento de agentes;
- SIMDEI: ferramenta de simulação para animar e analisar especificações do ISLANDER.

2.3 Sistemas Multiagentes Normativos

Uma das características principais associadas relacionadas aos conceitos de agência, presentes nas diversas definições de agentes é a autonomia. Para se ter agentes autônomos interagindo, cada qual com seu objetivo, assim como em sociedades humanas, é necessário algum tipo de normatização, seja esta implícita ou explícita. Neste contexto, os

SMA normativos (*Normative Multi-Agent Systems*) de acordo com (BOELLA; TORRE; VERHAGEN, 2007) são definidos da seguinte forma: *"um SMA normativo é um SMA juntamente com um sistema normativo onde os agentes podem decidir por seguir as regras explicitamente representadas, e por outro lado o sistema normativo especifica como e em qual extensão os agentes podem modificar as normas."* Neste contexto, ainda de acordo com (BOELLA; TORRE; VERHAGEN, 2007), o uso de normas é um elemento chave para a inteligência social humana e pode ser essencial também para agentes artificiais que colaboram com humanos.

Na área de SMA, o conceito de *norma* é algo bastante abrangente, o que implica em diferentes visões para este conceito. Entre os diversos conceitos de normas pode-se citar o encontrado em (LÓPEZ; LUCK, 2004), onde normas são definidas como mecanismos que uma sociedade, grupo ou organização utiliza para influenciar o comportamento dos agentes que a compõem. Outro conceito um pouco mais genérico, seria o de (CONTE; CASTELFRANCHI; DIGNUM, 1999), que define uma norma como uma obrigação de um dado conjunto de agentes para cumprir ou se abster de uma determinada ação. Em (BOELLA; TORRE; VERHAGEN, 2007), os autores se referem a uma norma como sendo um princípio de ação correta que relaciona membros de um grupo que serve para guiar, controlar ou regular o comportamento aceitável. Neste contexto, em (LOPEZ; LUCK; D'INVERNO, 2007), um agente normativo é definido como um agente cujo comportamento é parcialmente definido pelas normas, enquanto um agente autônomo decide por adotar uma norma com base em seus próprios objetivos e motivações.

Entre as diversas classificações de normas pode-se citar (DIGNUM, 2002), que afirma existirem normas informais, quando não há uma regulação das normas e não há sanções formais pela violação da norma, enquanto as normas formais são as incorporadas em leis ou regulamentos de instituições que regulam o comportamento de pessoas em uma sociedade. Além disso, (CONTE; CASTELFRANCHI; DIGNUM, 1999), refere uma norma como sendo externa quando os agentes sujeitos à ela não possuem representação mental correspondente à norma. Neste contexto, *normas abstratas* seriam normas que tentam capturam diferentes situações e por isso são *abstratas* de diversas formas, sendo as algumas das formas mais comuns de abstração citadas em (DIGNUM, 2002):

- Referência a uma ação abstrata que pode ser implementada de diversas formas;
- Uso de termos vagos e que dependem de outras definições;
- Abstração de aspectos temporais;
- Abstração agentes e/ou papéis;
- Referência à ações ou situações que não são (diretamente) controláveis ou verificáveis pelo agente ou sistema computacional.

Na literatura de SMA é comum o uso de normas sociais com a finalidade de coordenar atividades ou obter cooperação para realização de objetivos de um determinado grupo. Em (RUSSELL; NORVIG, 2003, pág. 452) afirma-se que uma das maneiras mais simples pela qual um grupo de agentes pode garantir uma concordância em um plano conjunto é a adoção de uma convenção antes de iniciar a atividade conjunta. Algumas convenções, adotadas de maneira bastante ampla, podendo ser consideradas normas ou leis sociais. Integrando normas e inteligência individual, sistemas multiagentes normativos fornecem um modelo promissor para agentes humanos e artificiais obterem coordenação, cooperação, tomada de decisão em grupo, sociedades reguladas, instituições eletrônicas, sistemas

multiagentes seguros e outras possibilidades (BOELLA; TORRE; VERHAGEN, 2007). Além disso, as normas não apenas limitam a conduta de um agente, tornando-a mais uniforme e previsível, mas elas também fornecem novos comportamentos (e.g. pagar impostos, usar capacete em motos)(CONTE; FALCONE; SARTOR, 1999).

Entretanto, existem diversas questões relacionadas a implementação de SMA Normativos e simulações sociais baseadas em SMA, abaixo são citadas algumas destas, encontradas em (CONTE; CASTELFRANCHI; DIGNUM, 1999):

- Como os agentes adquirem normas? Em geral nos SMA, normas são tratadas como restrições pré-programadas, mas como tratar a aquisição de novas normas?
- Como e porque agentes autônomos se submetem a objetivos adquiridos através de normas?
- Como os agentes podem violar normas?

Estas questões se relacionam à autonomia do agente, que deveria ser capaz de raciocinar sobre adoção de uma norma, além de deliberar em cada situação sobre seguir ou violar uma norma. De acordo com (CONTE; CASTELFRANCHI; DIGNUM, 1999) a possibilidade de violar normas é crucial para resolver possíveis problemas de conflitos de normas, que frequentemente surgem entre tarefas associadas a diferentes papéis ou regras pertencentes a diferentes domínios de atividades. Ainda neste sentido, em (BOELLA; TORRE; VERHAGEN, 2007) os autores afirmam que nem todos os agentes comportam-se de acordo com as normas, e isto deve ser suportado pelo sistema; ou seja, as normas não são restrições duras, mas sim restrições suaves.

2.3.1 Modelo Normativo de Lopez e Luck

Um dos modelos mais completos de SMA Normativo, apresentado em (LÓPEZ; LUCK, 2004; LOPEZ; LUCK; D'INVERNO, 2007). Este modelo formaliza diversos aspectos de um SMA Normativo, desde a especificação formal de normas e agentes até normas de legislação, incluindo-se a formalização de como esta estrutura deve ser manipulada por um agente normativo. A seguir é apresentado resumidamente os principais aspectos do modelo com base em (LÓPEZ; LUCK, 2004; LOPEZ; LUCK; D'INVERNO, 2007).

Neste modelo, um agente autônomo é definido por um conjunto de objetivos, um conjunto de capacidades, um conjunto de motivações que representam suas preferências e um conjunto de crenças que representam sua visão do mundo. Além disso, cada objetivo possui uma importância, definida pelo agente.

Em relação às normas, cada norma é definida pelos seguintes conjuntos:

- objetivos normativos relacionados à norma;
- destinatários: agentes aos quais se aplica a norma;
- beneficiários da norma;
- contexto: conjunto de estados do ambiente onde a norma é válida;
- exceções: conjunto de estados do ambiente onde a norma não é válida;
- recompensas: conjunto de objetivos, disparados quando a norma é cumprida;

- punições: conjunto de objetivos, disparados quando a norma não é cumprida.

Os objetivos presentes no conjunto de recompensas e punições são objetivos a serem realizados por outros agentes específicos que têm como norma cumprir estes objetivos. Desta forma, o modelo possui meios para relacionar normas, ou seja, quando uma *norma primária* é cumprida, esta ativa uma *norma secundária* que deverá ser cumprida por um agente específico a fim de premiar o agente que executou corretamente uma norma. Da mesma forma acontece na violação de uma norma.

Neste contexto, um SMA Normativo é definido pelos seguintes conjuntos:

- membros: conjunto de agentes normativos;
- normas gerais;
- normas legislativas;
- normas para execução (*enforcing norms*);
- normas para premiação;
- ambiente: estado do ambiente.

Neste modelo alguns agentes desempenham papéis identificados como *autoridades* do sistema. Estes papéis são:

- legisladores: responsáveis por criar, modificar e extinguir normas;
- defensores (*defenders*): responsáveis por aplicar punições (executar *enforcing norms*) quando normas são violadas;
- promotores: responsáveis por executar normas de premiação (*reward norms*).

O modelo ainda inclui a formalização raciocínio normativo associado com as estruturas apresentadas nesta seção, maiores detalhes podem ser encontrados em (LOPEZ; LUCK; D'INVERNO, 2007).

2.4 Considerações Finais

A área de pesquisa de SMA é comumente divida entre SMA *reativos* e SMA *cognitivos*, porém em relação à modelagem de SMA, pode-se verificar abordagens *top-down* e outras *bottom-up*. Dentro das diferentes abordagens de SMA, pode-se verificar o enfoque dado para os agentes (*bottom-up*) ou para as organizações (*top-down*).

É comum nas abordagens *bottom-up* a premissa que o raciocínio individual dos agentes na busca de seus objetivos individuais irá naturalmente fazer com que o sistema como um todo atinja seus objetivos. Enquanto nas abordagens *top-down*, trabalha-se no nível das organizações, criando regras e padrões de comportamento para grupos de agentes de forma que a organização atinja seu objetivo, para que a partir disso os agentes e o sistema como um todo atinjam seus objetivos.

Nestas condições, é comum associar a *inteligência* de um SMA aos agentes ou à estrutura organizacional, deixando o ambiente como um elemento de menor importância, modelando-se o ambiente de forma a se encaixar aos agentes e organizações. Outro fato importante a se notar é que quando se refere à modelagem dos agentes, o aspecto levado

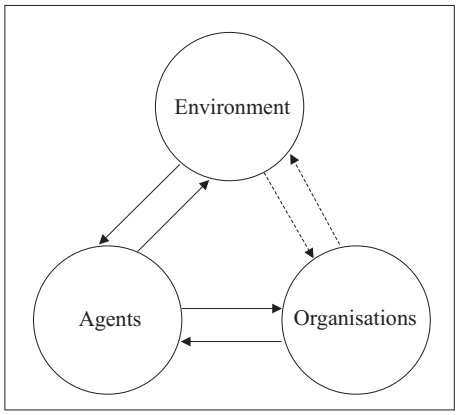

Figura 2.2: Componentes SMA

em consideração é apenas o raciocínio do agente, deixando de lado a ligação entre agentes e ambiente. No caso das abordagens *top-down*, é comum verificar que estas são definidas de maneira não-espacialmente localizada, ignorando o espaço onde esta existe e opera.

Assim como mostrado na Figura 2.2, na maioria dos casos os elementos que compõem um SMA: agentes, organizações, ambiente são elementos disjuntos, que interagem através de diversos mecanismos de interação (representados pelas setas). Outro fato também usualmente verificado, é que as estruturas de interação entre agentes e organizações são definidas claramente nas abordagens *top-down*, enquanto as interações entre ambiente e agentes são definidas como parte da infraestrutura do SMA, recebendo pouca importância na modelagem do SMA.

A linguagem ELMS, que é abordada no próximo capítulo, trata dos ambientes multi-agentes, assim como parte das interações agente-ambiente, que na Figura 2.2 é representada pelas setas contínuas. Enquanto a lacuna que pretende-se preencher com o modelo apresentado nesta tese é a lacuna existente entre ambientes e organizações, representada pelas setas pontilhadas. Desta forma, agregando-se ao modelo ELMS onde agentes e ambientes são especificados de maneira integrada, resultando em um modelo onde os elementos de um SMA são especificados e funcionam de maneira integrada.

Em relação às abordagens de organizações multiagente, abordadas na Seção 2.2 pode-se notar que os modelos não tratam da questão do ambiente. Outro aspecto levantado neste capítulo, na área de SMA Normativos, existem diversas pesquisas e abordagens em desenvolvimento, porém estas são extremamente formais, o que tornam sua implementação uma atividade complexa e pouco prática.

3 AMBIENTES MULTIAGENTES E O ELMS

De acordo com Wooldridge (1999), agentes são sistemas computacionais situados em um ambiente, e capazes de ações autônomas para realização dos objetivos para os quais foram projetados. Os agentes percebem e interagem entre si através de um ambiente, alterando o estado deste para alcançar estados onde seus objetivos sejam atingidos. Desta forma, ambientes são parte importante em um SMA, seja o mundo real, a Internet ou um ambiente virtual. Em algumas modelagens, ambiente também é considerado como parte importante na comunicação. Por exemplo, na comunicação falada, o falante cria ondas sonoras que irão se propagar pelo ar até chegar ao ouvinte, utilizando o ambiente como meio para realização da comunicação.

Nos sistemas multiagentes reativos, o ambiente tem um papel importante. Partindo do princípio que os agentes reativos não possuem memória, é apenas a percepção do ambiente que permite a eles tomarem decisões sobre suas ações. No outro extremo, agentes cognitivos possuem uma representação interna do ambiente no qual estão situados, e realizam decisões, adoção de objetivos, mudanças no curso de ações, baseados em mudanças que a percepção do ambiente causa em sua representação. Portanto, a modelagem de ambientes é igualmente importante em ambas as classes de sistemas multiagentes. A modelagem de ambientes é uma questão especialmente importante no desenvolvimento de SMA *Situados* (*Situated Multi-Agent Systems*), onde os agentes não agem diretamente em um ambiente real, tal como robôs com sensores e efetores reais. Porém, mesmo quando o ambiente do SMA é o *mundo real* e o agente é um robô com sensores e efetores, a modelagem do ambiente tem um papel importante no projeto do sistema. Qualquer robô deverá ter um conjunto de sensores que irá fornecer um conjunto predefinido de percepções provenientes do ambiente. Além disso, este também terá um conjunto de efetores que possibilitam um conjunto restrito de ações parametrizáveis. Portanto, as possíveis entradas dos sensores e as possíveis saídas dos efetores devem ser modeladas para facilitar o desenvolvimento do software de controle para tal robô.

Todavia, a literatura de sistemas multiagentes poucas vezes explicita esta parte da modelagem de sociedades de agentes, considerando os ambientes como parte da infraestrutura. Em (WEYNS et al., 2005) os autores afirmam que a maioria das abordagens populares reduzem o ambiente a apenas infraestrutura de comunicação ou falham em tratar o ambiente como uma entidade de primeira ordem em um SMA. Em alguns casos onde simplesmente não existe ambiente, o que Ferber chama de “SMA puramente comunicante” (*purely communicating MAS*) onde não existe ambiente e os agentes não fazem outras ações além de se comunicar diretamente através de mensagens (FERBER, 1999, pág. 11). Porém, neste caso o ambiente pode ser considerado a rede através da qual o agente se comunica.

A linguagem ELMS (**E**nvironment Description **L**anguage for **M**ulti-Agent

Simulation), criada para a simulação de ambientes para simulação multiagente, foi inicialmente apresentada em (OKUYAMA, 2003; OKUYAMA; BORDINI; ROCHA COSTA, 2005). A esta linguagem foi adicionada uma infraestrutura que permitisse a definição de uma estrutura normativa de maneira espacialmente distribuída, estas extensões são apresentadas em (OKUYAMA; BORDINI; ROCHA COSTA, 2007a,b). Como resultados destas extensões, foi obtida uma abordagem que integra de maneira explícita ambientes e estruturas organizacionais, como apresentado em (OKUYAMA; BORDINI; ROCHA COSTA, 2007c, 2008a).

A seguir, é apresentado de maneira geral a descrição de um ambiente físico usando o ELMS.

3.1 A Modelagem de Ambientes Utilizando ELMS

No modelo ELMS, entende-se por *modelagem do ambiente*, a modelagem de todos os aspectos do mundo exterior que um agente precisa saber para raciocinar e decidir seu curso de ação. Agentes também devem ser considerados partes integrantes do ambiente, visto que, do ponto de vista de um agente, qualquer outro agente é também parte do ambiente. Portanto, para definir agentes a partir deste ponto de vista, deve-se incluir na descrição do ambiente todas as propriedades que caracterizam os aspectos dos *corpos* dos agentes, que representam as propriedades de um agente que são perceptíveis para outros agentes. Desta forma, é necessário incluir na modelagem, de maneira explícita, as ações físicas e as capacidades de percepção que os agentes são capazes de realizar no ambiente em questão.

A descrição de um ambiente utilizando ELMS pode ser feita através da especificação de propriedades do ambiente que são modeladas como conjuntos de: objetos (referidos como *recursos*) do ambiente; *corpos* dos agentes; ações que cada agente pode realizar no ambiente; e tipos de percepções disponíveis para cada tipo de agente.

Nesse contexto, um agente pode ser descrito como tendo duas partes específicas: $agente = \{C_{Internas} + C_{Externas}\}$, onde as características internas($C_{Internas}$) seriam definidas pelo raciocínio do agente, que no modelo BDI seriam definidas pela manipulação de crenças, desejos e intenções. Enquanto as características externas($C_{Externas}$), as quais podem ser referidas como *Corpo do Agente* ou avatares, seriam dadas por: $Corpo(agente) = \{Prop_{Perceptiveis} + Percepcoes + Acoes\}$. Onde ($Prop_{Perceptiveis}$) refere-se as propriedades do agente que o projetista deseja incluir no ambiente, que poderão ser acessadas ou alteradas durante as interações com o ambiente.

No caso dos SMA, é comum a existência de vários agentes com características semelhantes, para facilitar a modelagem no modelo ELMS, são definidas *classes* de corpos de agentes que são instanciadas no ambiente. Assim, para cada classe de corpo são definidos como uma lista de propriedades (que define os aspectos perceptíveis dos agentes situados neste ambiente), uma lista de ações que estes são capazes de realizar (pró-ativamente) e uma lista de percepções as quais eles têm acesso.

Do ponto de vista do ambiente, as atividades de deliberação de um agente não são relevantes, pois são internas a este, ou seja, não são fisicamente observáveis para os outros agentes. Como mencionado antes, os aspectos internos dos agentes são descritos com o uso de AgentSpeak, que podem ser vistos no Apêndice C.

A especificação dos ambientes físicos pode ser definida como $Ambiente = \{Rep_{Espaco} + Objetos + Corpos(Agentes)\}$. Assim, cada objeto ou recurso é definido como $Recurso = \{Prop_{Perceptiveis} + Reacoes\}$. Como podem haver diversas instâncias

de objetos e recursos em um ambiente, define-se classes de recursos ou objetos que podem ser instanciados no ambiente, sendo cada classe composta um conjunto de propriedades e reações que estes podem realizar em resposta a estímulos externos ao objeto[1].

Para a definição dos tipos de percepção que cada tipo de agente tem acesso, é necessário definir quais propriedades do ambiente, agentes e objetos são perceptíveis para cada tipo de percepção. Também, as condições associadas com cada propriedade perceptível podem ser especificadas, as condições sob as quais uma propriedade é informada a um agente quando a percepção do agente é enviada a este. Na definição de percepções é possível especificar atributos de células, conteúdo de células, atributos de recursos ou agentes e variáveis de controle do ambiente.

Uma ação é definida pelas pré-condições que devem ser satisfeitas para execução desta e a seqüência de comandos que alteram as propriedades da estrutura de dados que representa o ambiente, de forna a representar as mudanças causadas pela execução da ação.

Em resumo, utilizando-se o ELMS para definição de um ambiente, é possível modelar um SMA aberto onde são definidas as diversas classes de corpos de agentes, que podem interagir com um conjunto pré-determinado de ações e percepções sobre uma representação espacial com tipos de recursos pré-definidos. Ou seja, um ambiente modelado usando ELMS seria:

$$Modelo(ambiente) = [Rep_{Espacial}] + Corpos(Agentes) + Recursos + Acoes + Percepcoes;$$

Onde *Corpos(Agentes)* refere-se à classes de corpos de agentes disponíveis para o ambiente e *Recursos* refere-se a classes de recursos que podem ser instanciados no ambiente; *Ações* refere-se aos tipos de ações disponíveis no ambiente e *Percepções* aos tipos de percepções disponíveis no ambiente.

Desta forma, uma instância de um ambiente ELMS conteria além das informações acima, informações sobre as instâncias de agentes e objetos presentes no ambiente:

$$Inst(ambiente) = Modelo(ambiente) + Inst(agentes) + Inst(recursos);$$

Na Seção 3.4, são apresentados as construções da linguagem ELMS para a especificação de ambientes.

3.2 Modelando os *Corpos* dos Agentes

Na descrição de um ambiente, cada classe de agentes é definida através da definição do *corpo do agente*, as capacidades sensoriais do agente e suas capacidades de efetuação.

Corpo do Agente: ($Corpo(agente) = Prop_{Perceptiveis} + Percepcoes + Acoes$) define as características dos agentes que são representadas no ambiente. Usando a linguagem ELMS, classes de "corpos" dos agentes são definidas por um conjunto de propriedade que caracterizam e são perceptíveis para outros agentes. Tais propriedades são representadas como variáveis *string*, *integer*, *float* e *boolean*. Para cada *corpo* é associado um conjunto de ações que o agente é capaz de realizar e as propriedades do ambiente que o agente é capaz de perceber.

[1] Os objetos podem reagir, apenas agentes são pró-ativos

Capacidades Sensoriais do Agente (Percepções): cada capacidade sensorial é usada para especificar quais propriedades do ambiente serão perceptíveis para cada agente que tenha o "corpo" com tal capacidade. Isto determina as propriedades do ambiente que serão percebidas pelo agente e as circunstâncias específicas onde estas são possíveis. Por exemplo, um agente poderá enxergar outros agentes em um raio de 2 células no escuro ou no raio de 4 células em boas condições. A definição de um tipo de percepção é formada por ($Percepcao = \{[Cond] + Propriedades\}$) um nome, uma lista opcional de pré-condições e uma lista de propriedades que são perceptíveis. Estas propriedades podem incluir propriedades de objetos, agentes, células grade, nodos do grafo ou variáveis de controle da simulação. Se todas as pré-condições forem satisfeitas, então os valores destas propriedades serão enviadas para o agente como resultado de sua percepção. Notem que a percepção poderá ser baseada na posição espacial do agente, porém isto não é obrigatório, qualquer tipo de percepção pode ser definido pelo projetista do ambiente, para atender as necessidades da aplicação. Outra questão importante relacionada à percepção é que os agentes normalmente não têm acesso completo ao ambiente, a percepção do ambiente não irá dar informações completas e precisas. Todavia, como tal restrição não é imposta pelo modelo ELMS, um ambiente pode ser definido desta forma se for apropriado em uma aplicação específica.

Capacidades de Efetuação do Agente (Ações): são as capacidades de realizar ações sobre o ambiente disponíveis para cada classe de "corpo" do agente. Cada definição de uma ação determina as mudanças no ambiente que o agente poderá realizar. Estas mudanças são definidas como atribuições de valores para os atributos do ambiente[2]. A produção (instanciação) de objetos previamente definidos e a deleção de instâncias também podem fazer parte de uma definição de ação. Para cada ação é definido um conjunto de pré-condições nas quais esta ação pode ser realizada. Assim, uma ação é dada por $Acao = \{[Cond] + Comandos\}$. É importante perceber que para que a coerência do ambiente seja mantida, as ações definidas deverão ser *atômicas*, uma vez que são executadas desta forma. Além disso, o encadeamento de ações é parte do raciocínio do agente, ou seja, uma seqüência de ações não deveria ser implementada como uma única ação disponível para os agentes no nível do ambiente.

3.3 Modelagem Espacial do Ambiente

O ambiente "físico" é modelado pela definição de objetos disponíveis no ambiente, as reações que podem ocorrer com tais objetos e a representação espacial do ambiente.

Objetos do Ambiente Físico: os objetos que estão presente no ambiente. Os agentes interagem com estes objetos através de ações que realizam sobre o ambiente. A estrutura dos objetos é definida por um conjunto de propriedade que são relevantes para serem representadas no ambiente, que poderão ser percebidas pelos agentes e alteradas no decorrer da simulação. Da mesma forma que os *corpos dos agentes*, as propriedades dos objetos podem ser representadas por um conjunto de variáveis do tipo *string*, *integer*, *float* ou *boolean*. As reações que uma classe de objetos pode realizar é dado por um conjunto de rótulos que identificam estas reações.

[2]Note que as propriedades dos corpos dos agentes também são propriedades do ambiente.

Reações dos Objetos: os objetos podem *reagir*, em condições específicas, em resposta à ações realizadas pelos agentes no ambiente. Tais reações são definidas como atribuições de valores à propriedades do ambiente, criação de instâncias de objetos pré-definidos e deleção de instâncias existentes de objetos. Note que todas as reações ativadas por alguma mudança no ambiente são executadas em um único ciclo de simulação, diferentemente dos agentes, onde cada agente pode realizar apenas uma ação por ciclo.

Representação Espacial: a representação espacial do ambiente pode ser feita por uma grade de células ou um grafo, de acordo com as necessidades da simulação ou preferências de projeto. Para simulações onde o posicionamento dos agentes e objetos é relevante, o uso da grade pode ser escolhido. Caso contrário, se os aspectos lógicos da representação espacial são mais relevante que o posicionamento preciso, uma representação baseada em um grafo pode ser mais adequada. Ambas as formas de representação espacial são opcionais e não-exclusivas, se necessário para alguma simulação específica, ambas poderão ser utilizadas ao mesmo tempo. Quando a grade é usada, o espaço é dividido em células formando uma grade que representa a estrutura espacial do ambiente, tanto em 2 ou 3 dimensões. Assim como os recursos, cada célula da grade podem ter reações associadas. Quando o grafo é utilizado, o espaço é representado pelos *nodos* que são conectados pelas *arestas*. Cada nodo do grafo representa uma localização espacial, onde uma reação pode ocorrer em resposta a alguma ação realizada. As arestas representam os caminhos entre os locais que o agente pode seguir, podendo ter valores ou pesos associados, de acordo com o modelo utilizado.

3.4 Construções da Linguagem ELMS

A linguagem ELMS usa uma sintaxe XML, porém a especificação de ambientes é feita através da interface da plataforma MAS-SOC, em desenvolvimento. Todavia, a especificação do ambiente pode ser feita em XML em um simples editor de texto, ou alguma outra ferramenta, se o usuário assim preferir.

Para a simulação de um ambiente são necessárias algumas informações extras específicas de sua operação, como inicialização de valores e a seleção de dados de saída da simulação do ambiente. Assim, uma especificação de ambiente em ELMS é composta por um conjunto de definições que podem ser de nove tipos, listadas a seguir:

1. Representação Espacial – Definição de uma grade ou um grafo, ambos opcionais, utilizados de acordo com a necessidade da aplicação para a representação espacial do ambiente.

2. Recursos – Definição das classes de recursos (objetos) que podem estar presentes no ambiente.

3. Agentes – Definição das classes agentes que podem estar presentes no ambiente da simulação.

4. Percepções – Definição dos tipos de percepções que podem ser realizadas no ambiente definido para a simulação.

5. Ações – Definição das ações que os agentes podem realizar no ambiente.

6. Reações – Definição das reações que os recursos podem ocasionar no ambiente.

7. Observáveis – Definição de uma lista de propriedades cujos valores mostrados e armazenados. Estas são as propriedades específicas da simulação que o usuário quer observar.

8. Valores da Simulação – Definição de valores correntes das propriedades dos elementos instanciados.

9. Inicialização – Definição de comandos que serão executados no ambiente antes do início da simulação.

Estes ítens são detalhados no Apêndice A.

3.5 Classificação de Ambientes

Em (RUSSELL; NORVIG, 2003, pág. 41) é fornecida uma classificação para os possíveis ambientes, como visto a seguir:

Completamente observável ou parcialmente observável. Se os sensores de um agente podem dar acesso ao estado completo do ambiente, então pode-se dizer que o ambiente é acessível para este agente. Um ambiente é efetivamente acessível se os sensores detectam todos os aspectos relevantes à escolha da ação.

Determinístico ou estocástico. Se o próximo estado do ambiente é completamente determinado pelo estado corrente e as ações selecionadas pelos agentes, então pode-se dizer que o ambiente é determinístico. A princípio, um agente não precisa se preocupar com a incerteza em um ambiente acessível e determinístico. Caso o ambiente seja inacessível, então este pode parecer estocástico, do ponto de vista do agente. Porém, se o ambiente é determinístico com exceção da ação dos outros agentes, este é chamado de *estratégico*.

Episódico ou seqüencial. Em um ambiente episódico, as experiências dos agentes são divididas em episódios. Em cada episódio, o comportamento do agente consiste apenas em perceber e agir, pois estas ações não irão interferir nos episódios subseqüentes.

Estático ou dinâmico. Se o ambiente pode mudar enquanto o agente está deliberando, então este ambiente é dinâmico para este agente, caso contrário o ambiente é estático. Se o ambiente não muda com a passagem do tempo, mas o desempenho do agente é medido pelo seu tempo de deliberação, então o ambiente é semidinâmico.

Discreto ou contínuo. Se existe um número determinado e limitado de percepções e ações, pode-se dizer que o ambiente é discreto. Um jogo de xadrez é um ambiente discreto, pois há um número fixo de possíveis movimentos para jogada, enquanto dirigir é contínuo, pois a velocidade e a localização dos outros veículos variam de maneira contínua.

Multiagente ou único agente A distinção entre ambientes multiagentes e de um único agente pode parecer clara, um agente resolvendo palavras cruzadas é claramente

um ambiente de um único agente, enquanto um agente jogando xadrez contra outro agente é um ambiente multiagente. Porém, isto dependerá de como é feita a modelagem e o que será considerado um agente dentro deste modelo. Dentro da classificação dos ambientes multiagentes, ainda é possível classificá-los como *cooperativos*, *competitivos* ou *semi-competitivos*.

A tabela 3.1, adaptada de Russell e Norvig (2003, pág. 43), apresenta alguns exemplos de ambientes e sua classificação segundo os critérios acima.

Tabela 3.1: Exemplos de Ambientes e Características

	Observável	**Determinístico**	**Episódico**	**Estático**	**Discreto**	**Agente**
Palavras Cruzadas	Completo	Determinístico	Seqüencial	Estático	Discreto	Único
Xadrez com relógio	Completo	Estratégico	Seqüencial	Semi	Discreto	Multi
Pôquer	Parcial	Estratégico	Seqüencial	Estático	Discreto	Multi
Gamão	Completo	Estratégico	Seqüencial	Estático	Discreto	Multi
Dirigir automóvel	Parcial	Estratégico	Seqüencial	Dinâmico	Contínuo	Multi
Sistema de diagnóstico médico	Parcial	Estratégico	Seqüencial	Dinâmico	Contínuo	Único
Análise de imagem	Completo	Determinístico	Episódico	Semi	Contínuo	Único
Robô carregador de blocos	Parcial	Estocástico	Episódico	Dinâmico	Contínuo	Único
Controlador de refinaria	Parcial	Estocástico	Seqüencial	Dinâmico	Contínuo	Único
Tutor interativo de inglês	Parcial	Estocástico	Seqüencial	Dinâmico	Discreto	Multi

Baseado na classificação acima, seria possível com a linguagem ELMS especificar ambientes que são, do ponto de vista dos agentes, parcialmente observáveis, estocásticos, seqüenciais, dinâmicos, multiagente, porém devem ser discretos. Nesta seção, mostra-se como cada uma destas características de ambientes podem ser modeladas em ELMS.

Parcialmente observável: um ambiente completamente observável é um ambiente onde o agente tem acesso ao estado completo do ambiente e um ambiente parcialmente observável é onde o agente ter acesso restrito ao estado do ambiente. Desta forma, na modelagem com o ELMS, o agente tem acesso apenas às propriedades do ambiente que foram escolhidas pelo projetista do ambiente, na definição das percepções.

Estocástico: Como os ambientes ELMS podem ser parcialmente observáveis, somada a possibilidade de poderem existir vários agentes realizando ações simultaneamente (ocasionando reações por parte do ambiente), do ponto de vista de um agente, os ambientes ELMS podem ser classificados como *estratégicos*.

Seqüenciais: nos ambientes ELMS, o estado corrente do ambiente é o produto das ações realizadas pelos agentes no estado anterior. Porém como tratam-se de agentes cognitivos, as ações tomadas por um agente poderão interferir em suas ações futuras, já que é um processo interno ao agente que determina como ele reage a cada ciclo de execução.

Dinâmico: O ambiente pode sofrer alterações enquanto um agente estiver deliberando, devido à ações de outros agentes presentes no ambiente. Mesmo no modo síncrono,

apesar do sistema aguardar pelas ações de todos os agentes, uma ação pode gerar uma mudança no ambiente que bloqueia a ação de outro agente, o que do ponto de vista do agente bloqueado, faria com que o ambiente parecesse dinâmico.

Multiagente: O ELMS permite a simulação de ambiente que são compartilhados por vários agentes. Que podem ser competitivos ou cooperativos de acordo com a aplicação em questão.

Discreto: como o "espaço físico" do ambiente é representado através de uma grade, o ambiente é discretizado em coordenadas inteiras.

3.6 Considerações Finais

Conforme mencionado anteriormente na Seção 2.4, os elementos de um SMA são usualmente definidos como elementos independentes. Com o uso da linguagem ELMS, é possível modelar o ambiente de forma a ligar explicitamente a definição do raciocínio dos agentes à definição do ambiente físico. Com a modelagem das percepções, efetores e da representação de características físicas dos agentes, integra-se o modelo de raciocínio de agente ao ambiente.

Esta forma de modelagem poderia ser aplicada no desenvolvimento de SMA, tanto em abordagens *bottom-up* quanto *top-down*. Assim, a modelagem do ambiente descreveria de maneira geral a *situação problemática* a ser resovida:

- **Percepções:** A definição das percepções delimita as informações de entrada que o agente terá para realizar o seu raciocínio;
- **Efetores:** A definição do conjunto de ações disponíveis para cada agente, delimita o conjunto de ações que o agente tem disponível para atingir seus objetivos;
- **Ambiente físico:** Através da definição dos objetos e agentes presente no sistema, define-se o escopo de tipos objetos e tipos de agentes com o qual o agente poderá interagir.

Mesmo em um cenário SMA aberto, com agentes heterogêneos, cada agente teria seu próprio conjunto de sensores e efetores, o que limita a maneira como os agentes "enxergam" uns aos outros e o conjunto de ações que estes podem realizar sobre o ambiente.

3.6.1 Funcionalidades Estendidas da Linguagem ELMS

A linguagem ELMS, inicialmente apresentada na dissertação de mestrado (OKUYAMA, 2003), não apresentava alguma das funcionalidades apresentadas nesta seção. Entre as novas funcionalidades adicionadas ao ELMS pode-se citar:

- Possibilidade de definir *métodos construtores* para objetos e agentes, executados logo após a instanciação, para inicializar suas propriedades.
- Uso do interpretador ***Jason***, na versão anterior o ELMS utilizava o interpretador AgentSpeak(XL) (BORDINI et al., 2002) que foi descontinuado.
- Representação espacial com grafo. A possibilidade de ter um grafo para representar o espaço, agrega novas possibilidades de modelagem à linguagem ELMS.
- Infraestrutura Normativa, detalhada a seguir, no Capítulo 4.

4 INFRAESTRUTURA NORMATIVA

Neste capítulo, são apresentadas as extensões feitas ao ELMS que permitem fornecer uma infraestrutura para disponibilizar informações normativas de maneira distribuída no ambiente do SMA. Estas extensões são referidas como *infraestrutura normativa*, enquanto é utilizado o termo *estrutura normativa* para designar a um conjunto de normas estruturadas que regulam uma organização. As extensões introduzidas no ELMS possibilitam a distribuição espacial de normas no ambiente, permitindo integrar o ambiente físico com estruturas organizacionais, aumentando significativamente as possibilidades do modelo de simulação.

4.1 Objetos Normativos

De maneira geral é possível identificar alguns objetos presentes no ambiente de um sistema social, que têm como objetivo informar os agentes do sistema sobre normas, dar algum orientação ou avisar sobre algum perigo potencial. Por exemplo, um cartaz em uma parede solicitando "silêncio" é um objeto do ambiente, mas também informa sobre uma norma que deve ser respeitada naquele espaço. Outro exemplo são os sinais de trânsito, que orientam sobre direções ou regulam a preferência em cruzamentos. A existência de tal sinalização, implica na existência de um código regulador em tal contexto, o que é referido como *normas situadas*.

Nos exemplos acima, as normas devem ser seguidas apenas em determinado escopo de espaço ou tempo, perdendo seu efeito completamente se as restrições de espaço e tempo não são satisfeitas, o que é a motivação para as normas situadas. Outra vantagem significativa de modelar algumas normas como normas situadas é o fato que o contexto espacial onde a norma deve ser seguida é determinado. Portanto, a norma pode ser *pré-compilada* em sua forma situada, tornado mais fácil para o agente operacionalizar a norma, e também facilitar a verificação de conformidade com as normas.

Como mencionado, a noção de *objetos normativos* pode ser relacionada a cartazes existentes em locais públicos, tais como bibliotecas ou restaurantes que dizem "mantenha o silêncio" ou "proibido fumar". Este mecanismo é usualmente utilizado entre humanos para descentralizar o serviço de regular o comportamento social, porque as pessoas adotam estas normas ou convenções quando têm acesso visual a tais cartazes. Isto aplicado para sistemas computacionais pode ser igualmente eficiente, especialmente para agentes, pois evita a necessidade de ter todas as normas implementadas em cada um dos agentes ou a necessidade de prover uma completa e exaustiva representação de todas as normas sociais em uma única estrutura acessada por todos os agentes, tal como é normalmente feito em diversas abordagens de organizações de agentes.

De maneira resumida, os ambientes descritos com o ELMS suportam *organizações*

situadas através de *normas situadas* e *estruturas de grupo situadas*. Isto é alcançado através de dois itens, onde o primeiro seria a *infraestrutura normativa* que permite a distribuição da infraestrutura normativa sobre o ambiente espacial. O segundo item seria um *princípio normativo*, que seria associado ao projeto do SMA e em especial ao raciocínio do agente, concebido na forma de uma regra condicional, acionada quando da percepção de uma norma através de um objeto normativo:

> *Quando o agente $\mathcal{A}$ realizando o papel relevante à norma $\mathcal{N}$, expressa em um objeto normativo percebido e estando $\mathcal{A}$ posicionado dentro do espaço referenciado pela norma $\mathcal{N}$, é esperado do agente $\mathcal{A}$ que raciocine sobre seguir ou não a norma $\mathcal{N}$; caso não tenha percebido o objeto normativo, o agente é isento de raciocinar sobre isto.*

Cada objeto normativo pode ser posicionado em um *espaço normativo*, que determina a primeira condição para que o objeto seja percebido: é apenas naquele *espaço normativo* que o conteúdo do objeto normativo é relevante. As condições sob as quais os objetos normativos podem ser percebidos são definidas pelo projetista da simulação usando construções da linguagem ELMS de maneira semelhante à definição de condições de percepções. *Objetos normativos* podem ser *lidos* por agentes em condições específicas, ou seja, um agente pode ler uma regra específica se este tiver uma habilidade específica para perceber aquele tipo de objeto. No caso mais típico, a condição poderia ser simplesmente estar fisicamente próximo ao objeto.

Uma definição de um objeto normativo contém, além da norma, meta-informações da norma. Os objetos podem ser definidos antes do início da simulação em um "arquivo de definição de normas" ou durante a simulação. Em ambos os casos, são definidos o *tipo* da informação contida, a origem da norma, a norma, o posicionamento do objeto, a condição de acesso e um rótulo de identificação da norma. Estes campos são detalhados a seguir:

Rótulo: uma *string* de identificação para uma eventual deleção ou edição do objeto normativo;

Tipo: o tipo de informação normativa contida no objeto; determina o nível de importância (aviso, obrigação, orientação);

Emissor: agente ou grupo que emitiu a norma; o emissor da norma *não* é necessariamente o mesmo agente que instanciou o objeto;

Origem: define a origem do poder que sustenta a norma; o papel que estava sendo desempenhado pelo agente na emissão da norma e a instituição que que permitiu a emissão da norma;

Norma: uma *string* que representa a informação normativa; esta deve estar no formato de um predicado AgentSpeak; para uma maior uniformidade na representação da norma optou-se por adotar a linguagem REI (KAGAL; FININ; JOSHI, 2003), apesar de ser possível utilizar outro formato desde que os agentes estejam preparados para processar;

Destinatários: os membros da organização (agentes, grupos e papéis) aos quais a informação normativa se aplica.

Posicionamento: o conjunto de espaços normativos onde a informação normativa se aplica. Se omitido, o objeto é assumido como sendo válido em qualquer lugar, mas normalmente sob condições específicas determinadas pelo projetista;

Condição: conjunto de condições sob as quais a informação normativa pode ser percebida; as condições podem ser temporais, espaciais ou relacionadas à grupos, papéis e habilidades, de acordo com o a necessidades da aplicação;

Supervisores: conjunto de agentes ou papéis que irão receber informações relacionadas à norma. Estes supervisores são detalhados na Seção 4.1.3;

4.1.1 Percepção Condicional de Normas

A existência de uma condição para a percepção do conteúdo dos objetos normativos possibilitam várias utilizações. A utilização mais direta, seria selecionar os destinatários das normas, evitando assim a transmissão de um número excessivo de normas para todos os agentes, dentre os quais poderiam estar diversos agentes aos quais a norma não se aplica.

Outra possibilidade de aplicação da percepção condicional seria possibilitar que os objetos normativos não informem normas que estejam fora do contexto temporal, no caso de uma norma do tipo “Proibido estacionar das 7h as 19h” não precisaria ser divulgada nos horários em que esta não é válida. Desta forma seria possível contextualizar temporalmente as normas. Assim, o raciocínio temporal exigido do agente seria simplificado. Uma vez que, a norma seria percebida apenas no contexto temporal em que está é válida e deve ser seguida.

Além disso, a possibilidade de se condicionar a percepção da norma com a validade da mesma simplifica de maneira expressiva o raciocínio normativo dos agentes, permitindo que estes recebam normas espacialmente e temporalmente contextualizadas, apenas quando válidas.

4.1.2 Definição de Objetos Normativos

Cada objeto normativo é definido da seguinte maneira:

```
NormObj nome{
    norm       := "has(agente,prohibition(smoke,true)";
    type       := "warning";
    position   := {library};
    addressees := {ALL};
    condition  := (true);
    issuedBy   := {agent1};
    source     := {groupA};
    supervisor := {library.adminStaff};
}
```

Uma observação importante é que o comportamento de aderência às normas não está relacionado apenas com a existência de um objeto normativo em algum lugar. Além da existência de tal objeto, é necessário que o agente perceba o objeto e seja capaz de interpretar a norma; e no caso de agentes autônomos, que eles raciocinem sobre sua opção de seguir ou não a norma indicada pelo objeto normativo.

4.1.3 Supervisores de Normas

No modelo MAS-SOC, é definida uma classe especial de agentes referidos como agentes *supervisores de normas* que podem monitorar a aderência às normas de uma organização ou um SMA. Considerando que os agentes são autônomos para decidir se seguem ou não uma norma indicada por um objeto normativo, somado à limitações na capacidade de percepção do ambiente, pode existir a necessidade de monitorar o comportamento destes agentes, de acordo com as finalidades da aplicação. Para ser capaz de agir como um supervisor de normas, um agente pode precisar de informações e habilidades extras. Para isto, é necessário definir o agente como *agente supervisor*, o que irá habilitá-lo a receber informações sobre a norma e as ações sendo realizadas por outros agentes em um determinado espaço normativo.

Agentes com as características de monitoração podem ser agentes externos à simulação em andamento, sendo agentes especialmente projetados para verificação de aderência a normas para evitar abusos ou testar estratégias de coordenação ou cooperação. Outra possibilidade é serem agentes participantes da simulação, cujo interesses necessitam que outros agentes sigam determinadas normas, já que para os agentes que compartilham o mesmo ambiente, pode ser de interesse que alguns agentes sigam certas regras. Por exemplo, de acordo com (CONTE; CASTELFRANCHI, 1995), um agente pode ser motivado a verificar a aderência às normas por outros agentes para assegurar que os custos da aderência estão sendo "pagos" pelos outros agentes também. Um agente que segue as regras irá querer que todos os outros agentes sujeitos às normas também sigam esta, caso contrário, o comportamento de seguir as regras pode se tornar uma desvantagem competitiva. Em (CONTE; CASTELFRANCHI, 1995), estes agentes são referidos pelos autores como *defensores de normas*.

No contexto dos *espaços normativos*, as normas e as possíveis violações estão limitadas em um escopo espacial, o que torna muito mais fácil para identificar as violações destas normas. Através do uso de regras simples, um supervisor de normas pode verificar a aderência dos agentes às normas de acordo com suas capacidades poderá interromper o curso de ação de outro agente; impor penalidades ou punição apropriada; ou simplesmente reportar a infração a uma instituição ou ao usuário.

É importante notar que os supervisores de normas não têm como objetivo intrínseco impedir ou interromper a infração de regras por outros agentes. Os agentes supervisores são agentes que têm acesso a informações extras para serem capazes de verificar a aderência às normas. O projetista da aplicação pode habilitar tal capacidade em um agente apenas para facilitar que o agente atinja seu objetivo, para usar esta informação para monitorar a aplicação ou como entrada para um sistema de reputação, entre outras possibilidades.

4.2 Espaços Normativos

Espaços normativo (*normative places*), são abstrações para definir limites espaciais que podem estar associados à um conjunto de atividades ou a um grupo de agentes. Dentro destes limites, determinadas regras devem ser seguidas. Assim, estes espaços podem ser utilizados para representar o local físico ao qual uma estrutura organizacional está relacionada, ou seja, um *espaço normativo* define escopos espaciais de uma organização e conseqüentemente das normas relacionadas a para cada escopo.

Nos *espaços normativos*, as normas relevantes são expressas através dos objetos normativos. Por exemplo, considere um grupo de pesquisa cujos agentes *pesquisadores* fa-

zem seu trabalho tanto em um laboratório quanto em uma biblioteca. No laboratório, as interações possíveis entre os pesquisadores, equipe de apoio e ambiente são reguladas de maneira específica dentro escopo espacial do laboratório (tais como tipos e direitos de acesso à área e equipamentos). A informação sobre como agir na biblioteca é definida especificamente para o contexto espacial da biblioteca, onde os pesquisadores irão assumir o papel de *usuários* da biblioteca. Desta forma, a informação normativa relevante para cada local é armazenada neste local com o uso dos objetos normativos.

Um *espaço normativo* é definido por um rótulo identificador (nome) e seus limites espaciais compostos por um conjunto de células de uma grade ou nodos de um grafo. Para cada *espaço normativo* poderá ser definido um conjunto de papéis relacionados à atividades realizadas neste espaço; sendo as relações entre estes papéis definidas através de *objetos normativos* localizados em tal espaço.

Um espaço normativo pode ter intersecções com outros espaços normativos, ou podem ser contidos por outro. Por exemplo, um espaço normativo *escola* pode conter um espaço normativo “sala de aula” e outro “biblioteca”.

A definição dos *espaços normativos* em ELMS permite a especificação de uma abstração espacial onde determinadas normas são válidas e relevantes, como será mostrado na próxima seção. Os papéis relacionados à cada espaço normativo são relativos especificamente às atividades sendo realizadas em tal local, que podem não ter uma relação direta com o papel adotado por um agente em uma outra estrutura organizacional. Por exemplo, em um SMA que representa uma cidade, pode existir um espaço normativo “rua” onde um agente dirigindo um carro terá o papel de “motorista”, enquanto outro sem carro terá o papel de “pedestre”. Em tal caso, do ponto de vista de projeto de tal seção do ambiente e sua estrutura normativa, os agentes estão simplesmente movendo-se de um local a outro, independentemente de qualquer outro papel que possam ter em outras organizações presentes no SMA.

A área abrangida por um espaço normativo pode aumentar ou diminuir durante a simulação, uma vez que se trata de ambientes possivelmente dinâmicos que podem estar associados com organizações possivelmente dinâmicas. Assim, a área de influência de uma organização pode se expandir ou reduzir dinamicamente, de acordo com as características da aplicação, bastando apenas alterar o conjunto de células ou nodos definidos para aquele espaço normativo, ou pelo deslocamento de objetos normativos, ou pela criação de novos objetos normativos. As áreas dos espaços normativos poderão ser alteradas durante a simulação por agentes que estejam habilitados pela organização a alterar tal estrutura, de acordo com as necessidades da aplicação, definido pelo projetista.

Tais situações podem ocorrer basicamente de duas formas:

- Quando a organização deliberadamente altera a área de um espaço normativo para influenciar o comportamento dos agentes;

- Quando a organização reconhece que o comportamento prescrito para os agentes em um determinado local é praticado em uma área diferente da qual deveria ser. Esta área pode ser maior ou menor que a área prescrita pela organização. Em ambos os casos, a organização pode alterar sua área de operação (uma estrutura normativa) para refletir o comportamento (emergente) dos agentes.

Assim, um comportamento social diferente pode emergir se alterada a distribuição dos objetos normativos onde uma organização específica está situada, ou se forem criados novos objetos normativos. Desta forma, pode-se criar situações onde uma organização

onde esta define estruturas normativas (*top-down*) que influenciam o comportamento dos agentes, resultando em um comportamento emergente (*bottom-up*) que pode ser analisado para ajustar os objetos normativos e resultar na emergência de um novo comportamento (*emergência de segunda ordem*). Estas situações podem ocorrer em várias simulações sociais, sendo a existência de abstrações de alto-nível para modelar tais situações uma grande facilidade no desenvolvimento de tais aplicações.

4.2.1 Adoção Implícita de Papéis

Nos casos onde os papéis não são definidos previamente pelo sistema, é comum um agente identificar-se explicitamente para outros agentes ou instituição para adotar um papel específico. Por exemplo, em um website que oferece serviços bancários, um agente deverá se autenticar, com nome de usuário e senha, para ter acesso à informações privadas sobre sua conta, adotando explicitamente o papel de um *usuário identificado*, adquirindo acesso a direitos específicos de seu papel. Porém, caso o usuário não se identifique, este ainda poderá ter acesso a informações públicas disponíveis no website. Neste caso, a este usuário estará implicitamente adotando o papel *default* de *visitante anônimo*. Caso este usuário não-identificado inicie uma tentativa de acessar dados privados do banco, por esta atividade este está implicitamente o papel de *invasor*.

Em cada ambiente espacialmente e temporalmente limitado, o agente poderá adotar papéis temporários de acordo com a atividade que esteja realizando neste local. A adoção de tais papéis pode acontecer de maneira explícita ou implícita. Em um espaço normativo existe um número limitado de atividades que podem ser realizadas, uma vez que o contexto é específico e as ações são limitadas pela especificação do ambiente. Portanto, para cada espaço normativo poderá ser identificado e definido um conjunto limitado de *papéis admissíveis*, regulados pelos objetos normativos presentes neste local. Os papéis admissíveis em cada espaço normativo podem ser associados a classes de agentes, realização de uma atividade ou pode ser atribuído um papel *default* para o agente, até que este se identifique e adote determinado papel.

Através de informações específicas, tal como posicionamento espacial, orientação, posse de determinados objetos, atributos do agente ou papéis em organizações, a adoção implícita pode ser realizada, o que pode ser definido para cada espaço normativo com o uso de regra simples.

Na definição dos espaços normativos, juntamente com a definição do conjunto de papéis possíveis, pode-se definir as condições nas quais os papéis podem ser adotados implicitamente. Abaixo, alguns exemplos, com uso de pseudo-código, de como isto pode ser feito:

Papel *Default*: um papel *default* pode ser definido para cada local, por exemplo em uma biblioteca, o papel *usuário* pode ser atribuído como *default*;

Posse de um item: se o agente possui um objeto que o relacione a um papel. Por exemplo, um cartão de identificação pode associar o agente ao papel de *funcionário* da biblioteca;

Posicionamento: de acordo com sua posição, um agente pode ter um papel específico, por exemplo um agente em um carro, sentado no assento do motorista, será atribuído o papel de *motorista*:

Papel Relativo: o papel em uma estrutura organizacional pode ser associado a um papel

em outra organização. Por exemplo um pesquisador de uma universidade, poderá realizar o papel de *pesquisador-visitante* em uma outra universidade;

Os papéis admissíveis em cada espaço normativo são estritamente relativos às funções que estão sendo realizadas neste local, o que pode não ter uma relação direta entre papéis existente em uma outra organização. Por exemplo, em um espaço normativo *rua*, um agente com um carro terá o papel de *motorista*, enquanto outro sem carro terá o papel de *pedestre*. Em tal caso, do ponto de vista do projeto do ambiente e sua infraestrutura normativa, os agentes estão simplesmente movendo de um ponto a outro, independentemente de qualquer outro objetivo que eles possam ter.

4.2.2 Definição de Espaços Normativos

Um espaço normativo é definido da seguinte forma:

```
NormSpace library{
    area  :={cell[0,0], cell[0,1], cell[1,0], cell[1,1]};
    owner := {role(adminChief)};
    roles := {user, staff, adminStaff, adminChief}
    roleAdoption{
        condition: (agent.hasAdminBadge == true)
              -> agent.role = adminStaff;
        condition: (agent.hasStaffBadge == true)
              -> agent.role = staff;
        defaultRole: user;
    }
}
implements(agentA, library.adminChief)
```

No exemplo acima, é definido um espaço normativo de que ocupa quatro células da grade, rotulado como *library*. Neste espaço estão podem ser adotados os papéis *user*, *staff* e *adminStaff*. Por exemplo, se o agente tiver a condição `agent.hasStaffBadge` como verdadeira, a este será atribuído o papel de *staff*, caso nenhuma das condições seja verificada, será atribuído o papel *default*, que neste caso seria `user`.

Assim, a definição de um espaço normativo é definido pelos itens:

Rótulo: uma *string* de identificação para uma eventual deleção ou edição do espaço normativo;

Area: a área de influencia do espaço normativo;

Owner: conjunto de agentes que possuem direitos para alterar o espaço normativo;

Roles: conjunto de papéis que podem ser desempenhados dentro do espaço normativo;

Role Adoption Rules: regras para adoção implícita de papéis;

implements(role, agent): indicação dos papéis implementados por instâncias de agentes.

4.3 Contextualização Espacial de Normas

O objetivo dos objetos normativos não é simplesmente servir como meio para divulgar normas genéricas. As normas informadas através dos objetos normativos devem ser contextualizadas (pelo projetista ou pelo agente emissor da norma), incorporando informações específicas para o espaço normativo onde esta é relevante.

Como o contexto espacial da norma é limitado e determinado pelos espaços normativos, uma norma genérica pode ser *pré-compilada* com tal informação, para que esta se torne menos abstrata. Este processo tem como objetivo facilitar a operacionalização das normas, já que a norma se encontra "pronta para uso" no contexto espacial onde é relevante. Entre as vantagens de ter normas menos abstratas está a de facilitar a verificação de aderência às normas e também a redução dos erros de interpretação que podem ocorrer com normas abstratas não-contextualizadas.

Por exemplo, uma norma com o conteúdo: "seja gentil com os idosos", pode ser um tanto difícil de se operacionalizar e verificar, de maneira geral. Porém, em um contexto espacial fixo, tal como um ônibus ou metrô, a norma poderia ser contextualizada como: "Ceda seu assento para os idosos", ou em um cruzamento de avenidas: "ajude os idosos a atravessar a rua". Desta forma, a norma seria muito mais facilmente interpretada pelos agentes e conseqüentemente muito mais facilmente verificada por algum mecanismo de verificação de normas.

4.4 Linguagem para Especificação das Normas

Para a especificação de normas informadas através dos objetos normativos, esta deve ser um predicado válido. É conveniente a adoção de uma padronização de um formato para especificação de normas, sendo possível utilizar qualquer formato desde que os agentes estejam preparados para processar. No contexto do projeto MAS-SOC, a linguagem para definição de normas para uso nos objetos normativos deve possibilitar, ao menos, a definição de direitos e obrigações. Como tratam-se de normas referentes à espaços normativos que podem ser sobrepostos, a linguagem deve possuir meios que possibilitem a resolução de conflitos.

Para que haja uma maior uniformidade na representação de normas, optou-se por adotar a linguagem REI (KAGAL; FININ; JOSHI, 2003). A linguagem REI é uma linguagem destinada à definição de políticas para computação pervasiva, sendo bastante adequada e abrangente para o uso no MAS-SOC. Considerando que o paradigma da computação pervasiva é bastante adequado à idéia de diversos espaços normativos em um ambiente físico simulado. Na linguagem REI existem construções para definir direitos, proibições, obrigações e dispensas (liberação de obrigação). Além disto a linguagem possui várias outras construções, entre as quais construções específicas para resolução de conflitos (e.g. prioridades), especificação de como as ações devem ser executadas e gerenciamento de delegação de tarefas. Uma descrição mais completa pode ser encontrada em `http://www.cs.umbc.edu/~lkagal1/rei`. A seguir são apresentadas algumas das construções disponíveis na linguagem REI, resumidas de (KAGAL; FININ; JOSHI, 2003).

Direitos, proibições, obrigações e dispensas:

- `has(agent_ag, right(action_act, Conditions))`: significa que o agente *agent_ag* tem o direito de executar a ação *action_act* se as condições *Conditions* forem satisfeitas. A noção de direito ou permissão pode ser interpretada como

a expressão deôntica $\sim$O$\sim$.

- `has(agent_ag, prohibition(action_act, Conditions))`: significa que o agente *agent_ag* é proibido de executar a ação *action_act* se as condições *Conditions* forem satisfeitas. A noção de proibição pode ser interpretada como a expressão deôntica O$\sim$.

- `has(agent_ag, obligation(action_act, Conditions))`: significa que o agente *agent_ag* é obrigado à executar a ação *action_act* se as condições *Conditions* forem satisfeitas. A noção de obrigação pode ser interpretada como o operador deôntico O.

- `has(agent_ag, dispensation(action_act, Conditions))`: significa que o agente *agent_ag* é dispensado de executar a ação *action_act* se as condições *Conditions* forem satisfeitas. A noção de proibição pode ser interpretada como a expressão deôntica $\sim$O.

Definição de prioridades:

- `overrides(R1,R2)`: significa que a regra *R1* sobrepõe a regra *R2*.

A linguagem REI sugere uma forma de representação das ações pela qual uma ação pode ser melhor compreendida neste nível de abstração[1]. Especificação de ações é feita da seguinte forma:

- `action(ActionName, TargetObjects, Pre-Conditions, Effects)`: a ação `ActionName` que têm como objeto-alvos `TargetObjects`, pré-condições `Pre-Conditions` e ocasionam os efeitos `Effects`. No exemplo `action(printOnePage, printer, (hasCartridge(printer), hasPaper(printer)), paperDecrease(printer))` indica que a ação `printOnePage` têm como objeto alvo o objeto `printer`, que precisa ter como pré-condições `hasCartridge` e `hasPaper` e ocasionará como efeito no objeto-alvo o decréscimo de papel.

Além da especificação em alto nível das ações, também é possível especificar alguns operadores de ações:

- `seq(A,B)`: significa que as ações `A` e `B` devem ser executadas em seqüência.

- `nond(A,B)`: significa uma escolha entre as ações `A` e `B` onde apenas uma pode ser escolhida.

- `repetition(A)`: significa que a ação `A` pode ser executada repetidas vezes.

- `once(A)`: significa que a ação `A` pode ser executada apenas uma vez.

[1]Esta é uma forma de representação em um nível de abstração mais elevado em relação a definição de ações feita no ELMS.

4.5 Biblioteca de Planos Relacionados à Normas

Para facilitar a programação do raciocínio dos agentes, foram desenvolvido alguns planos AgentSpeak para lidar alguns tipos de normas. Estes planos estão organizados em arquivos que podem ser importados em outro arquivo AgentSpeak através do uso da diretiva `include`. Desde que tais planos estão disponíveis em arquivos comuns, é também possível usá-los como modelos para construir planos personalizados de acordo com as necessidades específicas da aplicação.

Para programar um agente que nunca viola uma proibição para executar uma ação `act`, o programador deve substituir em seu programa AgentSpeak todas as ocorrências[2] de `act` por `!execute(act)` e incluir os seguinte planos na biblioteca de planos do agente:

```
+!execute(Action)
     : not prohibition(Action,_)
     <- Action.

+!execute(Action)
     : prohibition(Action,Condition)
       & not Condition
     <- Action.

+!execute(Action)
     : prohibition(Action,Condition)
       & Condition
     <- fail.
```

Como outro exemplo, para programar um agente que sempre cumpre uma obrigação determinada por uma organização de sua confiança, a não ser que o agente seja dispensado da obrigação ou se esta violar alguma proibição:

```
+has(Self,obligation(Action,Condition))[sourceOrg(SO)]
    :  .my_name(Self) & trusted(SO) & Condition
    <- !checkDispensation(Action);

//dispensa existe, condição falsa
+!checkDispensation(Action) : .my_name(Self) & not
    Condition & has(Self,dispensation(Action,Condition))
    <- !execute(Action).

//dispensa existe, não faz nada
+!checkDispensation(Action)
    : .my_name(Self)
     & Condition
     & has(Self,dispensation(Action,Condition)).

//não há dispensa
+!checkDispensation(Action)
    <- !execute(Action).
```

No trecho de código acima, para tratar o evento que indica uma nova obrigação que foi percebida, o agente verifica se a organização que emitiu a norma é de sua confiança. Em caso positivo, as condições para a obrigação e então verifica se existe uma dispensa para tal obrigação. Caso contrário, a ação é executada, após verificar se não existe uma

[2]Utilizando o ***Jason***, é possível usar o pré-processamento para fazer todas as substituições automaticamente.

proibição relacionada a tal ação. Neste caso específico, o código AgentSpeak foi desenvolvido de forma a dar prioridade à proibição. O tratamento de contradições que podem ocorrer em sistemas normativos tais como uma ação ser obrigatória e proibida é um tópico de pesquisa corrente que não foi abordado neste simples exemplo.

4.6 Acesso Dinâmico à Infraestrutura

Para alteração da infraestrutura normativa durante a simulação, os agentes podem enviar comandos para alterar tanto objetos normativos, quanto espaços normativos. Um processo, que implementa a mesma interface de comunicação de um agente, recebe as mensagens dos agentes, verifica se o agente que enviou a mensagem tem direitos suficientes para alterar o item e realiza a atualização, conforme a Figura 4.1. As mensagens são enviadas da mesma maneira que para um agente comum, utilizando-se a infraestrutura do ***Jason***.

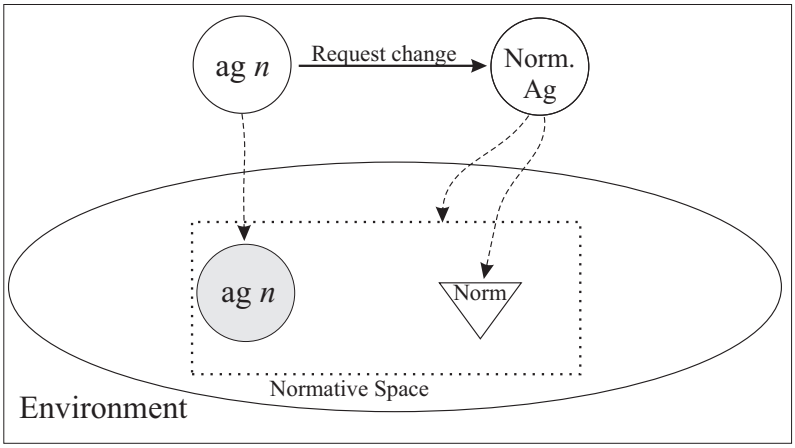

Figura 4.1: Alteração Dinâmica

- `create(Tipo,Id,[Property list])`, para criar um novo objeto normativo ou espaço normativo; exemplo:`create("NormObj","normObj1","position:library; condition:true")`;
- `set(Id,[Property list])`, para alterar propriedades de um item existente;
- `get(Id,Property])`, retorna via mensagem o valor da propriedade;
- `getIds`, retorna via mensagem os identificadores de itens existentes;
- `implements(NormSpace.Role, Agent))`, para incluir relação de implementação de um agente e um papel do espaço normativo;
- `remove(NormSpace.Role, Agent))`, para remover relação de implementação de um agente e um papel do espaço normativo;
- `getImplementation(NormSpace)`, retorna as relações de implementação existentes no espaço normativo.

4.7 Considerações Finais

Na área de pesquisa dos sistemas multiagente normativos, vários aspectos sofisticados de sistemas normativos são apresentados, tal como um modelo explícito de sanções e *enforcement* das normas, além de uma base formal para representação de normas e raciocínio (BOELLA; TORRE; VERHAGEN, 2007; LOPEZ; LUCK; D'INVERNO, 2007; FORNARA; COLOMBETTI, 2007). O modelo apresentado é um primeiro passo para futuramente permitir a implementação de estruturas mais complexas. Sendo a vantagem desta abordagem a possibilidade de ser diretamente utilizada no desenvolvimento de SMA práticos. A partir disso, planeja-se incorporar gradativamente aspectos mais sofisticados de sistemas normativos no futuro.

A *Infraestrutura Normativa* tem como função prover os meios para informar os agentes sobre normas em um contexto espacial específico, permitindo aos agentes raciocinar sobre normas que estes podem não ter conhecimento prévio. Assim, a infraestrutura normativa não possui como característica específica a tarefa de garantir o cumprimento das normas, porém pode oferecer os meios para que agentes específicos tenham acesso a informações necessárias para verificar a aderência às normas por parte de outros agentes, tomando as atitudes necessárias caso seja este seu papel na organização. Como o principal objetivo do MAS-SOC é prover meios para possibilitar o desenvolvimento de simulações sociais com agentes cognitivos, o comportamento de não seguir as normas seria desejável (por parte do projetista da simulação) para poder observar como os outros agentes (realizando seus respectivos papéis organizacionais) reagiriam em tais situações.

Além disso, a existência de uma infraestrutura normativa, tal como a apresentada neste capítulo implica em características e questões que merecem um estudo mais aprofundado. Algumas das principais implicações que percebidas são apresentadas a seguir.

4.7.1 Raciocínio Normativo Limitado pela Percepção

Dado que uma norma publicada através de um objeto normativo é apenas acessível enquanto o objeto normativo é acessível, o raciocínio normativo referente àquela norma é limitado pelos limites da possível percepção do objeto normativo. Ou seja, quando um agente não segue uma norma que se espera ser seguida em um dado espaço normativo, pelo menos três razões podem ser usadas para explicar o fato: a primeira seria que o agente não percebeu o objeto; a segunda seria que o agente percebeu o objeto mas não foi capaz de compreender o conteúdo normativo do objeto; a última seria que o agente percebeu corretamente o objeto e seu conteúdo, mas decidiu não seguir a norma.

O problema de aderência às normas em situações com objetos normativos, portanto, deve levar em conta não apenas a possibilidade que agentes decidam seguir ou não as normas, mas também a possibilidade dos agentes não serem capazes de perceber corretamente os objetos normativos e seus conteúdos. Questões como responsabilidade e outras relacionadas à obediência à normas, incorporam portanto, não apenas os aspectos usuais de racionalidade e de afetividade, mas também questões relacionadas à percepção física em ambientes concretos. A abordagem apresentada, através da ligação do ambiente, organizações e sistemas normativos, destacou a importância de tais questões.

4.7.2 Adequação da Distribuição dos Objetos Normativos

Questões relacionadas à adequação da distribuição dos objetos normativos em ambientes normativos podem surgir, quando se trata com o tipo de infraestrutura normativa proposto. Ou seja, garantindo que o conjunto de objetos normativos está bem distribuído

e distribuído de maneira satisfatória é uma questão a ser resolvida pelos agentes responsáveis pela criação de objetos normativos.

Além disso, a regulação de exceções normativas também é feita com os objetos normativos, por exemplo “Proibido fumar neste recinto, exceto nas áreas destinadas à fumantes”. Portanto, garantir que a estrutura hierárquica das normas operando em um dado espaço normativo é acessível e compreensível para os agentes presentes ou que fazem uso do espaço é uma questão para os agentes que regulam o espaço.

5 MODELO MAS-SOC

No Modelo MAS-SOC, o raciocínio dos agentes é especificado em uma versão estendida do AgentSpeak(L), linguagem para programação de agentes BDI introduzida em (RAO, 1996). A interpretação é feita utilizando o ***Jason***, plataforma de agentes baseada em JAVA (BORDINI; HÜBNER; VIEIRA, 2005), uma apresentação básica sobre o ***Jason*** e AgentSpeak(L) podem ser encontrados no Apêndice C. Os ambientes onde os agentes se situam são especificado na linguagem ELMS, linguagem projetada para a descrição de ambientes multiagentes. Neste modelo, a descrição explícita do ambiente é uma parte importante na modelagem de um SMA, que até recentemente, era muito pouco abordado e considerado como "dado" (WEYNS et al., 2005) ao invés de ser uma parte essencial na construção de SMA.

Em (OKUYAMA; BORDINI; ROCHA COSTA, 2007a), foi apresentado um conjunto de extensões para introduzir uma infraestrutura normativa em um ambiente físico simulado. Através do uso desta infraestrutura, o ambiente pode ser integrado à estruturas de uma organização. É importante enfatizar que as extensões introduzidas na linguagem de descrição de ambientes não permite a descrição completa de organizações. Os conceitos de *objetos normativos* e *espaços normativos* adicionados à linguagem servem como uma forma para preencher a lacuna na descrição de ambientes e as estruturas organizacionais, permitindo assim uma melhor integração de uma organização em um ambiente.

A figura 5.1 mostra os principais componentes do MAS-SOC. Usando os códigos AgentSpeak, o interpretador ***Jason*** realiza para cada agente, seu processo de raciocínio. Os agentes interagem no ambiente através de "corpos" definidos na especificação do ambiente em ELMS, juntamente com outros elementos do ambiente. Existem também um conjunto de planos AgentSpeak (uma biblioteca de planos usuais) que são utilizados para facilitar a programação das simulações, representados na figura pela caixa *aspk lib*. Com a expansão do modelo MAS-SOC, formam introduzidas as estruturas organizacionais, que possui um agente dedicado para seu controle, permitindo a qualquer agente a consultar informações a este, que responderá de acordo com as permissões de cada agente. Outra estrutura seria relacionada a infraestrutura normativa, a partir da especificação da infraestrutura normativa, um agente específico instancia os objetos normativos e espaços normativos (retângulo pontilhado) no ambiente, que pode ser atualizada durante a realização da simulação, através de solicitações ao *Norm. ag*.

Apesar de requerer uma forma diferente de modelar um SMA, este modelo não exclui a possibilidade do uso em conjunto com outras metodologias, permitindo ao projetista utilizar uma metodologia de sua preferência. É possível modelar um SMA que tenha um ambiente ELMS utilizando diversas abordagens: iniciando pelas estruturas organizacionais (*top-down*), ou iniciando pelos agentes e suas interações (*bottom-up*). Em ambas abordagens, a modelagem das estruturas organizacionais e o raciocínio dos agentes irão

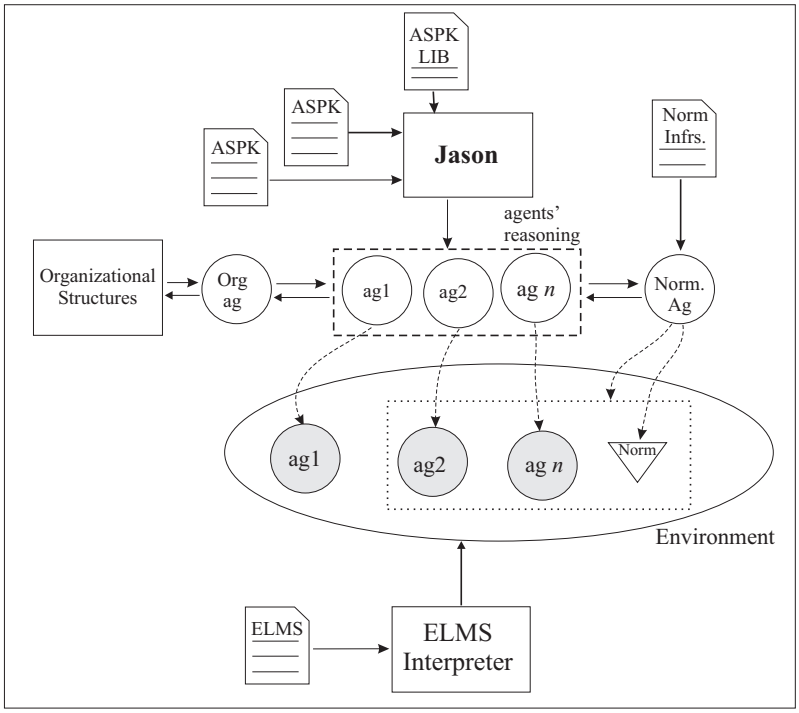

Figura 5.1: Componentes MAS-SOC

precisar de um *ajuste fino* para que atinjam os objetivos desejados. Tanto o *ajuste fino* do raciocínio dos agentes quanto do modelo organizacional podem se beneficiar de uma estrutura estável no modelo. Para isto, o modelo faz uso da descrição explícita de ambiente feita com a linguagem ELMS. O ambiente é uma parte essencial de um SMA e apesar de poder ter características muito dinâmicas, no que se refere à modelagem, seria a parte mais *estável* do sistema.

5.1 Ligando Organizações e Ambientes

É importante notar que a infraestrutura normativa agregada à descrição de ambiente não é uma abordagem para modelar organizações, apesar de ser possível utilizar para este fim, tal organização ficaria restrita a um modelo simplificado. Com a definição de *espaços normativos*, onde um conjunto de papéis organizacionais podem ser situados, pretende-se preencher a lacuna conceitual entre a maneira usual em que as organizações e ambientes são modelados. A partir dos conceitos associados à infraestrutura normativa introduzidos neste trabalho possibilitam a conexão de definições de organizações e ambientes. Além disso, a distribuição de informação organizacional e normativa pode facilitar a modelagem de grandes organizações.

Na figura 5.2 são apresentadas duas formas de integração entre ambiente e organizações. As organizações podem ser integradas diretamente aos agentes individualmente, mas nesta figura é mostrada apenas a integração entre ambiente e organização.

As duas formas básicas de ligação de organizações e um ambiente no modelo MAS-SOC são as seguintes:

Estática: como mostrado do lado esquerdo da Figura 5.2, a partir de uma descrição externa organização, o projetista pode modelar conceitualmente a estrutura normativa para refletir uma imagem estática da organização, convertendo estruturas organi-

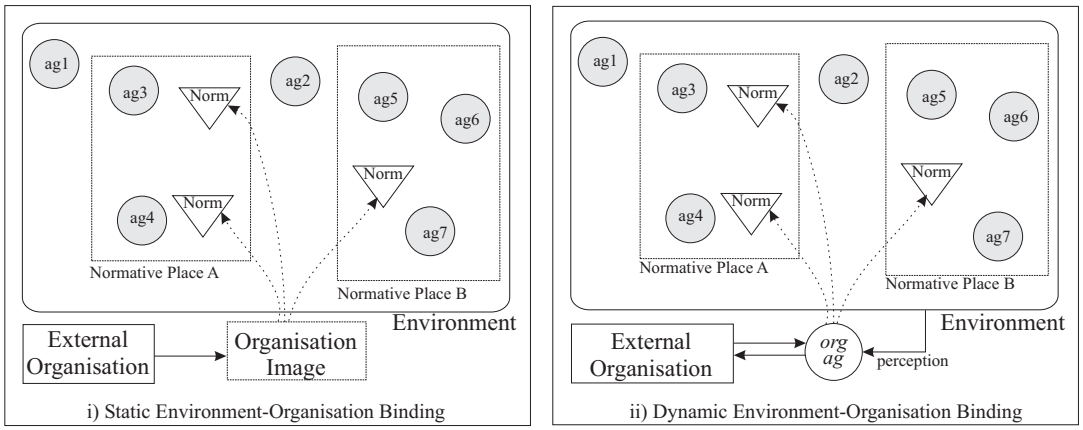

Figura 5.2: Ligação entre Organizações e Ambiente

zacionais em *papéis relativos* e relações organizacionais. Esta imagem existiria apenas conceitualmente ou na documentação, sem estruturas específicas que a implementam. Neste caso, os papéis seriam inserido nos *espaços normativos* enquanto as relações e os comportamentos organizacionais são prescritos em normas contidas nos objetos normativos, de modo a refletir a imagem conceitual da organização.

Dinâmica: como mostrado no lado direito da Figura 5.2, o agente *'org ag'*, obtém as informação disponível na descrição da organização e dinamicamente atualiza a estrutura normativa no ambiente. O agente que recebe as percepções do ambiente pode utilizar esta informação como *feedback* para a ferramenta externa de representação de organizações que poderá alterar a estrutura organizacional. O agente *'org ag'* pode fazer parte da simulação ou não, de acordo com a necessidade da aplicação; da mesma forma que o técnico de um time usualmente não faz parte do jogo, mas pode mudar a organização de seu time dinamicamente.

As formas de ligação entre ambiente e organizações, mencionadas acima, podem ser um complemento para uma abordagem organizacional existente. Usualmente, em abordagens de SMA centradas nas organizações, as organizações se conectam às *mentes* dos agentes, tal conexão não é conflitante com a abordagem apresentada para integrar organizações e ambientes. Mesmo quando as estruturas organizacionais têm conexões diretas com as *mentes* dos agentes, acredita-se que a conexão indireta com os agentes pode ser bastante útil. Utilizando-se estas formas simples de integração, seria possível integrar o ambiente com a maioria das abordagens existentes para organizações em SMA, como (VÁZQUEZ-SALCEDA; DIGNUM; DIGNUM, 2005; HÜBNER; SICHMAN; BOISSIER, 2002; GARCIA-CAMINO; NORIEGA; RODRÍGUEZ-AGUILAR, 2005). É claro que simplificações podem ser necessárias e algumas funcionalidades destas abordagens de organizações podem não ser capturadas pelo uso de tal integração.

5.1.1 Modelo Pop-Org

O modelo Pop-Org (Populations and Organisations) (DEMAZEAU; COSTA, 1996) é baseado em conceitos apresentados por (DEMAZEAU, 1995), define uma população de um SMA como um conjunto de agentes que o habitam, juntamente com o conjunto de comportamentos que os agentes são capazes de realizar e o conjunto de todos os processos de interações que os agentes podem estabelecer entre si ($Populacao(SMA) = Agentes + Interacoes$).

No modelo *Pop-Org*, uma organização invariante no tempo pode ser definida como um conjunto de papéis organizacionais e relacionamentos organizacionais. Sendo os papéis responsáveis por capturar as características do processo global que estão ligados aos processos locais dos agentes. Enquanto os relacionamentos seriam responsáveis por capturar as características do processo global que correspondem as influências mútuas entre os agentes. Desta forma uma organização poderia ser definida como $Org = (Ro; Li)$, onde Ro é o conjunto de papéis organizacionais existente e Li é um conjunto de relacionamentos organizacionais que representam as influências mútuas entre os papéis da organização. Esta forma de organização baseia-se na idéia que a organização de um sistema pode ser considerada como uma abstração da presença de agentes específicos no sistema, podendo ser definida como um conjunto de papéis organizacionais e relacionamentos organizacionais; onde os papéis capturam as características de processos globais que são partes do processo local dos agentes, enquanto os relacionamentos capturam as características do processo global que correspondem às influências mútuas entre os agentes (DEMAZEAU; COSTA, 1996).

Desta forma, para completar o modelo de um sistema, a população e a organização de um sistema devem estar adequadamente relacionadas, de forma que a organização seja implementada pela população. Assim, uma relação de *implementação* de uma organização *Org* por uma população *Pop* seria composta por agentes de *Pop* implementando papéis de *Org*; juntamente com relacionamentos (*links*) de *Org* sendo implementados por processos de interação de *Pop*.

Baseado nestes conceitos apresentados, criou-se uma forma simplificada para definir organizações baseadas no modelo Pop-Org para ser utilizada no contexto do Modelo MAS-SOC, apresentada a seguir.

5.1.2 Modelo Mínimo de Organizações

A fim de possibilitar no Modelo MAS-SOC a inclusão de conceitos que não estão espacialmente situados, foi incluída essa simplificação do modelo Pop-Org para exemplificar o relacionamento de um estrutura organizacional não espacial dentro do modelo. Esta se ligaria à simulação conforme apresentado na Seção 5.1, onde se menciona a ligação dinâmica do modelo MAS-SOC à uma organização.

Simplificando as definições do modelo Pop-Org para o contexto do modelo MAS-SOC, uma organização mínima poderia ser representada por um conjunto de papéis e um conjunto de normas que representariam as relações entre os papéis desta organização. Desta forma, para cada relacionamento entre papéis seria definido um conjunto de normas.

Este modelo mínimo pode ser utilizado para representar características de organizações de maneira *não-espacial* de forma a capturar os aspectos que seriam melhor definidos desta maneira. Assim pode-se definir a organização não-espacial, relacionando-se diretamente com sua imagem projetada na infraestrutura normativa espacialmente contextualizada.

A realização da implementação de uma organização é realizada basicamente pelos mecanismos de atribuição de papéis organizacionais para os agentes e o estabelecimento de relações entre agentes desempenhando os papéis organizacionais. Enquanto a dinâmica estrutural da organização seriam as operações de criação e deleção de papéis e normas relacionadas aos papéis. Estas operações podem ser realizadas pelos agentes definidos como *owner* da organização, que poderão incluir outros agentes nesta função, enquanto outros agentes poderão apenas receber as informações da estrutura organizacional.

Este modelo de organização mínimo baseado nos conceitos do modelo Pop-Org, se

concentra apenas na representação utilizada para representação de organizações invariantes no tempo. Desta forma, estariam representados apenas os estados momentâneos da organização, sem os mecanismos e a semântica que poderiam explicar o funcionamento dinâmico da organização, tais como os apresentados em (COSTA; DIMURO, 2008). Assim, neste modelo simplificado, o estado armazenado da organização seria composto pela definição de uma organização e relações de implementação de papéis por agentes:

```
Organisation familyA{
 roles = {father, mother, son, daughter};
   owner = {ag1};

   //norma não contextualizada:
   Norm norm1 = {has(father,obligation(educate(son),true))))} ;
}

implements(ag1, familyA.father);
implements(ag2, familyA.son);
implements(ag3, familyA.son);
```

No trecho acima está definida um organização simples com apenas quatro papéis e a definição de uma norma que definiria uma das formas de relacionamento entre os papéis. Em seguida, uma lista com as relações de implementação de papéis pelos agentes.

Além disso, através da troca de mensagens com o controle da estrutura de organização, é possível para o agente habilitado alterar a estrutura da organização, como criar papéis e normas. Os agentes que possuem direitos para alterar a estrutura de uma organização não-espacial podem ser parte da simulação ou não, conforme o exemplo da figura 5.2, de acordo com as características da simulação.

```
familyA.createRole("inLaw");
familyA.createNorm("norm2","has(father,right(use(car),true))");

createOrg("familyB");
```

Também é possível aos agentes criar novas organizações, sendo o agente que cria a organização designado como *owner*. Desta forma, é possível a existência de diversas organizações, que possivelmente podem ter normas conflitantes. Assim, a influência das diversas organizações é dada pela decisão dos agentes em seguir suas normas e implementar seus papéis.

Acesso Dinâmico à Estrutura

Para alteração da estrutura organizacional durante a simulação, os agentes podem enviar comandos de maneira semelhante a utilizada na infraestrutura normativa (Seção 4.6). Da mesma forma, um processo distinto, que implementa a mesma interface de comunicação de um agente, recebe as mensagens dos agentes, verifica se o agente que enviou a mensagem tem direitos suficientes para alterar a organização e realiza a atualização, conforme a Figura 5.3. As mensagens são enviadas da mesma maneira que para um agente comum, utilizando-se a infraestrutura do ***Jason***.

- `create(Id,[Property list])`, para criar uma nova organização;
- `set(Id,[Property list])`, para alterar propriedades de uma organização existente;
- `get(Id,Property])`, retorna via mensagem o valor da propriedade;

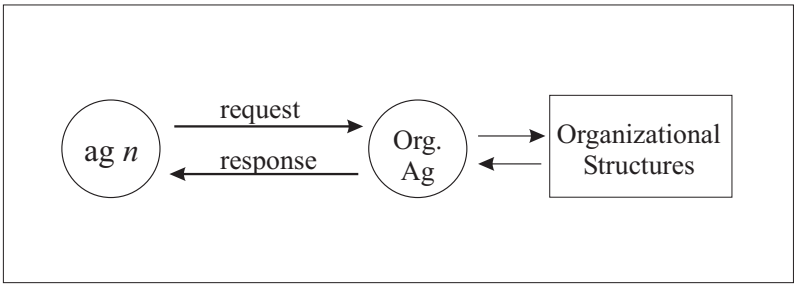

Figura 5.3: Acesso à Estrutura de Organizações

- `getIds`, retorna via mensagem os identificadores das organizações existentes;
- `implements(Org.Role, Agent))`, para incluir relação de implementação de um agente e um papel de uma organização;
- `remove(Org.Role, Agent))`, para remover relação de implementação de um agente e um papel da Organização;
- `insert(Org, Norm))`, para inserir norma da Organização;
- `removeNorm(Org, Norm))`, para remover norma da Organização;
- `getImplementation(Org)`, retorna as relações de implementação existentes na organização.

5.2 Componentes SMA x Componentes MAS-SOC

O modelo MAS-SOC possui componentes para desenvolvimento de simulações baseada em SMA. Cada componente aborda partes diferentes de um SMA, abrangendo grande parte dos componentes de um SMA, permitindo a programação do SMA sem que seja necessário grandes conhecimentos de programação tradicional. Apesar de não ter um componente específico para definição de organizações, é possível definir estruturas organizacionais simplificadas ou integrar uma outra abordagem de organização, conforme apresentado na seção 5.1.

Usualmente, os componentes de um SMA, tais como agentes, organizações e ambiente são representados e modelados como entidades independentes, porém acredita-se que estas entidades possuem conexões entre si, conforme a Figura 5.4.

A Figura 5.4 mostra os componentes que compõem um SMA. Abaixo, são identificadas as estruturas de cada região do diagrama, relacionando ao componente MAS-SOC que aborda tais estruturas, mesmo que de maneira parcial.

A: Seção específica dos agentes, nesta seção pode-se encontrar o raciocínio do agente, aprendizado, representação do conhecimento, planejamento, tomada de decisão. No MAS-SOC, esta seção seria abordada parcialmente através do uso do AgentSpeak interpretado pelo ***Jason***;

B: Interseção entre agentes e ambiente, onde pode-se citar os *corpos* dos agentes, incluindo os sensores e efetores dos agentes. No MAS-SOC, esta seção é representada através da descrição em ELMS dos *corpos* dos agentes e a definição de percepções e ações;

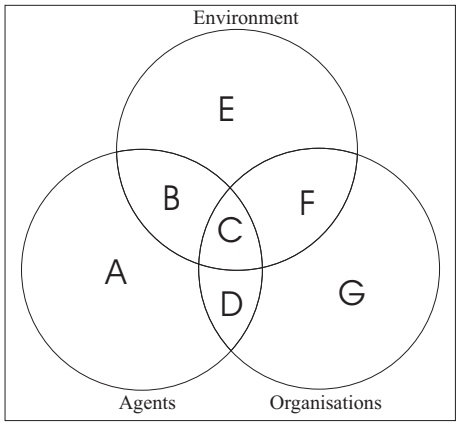

Figura 5.4: Componentes SMA

C: Interseção entre agentes-organizações-ambientes. No MAS-SOC seria representado por instâncias de agentes realizando papéis *relativos* em um escopo espacial determinado (*espaço normativo*) regulado por normas informadas através de instâncias de objetos normativos;

D: Interseção entre agentes e organizações, região onde pode-se encontrar a descrição do comportamento de papéis que podem ser implementados por agentes. No modelo MAS-SOC estes comportamento dos agentes poderiam ser descritos através de normas que delimitariam as formas de atividade do agente, conforme apresentado na Seção 5.1.2. Estes itens poderiam ser definidos também de maneira implícita nos agentes ou através da associação com uma abordagem para definição de organizações, como visto na seção 5.1;

E: Seção específica do ambiente, que seriam as estruturas físicas do ambiente, tal como a representação espacial, além de recursos e objetos presentes no ambiente. No MAS-SOC, estas estruturas são descritas através do ELMS, podendo a representação espacial ser realizada através de um grafo ou uma grade;

F: Interseção entre ambiente e organizações, onde pode-se encontrar estruturas organizacionais regulando a atuação dos agentes dentro de um escopo espacial definido ou estruturas organizacionais localizadas em um escopo espacial definido. Sendo representado no MAS-SOC pela definição de papéis relativos definido para um escopo específico ou associados aos mesmos, além dos espaços normativos e objetos normativos e as normas contidas nestes;

G: Seção específica das organizações, onde pode-se encontrar estruturas organizacionais das mais diversas; nestas, as mais usuais seriam os grupos, papéis e instituições. No modelo MAS-SOC, é possível representar tais estruturas através do modelo mínimo de organizações apresentado na Seção 5.1.2, ou integrar com outro modelo existente de organizações.

5.3 Simulação do Ambiente

A simulação do ambiente é realizada por um módulo do sistema que controla o acesso e mudanças realizadas na estrutura de dados que representa o ambiente. A estrutura de

dados que representa o ambiente é gerada pelo interpretador ELMS para a especificação ELMS dada como entrada.

Nesta seção é apresentada, de maneira geral, a forma como um ambiente é simulado. A Figura 5.5 mostra alguns estados de atividade do ambiente do lado esquerdo, um modelo simplificado dos estados de atividade do agente do lado direito e as principais comunicações entre agentes e ambiente no centro. Os estados do ambiente são descritos abaixo:

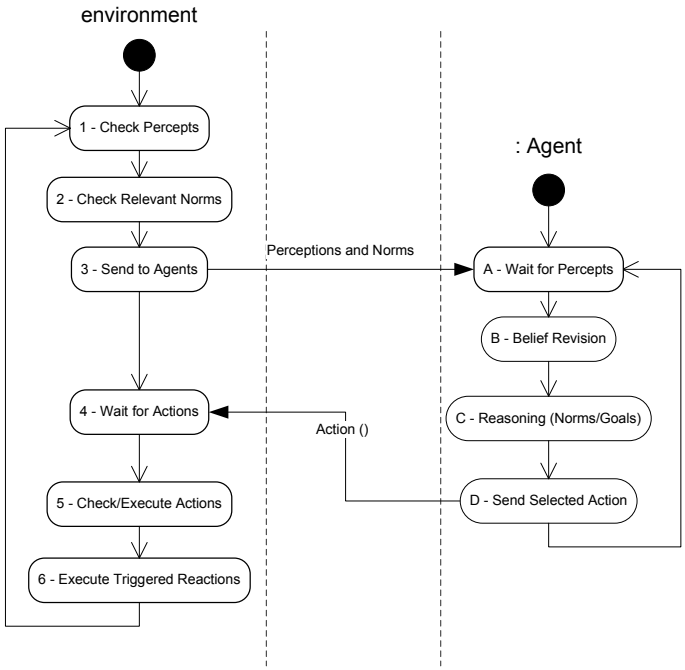

Figura 5.5: Ciclo Ambiente-Agente

1. Para cada agente, o ambiente verifica a classe do agente e as percepções que estão habilitadas para este. Então, são verificadas quais percepções têm suas pré-condições satisfeitas e coleta o conjunto de valores de propriedades resultante para o agente.

2. Os objetos normativos (conforme definidos no Capítulo 4), que estão no mesmo espaço normativo do agente são verificados para encontrar informações normativas relevantes para o agente. Se as condições forem satisfeitas, as normas são armazenadas para o serem enviadas ao agente.

3. O ambiente envia percepções e informações normativas específicas para cada agente.

4. O ambiente aguarda pela ação selecionada por cada agente.

5. O ambiente verifica para cada agente se a ação selecionada está habilitada para o agente que solicitou e também se as pré-condições desta ação estão satisfeitas. Caso positivo, o ambiente realiza as mudanças especificadas na definição da ação.

6. O ambiente verifica quais reações dos objetos são ativadas pelo estado corrente do ambiente. As reações ativadas são executadas, mudando o estado do ambiente. Então o ciclo é reiniciado voltando ao Item 1.

As linhas gerais que o agente deve seguir para interagir com o ambiente simulado é descrita a seguir:

A. O agente aguarda até o recebimento de percepções e informações normativas do ambiente.

B. O agente poderá então comparar as percepções recebidas com sua representação interna do mundo para detectar mudanças que possam ser relevantes para seu curso de ação.

C. O agente irá selecionar uma ação para execução, baseado em seus próprios objetivos e normas percebidas. Tal decisão envolve o balanço de vários fatores, entre os quais pode-se mencionar alguns:

   - crenças sobre o ambiente;
   - objetivos, estados de mundo que o agente deseja realizar;
   - seleção de qual objetivo atingir primeiro;
   - seleção de qual normas seguir ou não;
   - resolução de conflitos entre normas e objetivos;

D. O agente envia a ação selecionada para a execução no simulador do ambiente.

5.4 Trabalhos Relacionados

Como o modelo MAS-SOC apresenta características destinadas à definição de estruturas organizacionais, seria possível considerá-lo uma alternativa para modelagem de organizações. Porém, como apresentado na Seção 2.2 e mencionado nos capítulos anteriores, a infraestrutura do normativa do modelo MAS-SOC não se sobrepõe a modelos de organizações. Seria totalmente possível e bastante proveitoso o uso do modelo MAS-SOC com abordagens como OMNI e $\mathcal{M}$OISE$^+$, visto que estas abordagens não tratam de questões relacionadas ao ambiente diretamente, sendo o modelo MAS-SOC complementar a estes modelos.

Em relação as Instituições Eletrônicas, o foco de aplicações difere do foco do modelo MAS-SOC, visto que as IE têm como aplicações sistemas onde não se pode permitir a violação e normas, tais como sistemas para leilões e vendas eletrônicas. Ainda assim, não existe uma grande sobreposição entre os modelos, permitindo seu uso conjunto sem grandes necessidades de adaptações.

Em (GARCÍA-CAMINO et al., 2006), os autores apresentam os agentes *governor* que têm como função garantir que *agentes externos* cumpram suas obrigações sociais durante sua participação na instituição eletrônica. Os agentes externos interagem no ambiente através dos agentes *governor*, que também informam aos agentes sobre as normas e ações possíveis. Neste modelo, cada agente possuiria um agente *governor* associado. Estes agentes *governor* diferem dos agentes supervisores apresentados anteriormente, já que os agentes supervisores não têm como função básica impedir infrações das normas, os agentes *governor* agem como uma espécie de filtro impedindo que o agente externo execute ações que contrárias às normas existentes na instituição eletrônica.

De maneira similar às instituições eletrônicas, o trabalho de *computacional institutions* (RUBINO; OMICINI; DENTI, 2005), onde são definidas organizações virtuais reguladas por normas constitutivas e normas regulativas. Nas instituições computacionais, a

modelagem organizacional usa a abstração dos artefatos de coordenação como peças para construção, de modo bastante similar à noção de objetos normativos em organizações espacialmente distribuídas, mas ainda mantendo implícitas nos artefatos de coordenação o conteúdo normativo impostos aos agentes.

Em (FERBER; MICHEL; BÁEZ-BARRANCO, 2005), os autores apresentam o modelo AGRE, uma extensão ao modelo AGR. Estas extensões permitem a definição de grupos e estruturas físicas (espaços), como especializações de um espaço genérico. Neste modelo, baseado nos conceitos de Agentes-Grupos-Papéis, os grupos são conjuntos de agentes que compartilham características comuns, sendo utilizados como contexto para padronizar atividades e para particionar organizações. O modelo AGRE trata de maneira distinta o ambiente social (grupos) e os ambientes físicos (geométricos). No ambiente físico os agentes interagem através de *corpos* enquanto no ambiente social através de papéis, porém não existem relacionamentos diretos entre papéis e corpos na abordagem. Enquanto na abordagem apresentada nesta tese, tem-se os elementos que compõem o SMA definidos de maneira muito integrada.

5.4.1 Artefatos

A noção de *artifacts* (VIROLI; OMICINI; RICCI, 2005) e *coordination artifacts* (OMICINI et al., 2004) se assemelham em alguns aspectos à noção de objetos normativos. Como definido em (OMICINI et al., 2004), os artefatos de coordenação são abstrações para aumentar a automação de atividades de coordenação, sendo peças para criar efetivamente ambientes de trabalho compartilhados. Estes são definidos como abstrações de tempo de execução (*runtime*) que encapsulam e fornecem um serviço de coordenação para os agentes. Os artefatos (VIROLI; OMICINI; RICCI, 2005), são apresentados como uma abstração para representar ferramentas, serviços, objetos e entidades em um ambiente multiagente. Como peças para realizar a modelagem do ambiente, artefatos encapsulam as características do ambiente como serviços a serem usados pelos agente. O objetivo principal de um artefato de coordenação é ser utilizado como uma abstração de um serviço de coordenação ambiental oferecido para os agentes. Todavia, os artefatos de coordenação expressam regras apenas implicitamente, através de seu efeito prático nas ações dos agentes, sendo que seu impacto normativo não requer nenhum raciocínio normativo por parte dos agentes.

Cada artefato é definido através de uma interface de uso (*Usage Interface*), instruções de operação, uma função e estrutura interna. Na interface de uso são definidas as funções que dão acesso às funcionalidades do artefato. Na função do artefato é implementada a funcionalidade desejada do artefato, através do uso de sua estrutura. A especificação geral de um artefato é bastante ampla, o que permitiria *encapsular* todo o ambiente em artefatos.

No MAS-SOC, ao invés de ter um noção geral de objetos que por suas características físicas facilitam a coordenação, os *objetos normativos* são utilizado especificamente para armazenar informação simbólica que pode ser interpretada pelos agentes, de forma que estes possam conhecer as normas que devem ser respeitadas em um local determinado. Os objetos normativos têm a vantagem de manter aberta a possibilidade da autonomia do agente, como sugerida pelos autores em (CASTELFRANCHI et al., 1999), onde agentes são, em princípio, capazes de decidir se seguem normas ou não enquanto tentam efetivamente atingir seus objetivos. Isto é algo que não é possível se as ações de um agente forem limitadas por normas implícitas garantidas por mecanismos de coordenação. Outra importante diferença é que os objetos normativos são espacialmente distribuídos sobre

um ambiente *físico*, dentro do escopo espacial onde a norma se aplica e ligado à organização presente neste espaço. Enquanto o objetivo dos artefatos de coordenação é remover o fardo da coordenação dos agentes, este trabalho tenta simplificar o modo como os projetistas podem guiar o comportamento individual dos agentes conforme estes se movem em um ambiente onde diferentes organizações existem. Isto permite aos agentes adaptarem seu comportamento em diferentes contextos sociais. Apesar das diferenças conceituais, é possível encapsular objetos normativos em artefatos, ou seja é possível implementar artefatos para cumprirem funções de objetos normativos, sendo a funcionalidade destes artefatos informar os agentes sobre as normas.

De maneira similar aos *artifacts*, em (ALDEWERELD et al., 2006), os autores apresentam uma abordagem onde as normas são convertidas em protocolos, ou seja, para cada norma abstrata são criados um conjunto de objetivos que se seguidos resultariam no cumprimento do protocolo, resultando em uma representação de alto-nível da interpretação da lei, adicionados de restrições operacionais. Para cada um destes objetivos, parciais e finais, são definidos o que chamam de *landmarks* e também são definidos *landmarks negativos*. Atingir um *landmark* significa receber um *feedback* se o curso de ações está correto ou incorreto.

5.5 Considerações Finais

A integração entre agentes e ambiente, através da modelagem explícita da percepção, efetores e características *físicas* de agentes implica na verificação de coerência na definição do modelo de raciocínio do agente e o modelo do ambiente (e.g. uma ação selecionada pelo raciocínio do agente, deve ser disponível no modelo deste agente no ambiente). Na integração entre ambiente e estruturas organizacionais, com a contextualização de normas em seus respectivos escopos espaciais, também são necessárias diversas verificações a fim de garantir a corência do modelo como um todo. À primeira vista, isto pode parecer uma deficiência a ser tratada pelo modelo MAS-SOC. Porém, tais verificações de coerência do modelo não são explícitas em outros formas de modelagem pois estes modelos tratam os elementos de um SMA como entidades distintas e independentes. No modelo MAS-SOC, ao integrar as entidades de maneira explícita, tornou-se claro a necessidade de verificar a coerência na definição das entidades para permitir a interação entre estas, enquanto em outros modelos algumas destas incoerências poderiam não ser percebidas. Entre as possibilidades de incoerências pode-se citar:

- Chamada de ação inexistente. É definido no raciocínio do agente uma chamada para a realização de ação que não disponibilizada pelo ambiente modelado;

- Raciocínio dependente de percepção não associada ao agente. O raciocínio do agente pode ser definido tendo como suporte a expectativa de receber determinada informação do ambiente, que não está definida como perceptível para este agente.

- Atividade incompatível com espaço físico. A atividade definida pelas estrutura organizacional não é compatível com o espaço físico. Isto pode ocorrer pela existência de regras ou normas que impedem sua realização ou pela ausência de recursos necessário para realização da atividade;

- Atividade incompatível com o agente. A organização atribui a um agente uma atividade que não é compatível com as habilidades do mesmo.

Ao apresentar a infraestrutura normativa salientou-se a existência de intersecções entre as entidades básicas de um SMA, conforme apresentado na Seção 5.2, especialmente o item C, contrastando-se com o modelo de SMA tradicional, apresentado na Figura 2.2, na Seção 2.4.

Outro aspecto muito discutido na literatura de SMA que deve ser salientado é como lidar e como tratar SMA abertos e heterogêneos, onde agentes programados de maneira separada do sistema podem entrar e interagir com os agentes presentes no sistema de maneira a se manter a coerência do sistema e permitir que os agentes atinjam seus objetivos. No modelo MAS-SOC, essa situação pode ser tratada da seguinte forma:

- **Avatares:** através da definição de classes de 'corpos dos agentes', associa-se ao agente recém-chegado um 'avatar' que define basicamente suas características de percepção e ação dentro do ambiente. Através das definições de ações e percepções, são definidas as principais *regras físicas* do ambiente que regulam as percepções e ações do agente. Desta maneira, pode-se definir, por exemplo, que um agente não pode ver através de uma parede (limitação física da percepção) ou atravessar uma parede, limitando os parâmetros de uma ação. Tanto as ações como percepções estão associadas à precondições que são verificadas antes de sua execução, impossibilitando sua realização em caso indevido. Desta forma, é garantida a consistência e integridade do sistema, seja este um SMA fechado ou aberto. Diferentemente, no caso de normas de uma estrutura organizacional, um agente poderá raciocinar sobre uma determinada norma e decidir pela violação, neste caso este poderá enfrentar sanções da organização, porém sem implicações para o funcionamento, integridade e coerência do sistema.

- **Adoção implícita de papel:** ao entrar em um espaço normativo, o agente recebe um papel de acordo com o seu perfil de atividade ou um papel *default*, o que irá implicar em um determinado conjunto de direitos e deveres que serão informados através dos objetos normativos. Este tratamento seria bastante adequado a um agente recém-chegado no sistema, sendo que este poderá adotar explicitamente um papel específico caso possua os requisitos especificados.

5.5.1 Definindo o Raciocínio dos Agentes

Conforme já mencionado, no modelo MAS-SOC, faz se uso da linguagem AgentSpeak em conjunto com o ***Jason*** para a definição do raciocínio dos agentes. O uso do ***Jason*** permite a execução dos agentes de forma distribuída através de estruturas disponíveis juntamente com o interpretador, além de disponibilizar a infraestrutura de comunicação entre os agentes.

Desta forma, a interação inter-agentes pode ser realizada tanto diretamente quanto via ambiente. Assim, caso a comunicação agente-agente não seja importante para a modelagem do ambiente, esta poderá ser realizada apenas através da infraestrutura de comunicação do ***Jason***. Caso contrário, o projetista pode implementar formas de comunicação indiretas através do ambiente, ou registrar as comunicações diretas em propriedades do ambiente.

5.5.2 Organizações Não-Espaciais

Neste capítulo foi apresentada um modelo mínimo para definição de estruturas organizacionais não-espaciais. Esta estrutura tem como objetivo armazenar e representar informações organizacionais que sejam melhores representadas desta forma. Assim,

tornou-se possível representar estruturas organizacionais não-espaciais dentro do modelo MAS-SOC, de forma não conflitante com a infraestrutura normativa do modelo.

6 ESTUDO DE CASO

Neste capítulo são apresentados dois estudos de caso, baseados em cenários utilizados na literatura. Os principais pontos da modelagem destes cenários com o uso da abordagem MAS-SOC são apresentados, enfatizando os aspectos que caracterizam a abordagem.

6.1 Estudo de Caso 1: Cenário Normativo

Neste primeiro estudo de caso, é apresentado apenas uma situação simples de uso de objetos normativos e espaços normativos. Este cenário é baseado no cenário apresentado por Conte, Castelfranchi e Paolucci (1998), um cenário no qual os agentes estão inseridos em um ambiente onde eles podem comer a comida que encontram, atacar outro agente para tomar sua comida ou se mover em busca de comida.

6.1.1 Cenário Modelado

Neste cenário, um agente é dono de qualquer item de comida livre que esteja próximo a ele, a uma distância de até 2 células. A quantidade comida no ambiente é fixa, após a comida ser consumida (o que leva 2 ciclos), ela desaparece e reaparece em outra posição da grade. Os agentes podem *enxergar* comida em um raio de 1 célula, mas podem sentir a comida em um raio de 2 células. O espaço físico é representado por uma grade de 10×10 células. Em cada célula pode haver apenas um agente, o agente deve estar posicionado na mesma célula que a comida para consumi-la. Os agentes possuem uma força inicial, que utilizam para movimentar-se na grade e para atacar outros agentes.

As normas usadas neste cenário estão relacionadas essencialmente com a propriedade da comida e meios que prescrevem comportamento não-agressivo. No cenário original, as normas eram válidas uniformemente em toda a grade, ou seja, todo o espaço físico era tratado como um único espaço normativo; mas no cenário modelado, os objetos normativos são utilizados para definir múltiplos espaços normativos menores na grade.

Os ataques podem ocorrer quando exista um agente em uma célula com comida e outro em uma célula adjacente, o agente na célula adjacente pode realizar o ataque, que tem como resultado a comida movendo-se (ou mantendo-se) para a célula onde se encontra o agente com mais unidades de energia. Como as ações são definidas de maneira atômica (i.e. não podem ser interrompidas), a ação de comer deve ser realizada duas vezes para o consumo da comida e a reposição de energia. Isto possibilita que um agente seja *interrompido* enquanto está realizando a atividade de comer.

Uma versão simplificada da descrição do ambiente é dada a seguir. A descrição de ambientes em ELMS é armazenado em um formato XML, porém para tornar o código mais legível, foi adotado uma representação em pseudo-código apenas para demonstrar

os pontos principais da modelagem do ambiente em ELMS.

```
environment.name:="FOOD-CHASERS";
environment.grid(10,10);

Resource food{
        owner: string("none");
        id: integer;
        eaten : integer(0);
        appear: Reaction;
}

AgentBody agent{
    id: integer(SELF);
    power: integer(50);
    vision: PERCEPTION;
    sense_food: PERCEPTION;
    walk: ACTION;
    attack: ACTION;
    eat: ACTION;
}

Perception vision{
    cell[+0][+0].contents;     cell[+0][+1].contents;
    cell[+0][-1].contents;     cell[+1][+0].contents;
    cell[-1][+0].contents;
    cell[+0][+0].food.owner;   cell[+0][+1].food.owner;
    cell[+0][-1].food.owner;   cell[+1][+0].food.owner;
    cell[-1][+0].food.owner;
}

Perception senseFood{
    cell[+1][+1].food.id; cell[-1][-1].food.id;
    cell[-1][+1].food.id; cell[+1][-1].food.id;
    cell[+0][-2].food.id; cell[+0][+2].food.id;
    cell[-2][+0].food.id; cell[+2][+0].food.id;
}

Action eat(agentId: integer, FoodID:integer){...}
Action walk(x,y:integer){...}
Action attack(aggressor, victim ){ ...};
Reaction appear{...};
```

No trecho de código acima, são definidas a grade, a classe de recurso *food*, e a classe de corpos dos agentes permitidos na simulação. O corpo do agente é definido como sendo capaz de realizar três tipos de ações: `walk`, `attack`, e `eat`. A percepção `vision` permite ao agente verificar o conteúdo da célula corrente e as 4 células vizinhas, enquanto `sense_food` permite a este perceber a presença de instâncias de `food` em um raio de 2 células.

São definidas as diversas ações: a ação de comer, que deve ser realizada duas vezes para que a comida seja removida da célula e o agente receba as unidades de energia. As outras ações que aparecem são para movimentação e ataque. Uma reação é definida para a *comida*, quando uma é removida do cenário, uma nova instância é criada em outra posição da grade.

6.1.2 Infraestrutura Normativa

Neste exemplo a grade é divida em quatro espaços normativos de mesmo tamanho, tendo objetos normativos definidos e posicionados em três dos quatro quadrantes, conforme o trecho abaixo:

```
NormSpace upper-left{
    environment.grid[0..4][0..4];
    roles = {agent};
    roleAdoption{
       defaultRole : agent;
   }
}

NormSpace upper-right{
   environment.grid[5..9][0..4];
   roles = {agent};
   roleAdoption{
       defaultRole : agent;
   }
}

NormSpace lower-left {
    environment.grid[0..4][5..9];
   roles = {agent};
   roleAdoption{
       defaultRole : agent;
   }
}

NormObj norm-obj1 {
    type       := "prohibition";
    place      := {upper-left};
    norm       := "has(agent,prohibition(attack(SELF,AG),true))";
    issuedBy   := SYSTEM;
    sourceOrg  := SYSTEM;
    sourceRole := SYSTEM;
    addresses  := {role(agent)};
    condition  := true;
}

NormObj norm-obj2 {
    type       := "prohibition";
    place      := "upper-right"
    norm       := "has(agent,prohibition(eat(SELF,FOOD),
                         not owns(SELF,FOOD)))";
    issuedBy   := SYSTEM;
    sourceOrg  := SYSTEM;
    sourceRole := SYSTEM;
    addressees := {role(agent)};
    condition  := true;
}

NormObj norm-obj3 {
    type       := "prohibition";
    place      := "lower-left"
    norm       := "has(agent,prohibition(attack(SELF,AG),true))";
    issuedBy   := SYSTEM;
```

```
    sourceOrg  := SYSTEM;
    sourceRole := SYSTEM;
    addressees := {role(agent)};
    condition  := true;
}

NormObj norm-obj4 {
    type       := "prohibition";
    place      := "lower-left"
    norm       := "has(agent,prohibition(eat(SELF,FOOD),
                                  not owns(SELF,FOOD)))";
    issuedBy   := SYSTEM;
    sourceOrg  := SYSTEM;
    sourceRole := SYSTEM;
    addressees := {role(agent)};
    condition  := true;
}
```

O objeto normativo no exemplo acima são bastante simples, apenas para ilustrar como estes podem ser modelados neste modelo. Por exemplo, os objetos `norm-obj1` , e `norm-obj3` dizem que um agente não pode atacar outro agente, enquanto a *norm-obj3* diz que o agente não deve comer a comida que não seja de sua propriedade.

Desta forma, o comportamento esperado dos agentes será diferente nos quatro quadrantes do ambiente:

- no quadrante superior esquerdo, os agentes são informados a não roubar a comida que está em posse de outro agente, já que a norma situada indica que os agentes são proibidos de atacar outros agentes. Porém não há regulação sobre um agente comer a comida de outro que não esteja comendo no momento;
- no quadrante superior direito, os agentes estão proibidos de comer a comida que esteja rotulada como propriedade de um agente específico;
- no quadrante inferior esquerdo ambas as normas acima estão em vigor;
- no quadrante inferior direito, não há regras;

É importante notar que o termo `prohibition` é utilizado como o operador condicional deôntico, com dois argumentos: o primeiro é a ação que está proibida e o segundo argumento é a condição a ser verificada. No cenário modelado os agentes não são obrigados a seguir as regras mas eles podem usar a informação normativa para se monitorar mutuamente, podendo utilizar essa informação como entrada de um eventual sistema de reputação.

6.1.3 Raciocínio dos Agentes

O raciocínio básico dos agentes nesta simulação é bastante simples, de maneira geral seria da seguinte forma:

- Caso haja comida na célula corrente, o agente irá consumí-la;
- Caso contrário, o agente irá se mover para a comida vista, que não esteja sendo comida;
- Caso contrário, o agente irá se mover na direção da a comida detectada;

- Caso contrário, o agente poderá atacar outro agente para roubar sua comida;
- Se o agente não sabe onde há comida, move-se aleatoriamente;
- Se o agente não possui unidades de energia, fica parado.

Um agente que obedece as normas poderia ser implementado utilizando a biblioteca de planos (ver Seção 4.5), da seguinte forma:

```
+food(instance(F),position(+0,+0))
<- !+execute(eat(F));
   !+execute(eat(F)). //eat must be executed twice

+food(instance(F),position(X,Y))
  : not agent(instance(A),position(X,Y)
  <- walk(X,Y).

+food(instance(F),position(X,Y))
  : not agent(instance(A),position(X,Y)
  <- +!execute(attack(SELF,A)).

+agent(instance(SELF),position(X,Y))
    <- walk(random(0,10),random(0,10)).
```

No trecho abaixo, mostra como poderia se implementar um comportamento diferente. Um agente ao ser atacado, mesmo em uma área onde o ataque era permitido, pode decidir deliberadamente a se outorgar (incluir em suas crenças) o direito de revidar o ataque. Este direito terá prioridade sobre uma eventual proibição de ataque:

```
+attack(A,SELF)
    <- +has(SELF, right(attack(SELF,A),true)).

+has(SELF, OP(ACT,COND))
    <- +OP(ACT,COND).
```

Outra opção seria verificar se no espaço normativo existe uma norma ativa relativa a ataques, e apenas incluir a exceção apenas se o ataque era proibido no espaço normativo.

```
+attack(A,SELF)
    : has(A,prohibition(attack,Condition))
    <- +has(SELF, right(attack(SELF,A),true)).
```

Além da inclusão do plano seria necessário incluir mais um plano, para verificar a existência de direito, na base de planos destinados à manipulação de proibições:

```
    +!execute(Action)
         : not prohibition(Action,_)
         <- Action.

    +!execute(Action)
         : prohibition(Action,Condition)
           & not Condition
         <- Action.

    +!execute(Action)
         : prohibition(Action,Condition)
           & Condition
           & right(Action,Cond2)
           & Cond2
```

```
          <- Action.

     +!execute(Action)
          : prohibition(Action,Condition)
            & Condition
          <- fail.
```

Estendendo o Cenário

Outra possibilidade para incluir novas restrições no mesmo cenário seria colocar mais restrições, como um grupo de agentes em comum acordo concorda a não se agredir na área sem normas, para tanto cria uma organização e os agentes utilizam uma insígnia para se identificar:

```
NormSpace lower-right {
   environment.grid[5..9][5..9];
   roles = {agent,orgMember};
   roleAdoption{
       condition   : agent.hasBadge = true -> orgMember;
       defaultRole : agent;
   }
}

NormObj norm-obj5 {
    type       := "prohibition";
    place      := {lower-right};
    norm       := "has(agent,prohibition(attack(SELF,AG),
                          implements(AG,lower-right.orgMember)))";
    issuedBy   := SYSTEM;
    sourceOrg  := SYSTEM;
    sourceRole := SYSTEM;
    addresses  := {role(orgMember)};
    condition  := true;
}
```

Outra possibilidade seria a definição de clãs, sejam por afinidade ou laços familiares, cujos agentes têm como não agredir membros do próprio clã. No caso e laços familiares, esta informação seria melhor armazenada na forma de uma organização não-espacial, já que os laços familiares não dependeriam de localização espacial.

```
Organisation clanA{
roles = {member};
  owner = {ag1};
  Norm norm1 = {has(member,prohibition(atack(SELF,AG),
                              implements(AG,clanA.member)))};
}

 implements(ag1, clanA.member);
 implements(ag2, clanA.member);
```

6.2 Estudo de Caso 2: *Cleaning Robots*

Este estudo de caso é baseado em um cenário de (BORDINI; HÜBNER et al., 2007). Neste cenário um robô que recolhe lixo em um espaço representado por uma grade e leva até o robô incinerador que fica fixo na posição central da grade. Para tornar o cenário um pouco mais complexo foram alteradas algumas da condições do cenário original:

1. inclusão de mais agentes para recolher o lixo;
2. o cenário é dividido em duas partes, onde cada agente é responsável por um setor (leste e oeste);
3. um agente que estiver fora do seu setor não possui obrigação de recolher lixo;
4. cada robô possui um fabricante e/ou modelo que pode requerer uma rotina de manutenção diferente. O fabricante A requer que o robô faça manutenção (*selfCheck*) a cada 50 ciclos (1 ciclo é contado cada vez que esvazia o repositório de lixo), enquanto o fabricante B requer que o robô realize manutenção a cada 100 ciclos;
5. em determinados pontos do cenário, é exigido que o robô se mova em modo silencioso;
6. determinados pontos do cenário não devem ser limpos;

6.2.1 Cenário Modelado

Os principais aspectos do ambiente definido em ELMS, seriam:

```
environment.name:="CleaningRobots";
environment.grid(10,10);

Resource garbage{
        id: integer;
}

Resource incinerator{
        id: integer;
}

AgentBody agent{
    id: integer(SELF);
    manufacter: string;
    cycles: integer(0);
    loaded: boolean(false);
    vision: PERCEPTION;
    counter:PERCEPTION;
    walk: ACTION;
    silentWalk:ACTION;
    collect: ACTION;
    dump: ACTION;
    selfCheck: ACTION;
}

Perception vision{cell[+0][+0].contents;}
Perception counter{agent(SELF).cycles;}

Action walk(direction){...}
Action silentWalk(direction){...}
Action collect(condition:exists(garbage,+0,+0) & loaded == false;
          {delete(garbage); loaded=true;}

Action dump(condition: exists(incinerator,+0,+0)
             & loaded==true; loaded=false;)
Action selfCheck{cycles=0;}
```

No trecho acima, é definida a grade onde os agentes procuram pelo lixo, o recurso que representa o lixo, e agente incinerador é também representado por um recurso, já que o único objetivo deste agente seria demarcar um ponto na grade. Na seqüência é definido o agente que realiza a limpeza, definido por seus atributos relevantes ao ambiente, percepções e ações.

6.2.2 Infraestrutura Normativa

No trecho abaixo são definidos quatro espaços normativos, sendo dois para dividir a área de limpeza entre dois agentes robôs. É definido o espaço normativo onde os agentes deverão se mover em modo silencioso e o espaço normativo referente a área de pesquisa onde os agentes não poderão coletar lixo. Em seguida são definidos três objetos normativos que indicam as normas do ambiente.

```
NormSpace sectEast{
   environment.grid[0..4][0..9];
    roles = {agent, cleaner};
    roleAdoption{
       condition: agent.name = agW -> agent.role = cleaner;
       defaultRole : agent;
   }
}

NormSpace sectWest{
    environment.grid[5..9][0..9];
    roleAdoption{
       condition: agent.name = agE -> agent.role = cleaner;
       defaultRole : agent;
   }
}

NormSpace silentArea {
    environment.grid[3..6][3..6];
    roles = {agent};
    roleAdoption{
       defaultRole : agent;
   }
}

NormSpace researchArea {
    environment.grid[0..2][5..6];
    roles = {agent,researcher};
    roleAdoption{
       defaultRole : agent;
   }
}

NormObj norm-obj1 {
    type       := "obligation";
    place      := {sectEast,sectWest};
    norm       := "has(cleaner,obligation(collect(_),
                              exists(garbage,+0,+0))";
    issuedBy   := SYSTEM;
    sourceOrg  := SYSTEM;
    sourceRole := SYSTEM;
    addresses  := {role(cleaner)};
    condition  := true;
}
```

```
NormObj norm-obj2 {
    type       := "prohibition";
    place      := "silentArea"
    norm       := "has(agent,prohibition(walk(_),true))";
    issuedBy   := SYSTEM;
    sourceOrg  := SYSTEM;
    sourceRole := SYSTEM;
    addressees := {role(agent)};
    condition  := true;
}

NormObj norm-obj3 {
    type       := "prohibition";
    place      := "researchArea"
    norm       := "has(agent,prohibition(collect(_),true))";
    issuedBy   := SYSTEM;
    sourceOrg  := SYSTEM;
    sourceRole := SYSTEM;
    addressees := {role(agent),role(cleaner)};
    condition  := true;
}
```

Através da definição de novos espaços normativos a área de limpeza seria facilmente redistribuída, podendo por exemplo dividir a grade em quatro quadrantes e fazer uso do mesmo objeto normativo para definir a obrigação do agente de limpar a área. Também seriam facilmente definidas novas áreas com diferentes restrições para o trabalho dos robôs.

Abaixo a definição dos grupos relacionados aos fabricantes:

```
Organisation manufacterA{
 roles = {robot};
 owner = {none};
 Norm n ={has(robot,obligation(selfCheck, agent(SELF,cycles(50))))};
}

Organisation manufacterB{
 roles = {robot};
 owner = {none};
 Norm n ={has(robot,obligation(selfCheck, agent(SELF,cycles(100))))};
}

implements(agE,manufacterA.robot);
implements(agW,manufacterB.robot);
```

6.2.3 Raciocínio dos Agentes

O raciocínio dos agentes no cenário original desta simulação é bastante simples, de maneira geral seria da seguinte forma:

- Se o robô estiver carregado, na posição da lixeira, ele deve descarregar;
- Se o robô esteja carregado, ele deverá ir até a lixeira;
- Se o robô estiver em uma posição onde existe lixo, deve recolher lixo;
- Caso contrário ir para a próxima posição;

A seguir, mostramos apenas alguns trechos do raciocínio dos agentes, que mostram as estensões ao raciocínio dos agentes:

```
+garbage(I,position(+0,+0))
   :obligation(collect(ANY),Cond) & Cond;
   <- collect(I);

 -!execute(walk(Dir))
   <- +!execute(silentWalk(Dir));

 +agent(SELF, cycles(X))
    :obligation(selfCheck, agent(SELF,cycles(Y))) & .gte(X,Y)
    <- +!execute(selfCheck).

 +agent(SELF,position(X,Y)
   <- !moveNextPosition();
```

No primeiro plano, o agente verifica a existência de lixo, e caso ele tenha a obrigação de coletar o lixo, o agente irá coletá-lo. No plano seguinte, ao receber o evento informando que o objetivo de executar a ação de andar falhou, o agente tenta movimentar-se novamente, porém no modo silencioso. No terceiro plano, o agente verifica o número de ciclos desde sua última manutenção e verifica a norma da organização, caso seu número de ciclos seja maior ou igual ao número especificado, o agente irá se comprometer com o objetivo de executar a manutenção; note que na organização não existe a opção de condicionar a percepção da norma à uma condição, no caso de objetos normativos seria possível informar a norma apenas quando a condição de ciclos fosse satisfeita. Finalmente, no último plano, como não percebeu a existência de lixo na célula corrente o agente cria o objetivo de se movimentar para célula seguinte.

O cenário básico (sem as extensões normativas) deste estudo de caso encontra-se implementado com uso do ***Jason***, por Rafael H. Bordini e Jomi F. Hübner, está disponível em `http://jason.sourceforge.net` na seção de exemplos.

6.3 Considerações Finais

No primeiro estudo de caso, à partir da modelagem do ambiente, encontra-se quase totalmente a situação problemática, que seria a disputa por comida em um cenário limitado com comida escassa. Os elementos da situação problemática não contemplados na modelagem do ambiente seriam as normas limitam a ação dos agentes e podem ser representadas com o uso da infra-estrutura normativa proposta neste trabalho. Assim, a tarefa restante é modelar o raciocínio do agente.

Nesta fase, as ações disponíveis aos agentes e as percepções provenientes do ambiente já estão definidas, o que facilita o desenvolvimento dos planos dos agentes, permitindo a realização de refinamentos e ajustes. O cenário originalmente apresentado em (CASTELFRANCHI; CONTE; PAOLUCCI, 1998) possui um ambiente explícito e bem definido, o que o torna um cenário propício para a inserção de novas possibilidades aos agentes e seu raciocínio. Conforme apresentado, a adição de novas normas e estruturas é bastante prática e direta.

No segundo cenário, pode-se perceber que a partir de um ambiente simples é possível inserir conceitos relativos a normas e organizações, em um cenário que inicialmente não previa tais situações. A estrutura implementada permitiria facilmente a inclusão de outros

agentes para recolher o lixo nas mesmas áreas, além de poder se reparticionar os espaços para limpeza de maneira simples. Através do uso das abstrações foi possível modelar de maneira simples as normas existentes nos cenários modelados. Assim, acredita-se que desta forma foi possível verificar a utilidade do uso destas abstrações, apesar de não ser possível quantificar o ganho obtido, é possível verificar que seu uso facilitou a implementação e operacionalização de normas, agregando algumas noções básicas de organizações.

Como se tratam de cenários de simulação social com normas, uma das características a que se busca é a possibilidade de permitir que o agente tome a decisão de seguir ou não as regras. Desta forma, um agente poderia tomar suas próprias decisões no caso de conflitos entre regras de diferentes instâncias ou entre conflitos do interesse próprio e as regras sociais. É importante notar que mesmo que se permita aos agentes a possibilidade de infringir as regras, a integridade do sistema fica garantida através da definição dos avatares, que com suas definições de ações e percepções, garantem que os agentes não irão realizar ações que comprometam a integridade da simulação.

7 CONCLUSÃO

Nesta tese foi apresentada uma abordagem para modelagem de sistemas multiagentes. Esta abordagem visa modelar de maneira integrada agentes, ambiente e organizações possibilitando o uso de abstrações que facilitam a criação de simulações sociais. Este modelo, inserido no contexto do projeto MAS-SOC, que tem como maior contribuição o suporte a criação de agentes BDI e simulações envolvendo este tipo de agentes. O modelo está em aperfeiçoamento, porém já produziu alguns resultados interessantes tais como apresentados em (OKUYAMA; BORDINI; ROCHA COSTA, 2007a, 2008a,b).

Assim, a abordagem MAS-SOC é uma proposta promissora para simulação social com agentes cognitivos, enquanto que outras plataformas disponíveis são em geral para agentes reativos ou simplesmente ignoram a modelagem do ambiente onde os agentes interagem. Isto permite que no futuro possa pensar-se em questões tais como conciliar emergência e cognição, como proposto por Castelfranchi (1998; 2001).

Em trabalhos anteriores, criou-se uma linguagem para descrição explícita do ambiente, a linguagem ELMS. Esta linguagem tinha como principal característica modelar agentes e ambientes de maneira integrada. Recentemente a linguagem foi estendida, integrando-se uma infraestrutura normativa aos SMA, integrando ambientes e organizações de maneira à possibilitar que as organização pudessem influir no ambiente físico simulado. A infraestrutura normativa, composta basicamente por *objetos normativos* e *espaços normativos*, fornece meios para a distribuição de normas no ambiente, permitindo a contextualização espacial das normas. A partir destas extensões, foi criado uma nova versão para o Modelo MAS-SOC apresentada nesta tese.

7.1 Principais Contribuições

As contribuições realizadas neste trabalho estão inseridas no contexto da modelagem e desenvolvimento de simulações com sistemas multiagentes. Entre as quais pode-se citar:

Contextualização espacial de normas: a distribuição das normas sobre o ambiente, possibilita que estas sejam pré-processadas e definidas especificamente para um escopo espacial fixo. Desta forma, torna-se muito mais simples a operacionalização das normas por parte dos agentes e também a verificação da aderência às normas por outros agentes e/ou sistemas.

Percepção condicional das normas: cada objeto normativo, possui condições nas quais informa agentes específicos sobre a existência de uma norma. Assim, torna-se possível incluir uma condição temporal à divulgação da norma, possibilitando a contextualização temporal da norma. Sendo a norma divulgada apenas nos momentos em que esta é válida e deverá ser respeitada.

Raciocínio normativo: A opção pelos objetos normativos que apenas informa as normas em um contexto fixo, possibilita que os agentes realizem o raciocínio normativo, ao invés de terem as normas implementadas embutidas em seu raciocínio, respeitando a autonomia dos agentes.

Aderência a normas previamente desconhecidas: O uso dos objetos normativos e as normas contextualizadas, possibilita que os agentes cumpram normas sem que estes tenham conhecimento prévio destas normas. Desta forma, é possível criar agentes mais simples que respeitem normas que podem sofrer alterações dinamicamente. Este aspecto é importante no desenvolvimento de simulações *abertas*.

Abordagem não-intrusiva: na maioria dos sistemas que fazem uso de normas, usualmente colocam fortes restrições contra o não cumprimento das normas, inclusive a implementação direta nos agentes. A abordagem apresentada, permite que o agentes tomem conhecimento das normas e decidam sobre seguir as normas.

Normas Espacialmente Distribuídas: a abordagem de ter as normas espacialmente distribuídas, facilita a modelagem pois para o projetista é possível analisar cada espaço normativo de maneira modular, independente dos outros espaços, facilitando desta forma a modelagem de grandes sistemas.

Identificação de Intersecções nas Entidades SMA: através da modelagem da infraestrutura normativa salientou-se a existência de intersecções entre as entidades básicas de um SMA,conforme apresentado na Seção 5.2, especialmente o item C, contrastando-se com o modelo de SMA tradicional.

A modelagem de ambientes conforme o modelo MAS-SOC, facilita a implementação os outros componentes do SMA, pois transforma a situação problemática do sistema em restrições que devem ser seguidas para implementação, especialmente no raciocínio dos agentes, porém de maneira menos acentuada na modelagem das estruturas organizacionais. As abstrações apresentadas para representação de espaços normativos, representação de normas espacialmente distribuídas no ambiente podem facilitar a modelagem de sistemas e simulações sociais.

Os modelos de organizações existentes, em sua maioria, não leva em consideração o ambiente como meio de propagação de seus objetivos e conceitos. Desta forma acredita-se que esta tese apresente uma contribuição para os SMA através da modelagem de ambientes, e também para a modelagem de organizações e sistemas normativos ao integrá-los ao ambiente. Apesar de apresentar características para definição de organizações, o modelo poderia ser utilizado em conjunto com outros modelos de organizações, possibilitando ao projetista aproveitar os melhores aspectos de cada abordagem. Desta mesma maneira, este trabalho se relaciona com os sistemas normativos, apesar de conter aspectos de sistemas normativos, o uso da infraestrutura normativa apresentada não exclui a possibilidade de uso em conjunto de outras abordagens para normas.

Os estudos de caso apresentados, apesar de simples, apresentaram o uso das abstrações na modelagem de cenários de simulação. Com o uso do modelo, foi possível fornecer aos agentes informações sobre normas, as quais foram utilizadas na tomada de decisão, respeitando-se a autonomia dos agentes. O uso das abstrações contribuíram para acrescentar elementos de interesse aos cenários, apesar dos estudos de caso não permitirem quantificar contribuição do modelo, acredita-se que foi possível verificar existência de contribuição. Assim, acredita-se que este trabalho contribui para a área de pesquisa de SMA, especialmente na modelagem de ambientes.

7.2 Trabalhos Futuros

Como trabalhos futuros, existem atividades a serem realizadas para dar continuidade ao desenvolvimento do modelo. Entre os quais pode-se citar:

- Pesquisar a possibilidade de incluir estruturas que possibilitem a definição mais detalhada das interações entre agentes e ambiente (agente-agente e agente-ambiente) utilizando escopos espaciais;
- Pesquisar a viabilidade e implicações de utilizar objetos normativos para distribuir planos normativos para atingir objetivos locais e outras opções para obtenção de novos planos (e.g. troca de planos (ANCONA et al., 2004)).
- Implementar e disponibilizar o modelo para que outros pesquisadores possam testar, para verificar através do uso em simulações as necessidades de evolução do modelo.
- Analisar questões associadas ao uso do modelo. Uma destas questões é sobre o fato de ter diversas normas distribuídas em escopos espaciais diferentes podendo resultar em diferentes reputações de um único agente no ambiente, levando a uma noção de *localidade da reputação*.

Através dos estudos de caso, foi possível verificar que o uso das abstrações fornecidas facilita a inserção de normas e outras estruturas no cenário. Dando continuidade ao modelo e visando a elaboração de cenários mais sofisticados, seriam necessários a inserção de outras estruturas e abstrações, tais como as existentes em (LOPEZ; LUCK; D'INVERNO, 2007), para inserir objetivos normativos e também seriam interessantes inserir outros protocolos que descrevam tarefas, tais como os existentes em (DIGNUM; VÁZQUEZ-SALCEDA; DIGNUM, 2005).

Outro aspecto interessante é que os agentes sendo condicionados na possibilidade de perceber a existência de um objeto normativo, o raciocínio normativo requerido pelos agentes que lidam com objetos normativos deverá ser de natureza não-monotônica. Além disso, o *raciocínio normativo* relacionado com a possibilidade de criar espaços normativos no ambiente, cada um com seu próprio propósito organizacional e estrutura normativa, nos leva a várias possibilidades que deverão ser analisadas no futuro.

Outras questões que não foram abordadas neste trabalho, também necessitariam de maior estudo posterior, tal como o cruzamento de informações das diferentes entidades multiagentes envolvidas para auxiliar no desenvolvimento da aplicação. Uma das possibilidades seria a verificação dos tipos de *papéis relativos* relacionados a um *espaço normativo* e os papéis da organização presente no local, pois a verificação da consistência destes papéis atualmente fica a cargo do projetista da simulação. Outra verificação possível seria a consistência de *missões* com os espaços relacionados e os recursos necessários. Além disso, o aspecto da interação dos agentes, tais como protocolos e linguagens para interação, não é tratado explicitamente pelo modelo, sendo possível defini-lo implicitamente nos agentes ou através de normas, porém a definição explícita poderia trazer benefícios, o que poderá ser estudado em trabalhos futuros.

REFERÊNCIAS

ALDEWERELD, H. et al.. Designing Normative Behaviour Via Landmarks. In: INTERNATIONAL WORKSHOPS ON AGENTS, NORMS AND INSTITUTIONS FOR REGULATED MULTI-AGENT SYSTEMS, ANIREM; OOOP, 2005. **Coordination, Organizations, Institutions, and Norms in Multi-Agent Systems**: revised selected papers. Berlin: Springer-Verlag, 2006. p. 157–169. Lecture Notes in Computer Science, v.3913).

ALECHINA, N.; JAGO, M.; LOGAN, B. Resource-bounded belief revision and contraction. In: INTERNATIONAL WORKSHOP ON DECLARATIVE AGENT LANGUAGES AND TECHNOLOGIES, DALT, 3., 2005, Utrecht, the Netherlands. **Proceedings...** [S.l.: s.n.], 2005.

ÁLVARES, L. O. C.; SICHMAN, J. S. Introdução aos Sistemas Multiagentes. In: JORNADA DE ATUALIZAÇÃO EM INFORMÁTICA, JAI, 16, 1997 Brasília. **Anais...** , 1997. BRASÍLIA : UNB, 1997.

ANCONA, D.; MASCARDI, V. Coo-BDI: extending the BDI model with cooperativity. In: INTERNATIONAL WORKSHOP ON DECLARATIVE AGENT LANGUAGES AND TECHNOLOGIES, DALT, 1., 2003, Melbourne, Australia. **Declarative Agent Languages and Technologies**: Revised Selected and invided papers. Berlin: Springer-Verlag, 2004. p.109–134. (Lecture Notes in Computer Science, v.2990).

ANCONA, D.; MASCARDI, V.; HÜBNER, J. F.; BORDINI, R. H. Coo-AgentSpeak: cooperation in agentspeak through plan exchange. In: INTERNATIONAL JOINT CONFERENCE ON AUTONOMOUS AGENTS AND MULTIAGENT SYSTEMS, AAMAS, 3., 2004. **Proceedings...** [S.l.: s.n.]. IEEE Computer Society, 2004. p.696–705.

BOELLA, G.; TORRE, L. van der; VERHAGEN, H. Introduction to Normative Multiagent Systems. In: NORMATIVE MULTI-AGENT SYSTEMS, 2007. **Anais...** Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI): Schloss Dagstuhl: Germany, 2007. n.07122. (Dagstuhl Seminar Proceedings). <http://drops.dagstuhl.de/opus/volltexte/2007/918> [date of citation: 2007-01-01].

BORDINI, R. H.; BAZZAN, A. L. C.; JANNONE, R. O.; BASSO, D. M.; VICARI, R. M.; LESSER, V. R. AgentSpeak(XL): efficient intention selection in BDI agents via decision-theoretic task scheduling. In: INTERNATIONAL JOINT CONFERENCE ON AUTONOMOUS AGENTS AND MULTI-AGENT SYSTEMS, AAMAS, 1., 2002, **Bringing People and Agents Together**: Proceedings. New York: ACM, 2002. v.3, p.1294–1302.

BORDINI, R. H.; COSTA, A. C. d. R.; HüBNER, J. F.; MOREIRA, A. F.; OKUYAMA, F. Y.; VIEIRA, R. MAS-SOC: a social simulation platform based on agent-oriented programming. **Journal of Artificial Societies and Social Simulation**, [S.l.], v.8, n.3, 2005.

BORDINI, R. H.; HÜBNER, J. F. et al. ***Jason***: a Java-based agentspeak interpreter used with saci for multi-agent distribution over the net. Manual, Release version 0.9.5.ed. [S.l.: s.n.], 2007. Disponível em: <http://jason.sourceforge.net/>. Acesso em 01/11/2008.

BORDINI, R. H.; HÜBNER, J. F.; VIEIRA, R. ***Jason*** and the Golden Fleece of Agent-Oriented Programming. In: BORDINI, R. H.; DASTANI, M.; DIX, J.; EL FALLAH SEGHROUCHNI, A. (Ed.). **Multi-Agent Programming**: languages, platforms and applications. [S.l.]: Springer-Verlag, 2005. p.3–37.

BORDINI, R. H.; HÜBNER, J. F.; WOOLDRIDGE, M. **Programming Multi-Agent Systems in AgentSpeak Using *Jason***. [S.l.]: John Wiley & Sons, 2007. (Wiley Series in Agent Technology).

BORDINI, R. H.; MOREIRA, A. F. Proving BDI Properties of Agent-Oriented Programming Languages: the asymmetry thesis principles in AgentSpeak(L). **Annals of Mathematics and Artificial Intelligence**, [S.l.], v.42, n.1–3, p.197–226, Sept. 2004. Special Issue on Computational Logic in Multi-Agent Systems.

BORDINI, R. H. et al.. The MAS-SOC Approach to Multi-agent Based Simulation. In: INTERNATIONAL WORKSHOP ON REGULATED AGENT-BASED SOCIAL SYSTEMS, RASTA, 1., 2002. **Regulated Agent-Based Social Systems**: revised selected and invited papers. Berlin: Springer-Verlag, 2004. p.70–91. (Lecture Notes in Computer Science, v.2934).

BORDINI, R. H.; VIEIRA, R.; MOREIRA, Á. F. Fundamentos de Sistemas Multiagentes. In: Jornada de Atualização em Informática, JAI, 20., 2001, Fortaleza. **Anais...** Fortaleza: SBC, 2001. p.3–41.

CARLEY, K. M.; GASSER, L. Computer Oranization Theory. In: WEISS, G. (Ed.). **Multiagent Systems—A Modern Approach to Distributed Artificial Intelligence**. Cambridge, MA: MIT Press, 1999. p.299–330.

CASTELFRANCHI, C. Simulating with Cognitive Agents: the importance of cognitive emergence. In: INTERNATIONAL WORKSHOP ON MUTI-AGENT SYSTEMS AND AGENT-BASED SYSTEMS, MASBS, 1998, Paris. **Proceedings...** Springer-Verlag, 1998. p.26–44. (Lecture Notes in Artificial Intelligence, v.1534).

CASTELFRANCHI, C. The Theory of Social Functions: challenges for computational social science and multi-agent learning. **Cognitive Systems Research**, [S.l.], v.2, n.1, p.5–38, Apr. 2001.

CASTELFRANCHI, C.; CONTE, R.; PAOLUCCI, M. Normative reputation and the costs of compliance. **Journal of Artificial Societies and Social Simulation**, Guildford, UK, v.1, n.3, 1998.

CASTELFRANCHI, C. et al. . Deliberative Normative Agents: principles and architecture. In: INTERNATIONAL WORKSHOP ON INTELLIGENT AGENTS VI, AGENT THEORIES, ARCHITECTURES, AND LANGUAGES, ATAL, 6., 1999, Florida, USA. **Proceedings...** Berlin: Springer-Verlag, 1999. p.364–378. (Lecture Notes In Computer Science, v. 1757).

CONTE, R.; CASTELFRANCHI, C. **Cognitive and Social Action**. London: UCL Press, 1995.

CONTE, R.; CASTELFRANCHI, C.; DIGNUM, F. Autonomous Norm Acceptance. In: INTERNATIONAL WORKSHOP ON INTELLIGENT AGENTS V, AGENT THEORIES, ARCHITECTURES, AND LANGUAGES, ATAL, 5., 1998, Paris. **Proceedings...** Berlin: Springer, 1999. p.99–112. (Lecture Notes in Computer Science, v.1555).

CONTE, R.; FALCONE, R.; SARTOR, G. Introduction: agents and norms: how to fill the gap? **Artificial Intelligence and Law**, [S.l.], v.7, n.1, p.1–15, 1999.

CONTE, R.; GILBERT, N.; SICHMAN, J. S. MAS and Social Simulation: a suitable sommitment. In: INTERNATIONAL WORKSHOP ON MULTI-AGENT SYSTEMS AND AGENT-BASED SIMULATION, FIRST INTERNATIONAL WORKSHOP, MABS, 1., Paris. **Proceedings...** Berlin: Springer, 1998. p.1–9. (Lecture Notes in Computer Science, v.1534).

COSTA, A. C. R.; DIMURO, G. P. Semantical Concepts for a Formal Structural Dynamics of Situated Multiagent Systems. In: COORDINATION, ORGANIZATIONS, INSTITUTIONS, AND NORMS IN AGENT SYSTEMS III, 2008. Berlin: Springer, 2008. p.139–154. (Lecture Notes in Artificial Intelligence, v.4870).

COUTINHO, L. R.; SICHMAN, J. S.; BOISSIER, O. Organizational Modeling Dimensions in Multiagent Systems. In: IBERAGENTS, 2006, Ribeirao Preto, SP, Brazil. **Anais...** [S.l.: s.n.], 2006.

DEMAZEAU, Y. From Cognitive Interactions to Collective Behaviour in Agent-Based Systems. In: EUROPEAN CONFERENCE ON COGNITIVE SCIENCE, 1995, Saint-Malo. **Proceedings...** [S.l.: s.n.], 1995.

DEMAZEAU, Y.; COSTA, A. C. R. Populations and organizations in open multi-agent systems. In: NATIONAL SYMPOSIUM ON PARALLEL AND DISTRIBUTED AI, PDAI, 1., 1996, Hyderabad, India. **Proceedings...** [S.l.: s.n.], 1996.

DIGNUM, F. Abstract Norms and Electronic Institutions. In: INTERNATIONAL WORKSHOP ON REGULATED AGENT-BASED SOCIAL SYSTEMS, RASTA, 1., 2002, Bologna, Italy. **Proceedings...** [S.l.: s.n.], 2002.

DIGNUM, V.; VÁZQUEZ-SALCEDA, J.; DIGNUM, F. OMNI: introducing social structure, norms and ontologies into agent organizations. In: INTERNATIONAL WORKSHOP ON PROGRAMMING MULTI-AGENT SYSTEMS, PROMAS, 2., 2004, New York, NY, USA. **Selected Revised and Invited Papers** Berlin: Springer, 2005. p.181–198. (Lecture Notes in Computer Science, v.3346).

d'INVERNO, M.; LUCK, M. Engineering AgentSpeak(L): a formal computational model. **Journal of Logic and Computation**, [S.l.], v.8, n.3, p.1–27, 1998.

FERBER, J. **Multi-Agent Systems**: an introduction to distributed artificial intelligence. Boston, MA, USA: Addison-Wesley Longman Publishing, 1999.

FERBER, J.; MICHEL, F.; BÁEZ-BARRANCO, J.-A. AGRE: integrating environments with organizations. In: INTERNATIONAL WORKSHOP ON ENVIRONMENTS FOR MULTI-AGENT SYSTEMS, E4MAS, 1., 2004, New York, NY, USA. **Revised Selected Papers**. Berlin: Springer, 2005. p.48–56. (Lecture Notes in Computer Science, v.3374).

FORNARA, N.; COLOMBETTI, M. Specifying and Enforcing Norms in Artificial Institutions. In: NORMATIVE MULTI-AGENT SYSTEMS, 2007. **Anais...** Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI): Schloss Dagstuhl: Germany, 2007. n.07122. (Dagstuhl Seminar Proceedings). <http://drops.dagstuhl.de/opus/volltexte/2007/909> [date of citation: 2007-01-01].

FRANCO, M. H. I.; COSTA, A. C. R. Towards a Protocol for Negotiations about Exchange Values Involved in Multiagent Interactions. In: INTERNATIONAL WORKSHOP ON COMPUTATIONAL MODELS OF NATURAL ARGUMENT, CMNA, 7., 2007, Hyderabad, India. **Proceedings...**. Hyderabad: IJCAI, 2007. p. 01–05.

GARCIA-CAMINO, A.; NORIEGA, P.; RODRÍGUEZ-AGUILAR, J. A. Implementing norms in electronic institutions. In: INTERNATIONAL JOINT CONFERENCE ON AUTONOMOUS AGENTS AND MULTIAGENT SYSTEMS, AAMAS, 4., 2005, Utrecht, The Netherlands. **Proceedings...** New York:ACM, 2005. p.667–673.

GARCÍA-CAMINO, A.; RODRÍGUEZ-AGUILAR, J. A.; SIERRA, C.; VASCONCELOS, W. W. A Distributed Architecture for Norm-Aware Agent Societies. In: INTERNATIONAL WORKSHOP ON DECLARATIVE AGENT LANGUAGES AND TECHNOLOGIES III, DALT, 3., 2005, Utrecht, The Netherlands. **Selected and Revised Papers**. Berlin: Springer, 2006. p.89–105. (Lecture Notes in Computer Science, v.3904).

GILBERT, N.; TROITZSCH, K. G. **Simulation for the Social Scientist**. [S.l.]: Open University Press, 1999.

HÜBNER, J. F.; SICHMAN, J. S. ao. SACI: uma ferramenta para implementação e monitoração da comunicação entre agentes. In: IBERO-AMERICAN CONFERENCE ON AI, 7., BRAZILIAN SYMPOSIUM ON AI, 15., 2000, Atibaia. **Open Discussion Track Proceedings**. São Carlos: ICMC/USP, 2000. p.47–56.

HÜBNER, J. F.; SICHMAN, J. S.; BOISSIER, O. $\mathcal{M}$OISE$^+$: towards a structural, functional, and deontic model for MAS organization. In: INTERNATIONAL JOINT CONFERENCE ON AUTONOMOUS AGENTS AND MULTI-AGENT SYSTEMS, AAMAS, 1., 2002, Bologna, Italy. **Proceedings...** [S.l.: s.n.], 2002.

HUHNS, M.; LARRY, M.; STEPHENS, L. Multiagent Systems and Societies of Agents. In: Weiss G. (ed.). **Multiagent Systems**: a modern approach to distributed artificial intelligence. Cambridge, MA: MIT Press, 1999. p.79–120.

JENNINGS, N. R.; SYCARA, K. P.; WOOLDRIDGE, M. J. A Roadmap of Agent Research and Development. **Autonomous Agents and Multi-Agent Systems**, [S.l.], v.1, n.1, p.7–38, 1998.

KAGAL, L.; FININ, T. W.; JOSHI, A. A Policy Language for a Pervasive Computing Environment. In: IEEE INTERNATIONAL WORKSHOP ON POLICIES FOR DISTRIBUTED SYSTEMS AND NETWORKS, POLICY, 4., 2003, Lake Como, Italy. **Proceedings...** [S.l.]: IEEE Computer Society, 2003. p.63–74.

KRAFTA, R.; OLIVEIRA, D. d.; BORDINI, R. H. The city as object of human agency. In: INTERNATIONAL SPACE SYNTAX SYMPOSIUM, 4., 2002, London. **Proceedings...** [S.l.: s.n.], 2002.

LÓPEZ, F. L. y; LUCK, M. A Model of Normative Multi-agent Systems and Dynamic Relationships. In: INTERNATIONAL WORKSHOP ON REGULATED AGENT-BASED SOCIAL SYSTEMS, RASTA, 1., 2002, Bologna, Italy.**Revised Selected and Invited Papers** Berlin: Springer, 2004. p.259–280. (Lecture Notes in Computer Science, v.2934).

LOPEZ, F. L. y; LUCK, M.; D'INVERNO, M. A Normative Framework for Agent-Based Systems. In: NORMATIVE MULTI-AGENT SYSTEMS, 2007. **Anais...** Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI): Schloss Dagstuhl: Germany, 2007. n.07122. (Dagstuhl Seminar Proceedings). <http://drops.dagstuhl.de/opus/volltexte/2007/933> [date of citation: 2007-01-01].

MACHADO, R.; BORDINI, R. H. Running AgentSpeak(L) Agents on SIM_AGENT. In: INTERNATIONAL WORKSHOP ON AGENT THEORIES, ARCHITECTURES, AND LANGUAGES, ATAL, 8., 2001, Seattle, WA. **Intelligent Agents VIII**: proceedings. Berlin: Springer-Verlag, 2002. p.158–174. (Lecture Notes in Artificial Intelligence, v.2333).

MOREIRA, A. F.; BORDINI, R. H. An Operational Semantics for a BDI Agent-Oriented Programming Language. In: INTERNATIONAL WORKSHOP ON LOGICS FOR AGENT-BASED SYSTEMS, LABS, 2002, Toulouse, France. **Proceedings...** [S.l.: s.n.], 2002. p.45–59.

MOREIRA, A. F.; VIEIRA, R.; BORDINI, R. H. Extending the Operational Semantics of a BDI Agent-Oriented Programming Language for Introducing Speech-Act Based Communication. In: INTERNATIONAL WORKSHOP ON DECLARATIVE AGENT LANGUAGES AND TECHNOLOGIES, DALT, 1., 2003, Melbourne, Australia. **Revised Selected and Invited Papers** Berlin: Springer-Verlag, 2004. p.135–154. (Lecture Notes in Artificial Intelligence, v.2990).

MOREIRA, A. F.; VIEIRA, R.; BORDINI, R. H.; HÜBNER, J. F. Agent-Oriented Programming with Underlying Ontological Reasoning. In: INTERNATIONAL WORKSHOP ON DECLARATIVE AGENT LANGUAGES AND TECHNOLOGIES III, DALT, 3., 2005, Utrecht, The Netherlands. **Selected and Revised Papers** Berlin: Springer, 2006. p.155–170. (Lecture Notes in Computer Science, v.3904).

OKUYAMA, F. Y. **Descrição e Geração de Ambientes para Simulações com Sistemas Multiagentes**. 2003. Dissertação (Mestrado em Ciência da Computação) — Instituto de Informática, UFRGS, Porto Alegre, RS.

OKUYAMA, F. Y.; BORDINI, R. H.; ROCHA COSTA, A. C. da. ELMS: an environment description language for multi-agent simulation. In: INTERNATIONAL WORKSHOP ON ENVIRONMENTS FOR MULTI-AGENT SYSTEMS, E4MAS, 1., 2004, New York, NY, USA. **Revised Selected Papers** Berlin: Springer, 2005. p.91–108. (Lecture Notes in Computer Science, v.3374).

OKUYAMA, F. Y.; BORDINI, R. H.; ROCHA COSTA, A. C. da. Spatially Distributed Normative Objects. In: INTERNATIONAL WORKSHOP ON COORDINATION, ORGANIZATION, INSTITUTIONS AND NORMS IN AGENT SYSTEMS II, COIN, 2., 2006, Riva del Garda, Italy. **Proceedings...** Berlin, Springer, 2007. (Lecture Notes in Computer Science, v.4386).

OKUYAMA, F. Y.; BORDINI, R. H.; ROCHA COSTA, A. C. da. Spatially Distributed Normative Infrastructure. In: INTERNATIONAL WORKSHOP ON ENVIRONMENTS FOR MULTI-AGENT SYSTEMS, E4MAS, 3., 2006, Hakodate, Japan. **Selected Revised and Invited Papers** Berlin: Springer, 2007. p.203–220. (Lecture Notes in Computer Science, v.4389).

OKUYAMA, F. Y.; BORDINI, R. H.; ROCHA COSTA, A. C. da. Augmenting Multi-Agent Environment Descriptions with a Normative Infrastructure. In: BRAZILIAN MEETING ON ARTIFICIAL INTELLIGENCE, ENIA, 6., 2007, Rio de Janeiro, RJ, Brazil. **Proceedings...** [S.l.: s.n.], 2007.

OKUYAMA, F. Y.; BORDINI, R. H.; ROCHA COSTA, A. C. da. A distributed normative infrastructure for situated multi-agent organisations. In: INTERNATIONAL JOINT CONFERENCE ON AUTONOMOUS AGENTS AND MULTIAGENT SYSTEMS, AAMAS, 7., 2008, Estoril, Portugal. **Proceedings...** IFAAMAS, 2008. p.1501–1504. v.3.

OKUYAMA, F. Y.; BORDINI, R. H.; ROCHA COSTA, A. C. da. A distributed normative infrastructure for situated multi-agent organisations. In: INTERNATIONAL WORKSHOP ON DECLARATIVE AGENT LANGUAGES AND TECHNOLOGIES, DALT, 6., 2008, Estoril, Portugal. **Revised Selected and Invited Papers**. Berlin: Springer, 2008. p.29–46. (Lecture Notes in Computer Science, v.5397).

OKUYAMA, F. Y.; VIEIRA, R.; BORDINI, R. H.; ROCHA COSTA, A. C. da. An Ontology for Defining Environments within Multi-Agent Simulations. In: WORKSHOP ON ONTOLOGIES AND METAMODELING IN SOFTWARE AND DATA ENGINEERING, WOMSDE, 2006, Florianópolis. **Anais...** [S.l.]: SBC, 2006. p.32–41.

OMICINI, A. et al.. Coordination artifacts: environment-based coordination for intelligent agents. In: INTERNATIONAL JOINT CONFERENCE ON AUTONOMOUS AGENTS AND MULTIAGENT SYSTEMS, AAMAS, 3., 2004, New York, NY, USA. **Proceedings...** [S.l.: s.n.], 2004.

PACHECO, O.; OKUYAMA, F.; DIAS, A. Simulación del Proceso de Compra de Artículos en un Mercado Virtual con Agentes BDI. In: CONFERENCIA LATINOAMERICANA DE INFORMÁTICA, CLEI, 30., 2004, Arequipa, Peru. **Articulos**. Peru: Sociedad Peruana de Computación, 2004. p.214–223.

PRIETULA, M.; CARLEY, K.; GASSER, L. (Ed.). **Simulating Organizations**: computational models of institutions and groups. Menlo Park, CA: AAAI Press / MIT Press, 1998.

RAO, A. S. AgentSpeak(L): BDI agents speak out in a logical computable language. In: INTERNATIONAL WORKSHOP ON MODELLING AUTONOMOUS AGENTS IN A MULTI-AGENT WORLD, MAAMAW, 7., 1996, Eindhoven, The Netherlands. **Proceedings...** Berlin: Springer-Verlag, 1996. p.42–55. (Lecture Notes in Artificial Intelligence, v.1038).

RAO, A. S.; GEORGEFF, M. P. Decision Procedures for BDI Logics. **Journal of Logic and Computation**, [S.l.], v.8, n.3, p.293–343, 1998.

RODRIGUES, M. R. **Um Sistema de Valores de Troca para Suporte as Interacoes em Sociedades Artificiais**. 2003. Dissertação (Mestrado em Ciência da Computação)— Instituto de Informática, UFRGS, Porto Alegre, RS.

RODRIGUES, M. R.; ROCHA COSTA, A. C. da; BORDINI, R. H. A System of Exchange Values to Support Social Interactions in Artificial Societies. In: AUTONOMOUS AGENTS AND MULTI AGENT SYSTEMS, AAMAS, 2., 2003, Melbourne, Australia. **Proceedings...** New York: ACM Press, 2003. p.81–88.

RUBINO, R.; OMICINI, A.; DENTI, E. Computational Institutions for Modelling Norm-Regulated MAS: an approach based on coordination artifacts. In: INTERNATIONAL WORKSHOP ON "AGENTS, NORMS AND INSTITUTIONS FOR REGULATED MULTI-AGENT SYSTEMS", ANI@REM, 1., 2005, Utrecht, The Netherlands. **Proceedings...** [S.l.: s.n.], 2005.

RUSSELL, S.; NORVIG, P. **Artificial Intelligence**: a modern approach. 2nd ed. [S.l.]: Prentice-Hall, Englewood Cliffs, NJ, 2003.

SICHMAN, J. S. ao; DEMAZEAU, Y.; BOISSIER, O. When can knowledge-based systems be called agents. In: BRAZILIAN SYMPOSIUM ON ARTIFICIAL INTELLIGENCE, SBIA, 9., 1992, Rio de Janeiro, Brazil. **Proceedings...** [S.l.]:SBC, 1992. p.172–185.

SIERRA, C. et al.. Engineering multi-agent systems as electronic institutions. **UPGRADE The European Journal for the Informatics Professional**, [S.l.], v.5, n.4, p.33–39, 2004.

VÁZQUEZ-SALCEDA, J.; DIGNUM, V.; DIGNUM, F. Organizing Multiagent Systems. **Autonomous Agents and Multi-Agent Systems**, [S.l.], v.11, n.3, p.307–360, 2005.

VIEIRA, R.; MOREIRA, Á. F.; WOOLDRIDGE, M.; BORDINI, R. H. On the Formal Semantics of Speech-Act Based Communication in an Agent-Oriented Programming Language. **Journal of Artificial Intelligence Research (JAIR)**, Danvers, MA, v.29, p.221–267, 2007.

VIROLI, M.; OMICINI, A.; RICCI, A. Engineering MAS Environment with Artifacts. In: INTERNATIONAL WORKSHOP ON ENVIRONMENTS FOR MULTI-AGENT SYSTEMS, E4MAS, 2., 2005, Utrecht, The Netherlands. **Proceedings...** [S.l.: s.n.], 2005. p.62–77.

WEYNS, D. et al.. Environment for Multiagent Systems: state-of-art and research challenges. In: INTERNATIONAL WORKSHOP ON ENVIRONMENTS FOR MULTIAGENT SYSTEMS, E4MAS, 1., 2005, New York, NY, USA. **Proceedings...** Berlin: Springer-Verlag, 2005. (Lecture Notes In Artificial Intelligence, v.3374).

WEYNS, D.; VIZZARI, G.; HOLVOET, T. Environments for Situated Multi-agent Systems: beyond infrastructure. In: INTERNATIONAL WORKSHOP ON ENVIRONMENTS FOR MULTI-AGENT SYSTEMS II, E4MAS, 2., 2005, Utrecht, The Netherlands. **Selected Revised and Invited Papers**. Berlin: Springer, 2006. p.1–17. (Lecture Notes in Computer Science, v.3830).

WOOLDRIDGE, M. Intelligent Agents. In: WEISS, G. (Ed.). **Multiagent Systems—A Modern Approach to Distributed Artificial Intelligence**. Cambridge, MA: MIT Press, 1999. p.27–77.

WOOLDRIDGE, M. **An Introduction to MultiAgent Systems**. [S.l.]: John Wiley & Sons, 2002.

WOOLDRIDGE, M.; JENNINGS, N. R.; KINNY, D. The Gaia Methodology for Agent-Oriented Analysis and Design. **Autonomous Agents and Multi-Agent Systems**, [S.l.], v.3, n.3, p.285–312, 2000.

APÊNDICE A CONSTRUÇÕES ELMS

Esta seção apresenta na forma de um pseudo-código as principais construções utilizadas na definição de ambientes ELMS. A linguagem ELMS foi originalmente definida em XML, mas para facilitar a compreensão, é utilizado este pseudo-código.

A.1 Representação Espacial

A.1.1 Opções da Grade

A existência de uma grade, tal como sua definição, é opcional. A grade pode ser bidimensional ou tridimensional, sendo os parâmetros as dimensões dos eixos X, Y e Z. Ainda na definição da grade do ambiente, uma lista de atributos pode ser definida: os atributos definidos nesta seção serão replicados para cada célula da grade. A declaração de um atributo possui um nome, o tipo de propriedade (integer, float, string ou boolean) e um valor inicial opcional. Nas expressões de inicialização (ver Seção A.11) dos atributos de cada célula podem ser utilizadas as palavras reservadas *X*, *Y* e *Z* para referenciar a posição da célula.

O exemplo abaixo mostra a definição de uma grade onde o eixo X tem dimensão 10 (pode assumir os valores de 0 a 9), e os eixos Y e Z dimensões 7 e 8, respectivamente. A dimensão do eixo Z é opcional. Cada uma das células da grade definida abaixo terá como atributos uma variável inteira e outra booleana. A variável inteira tem como nome `indiceCor` e valor inicial igual a zero. A variável booleana tem como nome `ocupado` e tem `falso` como valor inicial. Cada célula possui duas reações possíveis, `reacao1` e `r2`, a serem definidas posteriormente (ver Seção A.6).

```
Grid{
    dimensions (X,Y,Z);
    integer indiceCor = 0;
    boolean ocupado = false;
    Reactions_set{reacao1, r2};
}
```

A.1.2 Opções do Grafo

A definição do grafo e sua existência é opcional. Na definição do grafo, é definido primeiramente a lista de atributos dos nodos, sendo que todos os nodos terão os mesmos atributos, possivelmente com valores distintos. A declaração de um atributo possui um nome, o tipo de propriedade (integer, float, string ou boolean) e um valor inicial opcional. Assim como as células da grade, cada nodo pode ter um conjunto de reações que deve ser listado nesta parte da definição.

Após a definição dos atributos, cada nodo deve ser definido, definido-se seu nome e os valores iniciais de seus atributos, caso seja necessário. Em seguida, pode-se definir as arestas, definido-se um rótulo, nodo de origem, nodo de destino e um valor associado à aresta, no formato: `link(<id>,<from>,<to>,<value>)`. O valor associado à aresta pode ser tanto um peso ou um custo, de acordo com a uso do projetista.

```
Graph{
    integer indiceCor = 0;
    boolean ocupado = false;
    Reactions_set = {reacao1, r2}
    node_set = {a,b};
    link(ida,a,b,7);
    link(volta,b,a,8);
    }
}
```

O exemplo acima, define uma grafo onde cada nodo possui o atributo numérico `indiceCor` e outro booleano chamado `ocupado`. Os nodos são capazes de realizar as reações `reacao1` e `r2`. São definidos 2 nodos `a` e `b`. Neste grafo, são definidos 2 arestas, uma chamada `ida` e outra `volta`, que apesar de ligares os mesmos nodos, possuem direções diferentes e valores diferentes.

A.2 Recursos

Na seção de definição de recursos, são definidas as classes de recursos ou objetos, que podem ter diversas instâncias alocadas na simulação. Uma definição de classe de recurso possui um nome, uma lista de atributos e uma lista de reações. Os atributos são definidos da mesma maneira que os atributos das células (definindo nome, tipo e valor inicial). As reações que uma classe de recursos pode ter é dada por uma lista de nomes que identificam as reações (ver seção A.6). Note que uma mesma reação pode aparecer em diferentes classes de recursos.

O exemplo abaixo mostra a definição de um recurso que teria como nome `recurso` e seria capaz de executar as reações `reacao1` e `reacao2`, que são executadas quando suas precondições forem satisfeitas. Nas expressões de inicialização (ver seção A.11) dos atributos de cada recurso, podem ser utilizadas as palavras reservadas `SELF` e `SELFCLASS`, que representam o índice[1] de uma instância específica e o nome da classe de recurso, respectivamente. O recurso do exemplo abaixo tem como atributos duas propriedades do tipo string e uma propriedade inteira. A primeira propriedade *string* tem nome `nome` e valor inicial `agua`, a segunda *string* tem nome `tipo` e valor inicial igual a `mineral`, a propriedade inteira tem nome `qtde` e valor inicial igual a 10.

```
resource recurso{
    string nome = "agua";
    string tipo = "mineral";
    integer qtde = 10;
    Reactions_set = {reacao1, reacao2}
}
```

[1]Quando uma instância de um recurso é criada ou um agente entra na sociedade, um índice é atribuído a ele. Por exemplo, a décima instância de uma classe de recurso a ser criada ter índice 9 (a primeira tem índice 0).

A.3 Agentes

Nesta parte da definição de ambiente, são definidas as classes de agentes que podem participar das simulações em que este ambiente seja usado. Uma especificação de uma classe de agente contem seu nome, uma lista de atributos[2], uma lista de ações e uma lista de percepções permitidas para este agente. É necessário especificar uma lista de nomes ações que os agentes desta classe têm permissão para executar no ambiente. O conjunto de percepções é uma lista de nomes de percepções que estão disponíveis para uma classe de agente (o tipo de informação que o ambiente irá enviar para estes agentes a cada ciclo de percepção/ação na simulação). Assim como as reações dos recursos, os mesmos nomes de percepções e ações podem aparecer em várias definições, de forma que estes podem ser reaproveitados em diferentes classe de agentes. Da mesma forma que com os recursos, nas expressões de inicialização (ver seção A.11) dos atributos de cada recurso podem ser utilizadas as palavras reservadas `SELF` e `SELFCLASS` que representam respectivamente o índice de uma instância específica e o nome da classe do agente.

O exemplo abaixo apresenta uma definição de uma classe de agentes de nome `agente` que é capaz de realizar as ações `acao1` e `andar`, de acordo com suas precondições. Este agente também é capaz de realizar as percepções `visao` e `tato`, também dependentes de suas precondições. Essa classe de agente tem como atributo uma variável inteira de nome `id` que será inicializada com o número da instância da classe de agente. Outro atributo desta classe é o valor booleano de nome `acordado` que tem verdadeiro como valor inicial. As percepções e ações serão detalhadas nas seções que seguem.

```
Agent agente{
    integer id = SELF;
    boolean acordado = true;
    Actions_set = {acao1, andar};
    Perceptions_set = (visao, tato);
}
```

A.4 Percepções

Nesta seção são especificadas as percepções referidas nas definições de agentes. Uma definição de percepção é formada por um nome, uma lista opcional de precondições, e uma lista de nomes de propriedades. Estas propriedades podem ser quaisquer dos atributos associados com definições de recursos, agentes, células da grade ou valores de controle da simulação. Se todas as precondições são satisfeitas, então os valores das propriedades listadas serão enviadas para o agente como o resultados de sua percepção do ambiente. Note que a percepção pode ser baseada na posição espacial do agente na grade, mas isso não é obrigatório. Existem várias possibilidades que de definições que podem ser utilizadas de acordo com as necessidades do projetista do ambiente.

No exemplo que segue, está definida uma percepção de nome `visao`, cuja precondição é que o atributo `acordado` tenha valor igual a `TRUE`. O conteúdo de todas as células é enviado para cada agente que realiza este tipo de percepção e tem a precondição satisfeita. Além das palavras reservadas `SELF` e `SELFCLASS`, podem ser utilizadas as palavras reservadas `ALL` e `CONTENTS`. A palavra reservada `ALL` pode referenciar todas os índices de instâncias de uma classe ou todas as posições de um eixo da grade. A palavra reservada `CONTENTS` referencia o conteúdo de uma célula.

[2] Propriedades dos agentes que são perceptíveis ou características dos agentes enquanto parte do ambiente.

```
Perception visao{
    Precondition :(SELFCLASS[SELF].acordado == true);
    cell[ALL,ALL].CONTENTS;
}
```

Para definir percepções com variáveis de controle do ambiente, deve-se inserir da definição da percepção uma chave para atributo de elemento, colocando no campo para nome da classe a palavra reservada ENV e em atributo o nome da variável de controle, conforme o exemplo que segue:

```
ENV.STEP;
```

No exemplo acima, será enviado ao agente o conteúdo da variável de controle de ambiente STEP que indica o passo atual da simulação.

A.5 Ações

Nesta parte da definição do ambiente são definidas as ações são referenciadas nas definições de agentes. Uma definição de ação tem um nome, uma lista opcional de parâmetros, uma lista opcional de precondições, e uma seqüência de comandos que determinam quais mudanças no ambiente a ação causa. A lista de parâmetros informa quais parâmetros serão recebidos do agente para a execução deste tipo de ação. A seqüência de comandos pode ser formada por atribuições de valores para atributos; instanciação ou remoção de recursos; além de alocação ou reposicionamento de instâncias de agentes ou recursos na grade. Os comandos estão detalhados na seção A.12 Se as precondições forem totalmente satisfeitas, então todos os comandos da seqüência de comandos serão executados, na ordem em que foram definidos, ocasionando mudanças no ambiente. É importante enfatizar que as ações devem ser atômicas[3], apesar disto não ser obrigatório; porém, ao projetar o ambiente desta forma, a simulação se torna mais consistente. Os parâmetros são acessados pelos nomes e devem ser enviados pelo agente, na ordem em que foram definidos.

O exemplo abaixo define uma ação chamada moverDireita. Esta ação recebe como parâmetro uma valor inteiro que é referenciado pelo rótulo CASAS. Esta ação será executada, caso o parâmetro CASAS tenha um valor menor do que três, condição que está explicitada na seção PRECONDITION do código abaixo. A ação irá mover o agente que deseja realizar a ação para a célula à direita da célula atual do agente, dependendo do parâmetro que representa a posição relativa à direita para a qual o agente irá se mover.

```
Action moverDireita{
    Parameters{
        integer casas;
    }
    Precondition: (casas > 3);

    move(SELFCLASS[SELF],cell[+0,+0],cell[CASAS,+0]);
}
```

A.6 Reações

Esta seção define as possíveis reações dos recursos no ambiente. Para cada reação é definido um nome, uma lista de precondições, e uma seqüência de comandos. Os coman-

[3]Como o raciocínio, planejamento e o encadeamento de ações são atividades internas dos agentes, as ações definidas sobre o ambiente devem ser, em princípio, curtas e simples.

dos são definidos da mesma forma que para as ações, conforme visto na seção anterior. Todas as expressões na lista de precondições devem ser satisfeitas para a execução da respectiva reação. De maneira diferente das ações, onde apenas uma ação é selecionada pelo agente, todas as reações associadas aos recursos que tenham suas precondições satisfeitas serão executadas. As reações são executadas por recursos, na ordem em que aparecem na lista de reações dos recursos.

A.7 Observáveis

Nesta seção são definidas quais propriedades dos agentes, recursos e ambiente serão enviadas como resultados da simulação. A freqüência de saída destes resultados parciais depende de parâmetros definidos pelo usuário. As propriedades listadas podem ser aquelas associadas com qualquer instância de recursos, agentes, células da grade e valores de controle da simulação.

No exemplo abaixo, a cada vez que forem geradas as saídas para o usuário, esta terá como conteúdo o valor da propriedade `indiceCor` e o conteúdo de todas as células da grade. O conteúdo de uma célula é referido pela palavra reservada `CONTENTS`.

```
Observable{
    cell[ALL,ALL].indiceCor;
    cell[ALL,ALL].CONTENTS;
}
```

A.8 Valores da Simulação

Esta seção define os valores correntes das variáveis de controle do ambiente e dos valores das propriedades de cada instância dos agentes e recursos. Também nessa seção, a posição dos agentes e recursos na grade ou grafo são definidos, caso sejam definidos. Os valores das variáveis de controle do ambiente são definidos com nomes pré-definidos. Os valores para propriedades de agentes e recursos são acessados pelo nome do elemento (agente ou recurso), índice da instância específica e o nome da propriedade. As posições dos agentes e recursos na grade são determinados por construções específicas da linguagem, como nos exemplos abaixo:

```
Instance agente[3].cor = "AZUL";
```

O exemplo acima define o valor corrente da propriedade `cor` como sendo `AZUL`, para a instância de índice 3 da classe `agente`.

```
cell[1,2].ocupado = true;
cell[1,2].CONTENT = {agente[3]};
```

O código acima indica que, na posição $(1, 2)$ da grade, a propriedade `ocupado` possui valor "TRUE". Este trecho indica também que a instância de índice 3 da classe `agente` está presente na célula.

A.9 Inicialização

Nesta seção, recursos do ambiente podem ser instanciados e posicionados na grade (recursos podem ser criados dinamicamente no ambiente). Todos os comandos desta seção são executados antes do início da simulação. Os comandos possíveis são apresentados na seção A.12.

A.10 Declaração de Atributos

Como mencionado anteriormente, os tipos dos atributos possíveis são: booleano (boolean), inteiro (integer), ponto flutuante (float)[4] e texto (string). A declaração dos atributos é composta pela *tag* do tipo do atributo que contém o nome e valor inicial da propriedade. Com exceção dos atributos do tipo texto, é possível inicializar as propriedades com expressões (vistos a seguir).

A.11 Expressões

As expressões possíveis na linguagem ELMS são formadas por operadores matemáticas, lógicos e relacionais. Também é possível utilizar os comandos `RAND` e `RANDOM`. Entre os operadores relacionais estão incluídas a comparação de igualdade (`==`), desigualdade (`!=`), maior que (`>`) e menor que (`<`). Entre as operações matemáticas estão as operações de soma (`+`), subtração (`-`), multiplicação (`*`), divisão (`/`), resto de divisão (`%`), somatório (`SUM`) e produtório (`PROD`). Entre os operadores lógicos, estão disponíveis conjunção (`&`), disjunção (`|`) e negação (`!`).

Os operadores relacionais possuem dois operandos, sendo que cada operando pode ser uma operação, uma constante ou um atributo de elemento ou célula. As operações lógicas de conjunção e disjunção aceitam dois operandos, onde cada operando pode ser uma outra operação, uma constante ou um valor de atributo ou célula. A operação de negação é unária.

As operações matemáticas de soma, subtração, multiplicação, divisão e resto de divisão aceitam dois operandos, onde cada operando pode ser outra operação, uma constante ou o valor de um atributo de elemento ou célula. As operações de somatório e produtório aceitam como operador um atributo de uma classes de recurso ou agentes, tendo como resultado a soma ou produto dos valores do atributo em todas as instâncias desta classe. As operações de somatório e produtório também aceitam como parâmetro um atributo da grade, sendo possível definir-se os eixos que serão percorridos na realização da soma ou produto. Em versões futuras, planeja-se possibilitar o uso de intervalos na realização destas operações.

No exemplo a seguir é realizada a soma do atributo `total_a` de todas as células da grade.

```
SUM(cell[ALL,ALL].total_a);
```

O comando `RAND` gera um número randômico entre 0 e 1, enquanto o comando `RANDOM` tem como parâmetros um valor mínimo (inclusive) e um valor máximo (exclusivo), gerando um número randômico neste intervalo. Estes comandos podem ser utilizados em quase todas as seções do código, com exceção da seção de “valores da simulação”. No exemplo abaixo, será gerado um valor randômico entre 0 e 9.

```
RANDOM(0,10)
```

A.12 Comandos

Os comandos possíveis na linguagem ELMS são a atribuição(representado pelo operador (`=`), inserção de elemento[5] na grade (`IN`), o comando de inserção randômica

[4]Este tipo de atributo deve ser evitado dentro das percepções, pois não é suportado pela versão corrente do AgentSpeak.

[5]Refere-se como “elemento” tanto objetos (recursos) como agentes.

(IN_RAND), remoção de elementos da célula (OUT), realocação de elementos (MOVE), instanciação de elementos (NEW) e exclusão de instância (DELETE).

Os comandos de inserção e remoção de elementos na grade têm como parâmetro a classe e o índice de um elemento já instanciado e uma posição na grade onde o elemento será alocado ou removido. Há também o comando de inserção randômica, que possui como parâmetros o elemento a ser inserido (nome da classe e o índice) e a precondição que deve ser atendida pela célula para que a inserção (em uma célula aleatória mas que satisfaça a precondição) seja realizada.

O comando de realocação possui como parâmetros um elemento, uma posição de origem e outra de destino. Deve-se lembrar que um elemento pode ocupar mais de uma posição na grade e os elemento têm como ponto de referência[6] a primeira célula onde foi inserido. Com o comando de realocação, estará sendo realizada a movimentação de todo o elemento, através da mudança do ponto de referência. Tanto a especificação do índice dos elementos como as posições na grade aceitam expressões, constantes ou atributos de células ou elementos.

O comando de instanciação possui como parâmetros a classe, o número de instâncias a ser criado, uma posição (opcional) onde as instâncias serão posicionas e um rótulo (opcional) para uma variável do tipo inteira à qual será atribuído o índice da primeira instância criada. No exemplo abaixo é instanciado um recursos do tipo recurso1 que será alocados na posição (3,4) da grade. Será armazenado em uma variável inteira chamada newRef o índice instância criada. Finalmente, o comando de exclusão de instância possui como parâmetro o nome da classe e uma expressão que indica o índice da instância a ser excluída.

```
NEW (recurso1,"newRef",cell[3,4])
```

[6] O ponto de referência usado para cálculo de posições relativas.

APÊNDICE B FORMATO DAS PERCEPÇÕES ELMS

As percepções que o ambiente envia para os agentes seguem o formato de crenças AgentSpeak. Nesta seção será descrito o formato destas percepções, que são reconhecidas pelo interpretador ***Jason***, visto na seção C.2.

As percepções que compõem o conjunto de percepções podem ser atributos de células, conteúdo de células, atributos de recursos, atributos de agentes ou valores das variáveis de controle do ambiente. Cada átomo de crença, que representa uma informação da percepção, é enviado como um item em uma lista de strings. O formato de cada tipo de percepção é explicitado nas seções a seguir.

B.1 Atributos de Células

O formato de uma percepção de um atributo de célula segue a seguinte forma:

`cell(position(`X`,`Y`,`Z`),`*atributo*`(`*Valor*`))`

Onde X,Y e Z representam a posição da célula, *atributo* representa o nome do atributo (propriedade) da célula e *Valor* representa o valor da propriedade. Caso a grade seja bidimensional não será incluído o valor de Z. A coordenada da célula terá valores absolutos se a percepção foi definida com valores absolutos e terá valores relativos à posição do agente caso contrário. Caso a propriedade seja do tipo booleana, será enviado `t` para verdadeiro[1] e `f` para falso.

B.2 Atributos de Elementos

No caso de percepções de atributos de elementos, o formato do átomo de crença é:

classe`(instance(`N`),`*atributo*`(`*Valor*`))`

Onde *classe* denota a classe do elemento, N representa o índice da instância da classe, *atributo* indica a propriedade do elemento e *Valor* indicando o valor da propriedade.

B.3 Conteúdo de Célula

O conteúdo de uma célula é enviado para o agente conforme o seguinte formato:

classe`(instance(`N`)position(`X`,`Y`,`Z`))`

Onde *classe* representa a classe do elemento, N indica o índice da instância e X, Y e Z denotam a posição, sendo que Z será enviado somente se a grade seja tridimensional.

[1] "true" é palavra reservada do interpretador AgentSpeak.

B.4 Variáveis de Controle do Ambiente

As percepções contendo valores das variáveis de controle do ambiente seguem o seguinte formato:

atributo(*Valor*)

Onde *atributo* indica o nome da variável e *Valor* representa o valor da propriedade.

APÊNDICE C AGENTSPEAK E JASON

C.1 AgentSpeak(L)

A linguagem AgentSpeak(L) foi introduzida por Rao em (RAO, 1996) como uma linguagem abstrata para programação de agentes baseada no modelo BDI. Esta linguagem foi mais tarde formalizada em (d'INVERNO; LUCK, 1998) na linguagem de especificação formal Z. A linguagem AgentSpeak(L) é particularmente interessante por manter importantes aspectos de sistemas de planejamento nos quais foi baseado. Porém somente em (MACHADO; BORDINI, 2002) foi apresentado um primeiro protótipo de um interpretador para a linguagem. As relações entre a lógica BDI (RAO; GEORGEFF, 1998) e o AgentSpeak(L) são apresentadas em (BORDINI; MOREIRA, 2004) e um interpretador foi desenvolvido baseado na sua semântica operacional formal. Em resumo, a linguagem AgentSpeak(L) integra noções da arquitetura BDI e da programação em lógica, fornecendo uma abordagem abstrata elegante para a programação de agentes BDI. A arquitetura BDI, por sua vez, é a abordagem predominante para a implementação de agentes *inteligentes* ou *racionais* (WOOLDRIDGE, 2002). A seguir é apresentado a sintaxe básica da linguagem AgentSpeak(L), resumida de (BORDINI et al., 2002). Para uma descrição mais completa veja (d'INVERNO; LUCK, 1998; MOREIRA; BORDINI, 2002).

O comportamento de um agente definido em AgentSpeak(L), é descrito através da especificação de um conjunto de crenças iniciais e um conjunto de planos. As definições a seguir introduzem as noções necessárias para a especificação de tais conjuntos. Existem várias similaridades com a sintaxe do Prolog, incluindo a convenção de uso de iniciais maiúsculas para os nomes de variáveis, e aspectos relacionados à forma dos predicados e unificação.

Um *átomo de crença* é simplesmente um predicado em sua notação usual, enquanto átomos de crença ou suas negações são *literais de crença*. O conjunto inicial de crenças é apenas uma coleção de literais de crença que não dependem de variáveis.

No AgentSpeak(L) é possível definir dois tipos de objetivos (*goals*): objetivos de realização (*achievement goals*) e objetivos de teste (*test goals*). Objetivos de realização são predicados, assim como as crenças, precedidos com o operador '!' enquanto os objetivos de teste são precedidos do operador '?'. Objetivos de realização declaram que o agente quer alcançar um estado de mundo onde o predicado associado é verdadeiro. Objetivos de teste declaram que o agente quer testar se o predicado associado é verdadeiro, isto é, se este pode ser unificado com a base de crenças do agente.

Os eventos disparadores (*triggering events*) definem quais eventos podem iniciar a execução de planos. No AgentSpeak(L) são definidos dois tipos de eventos: aqueles relacionados à adição, precedidos pelo operador '+', e aqueles relacionados à remoção, precedidos pelo operador '-', de atitudes mentais(crenças ou objetivos).

De acordo com o modelo de agente tradicionais de agentes, para realizar ações no ambiente, os planos devem fazer referências à *ações básicas* que o agente é capaz de realizar no ambiente onde está inserido. Estas ações são referenciadas como predicados simples, que são símbolos predicativos especiais utilizados para este propósito.

Um plano AgentSpeak(L) possui um cabeçalho que é formado por um evento disparador que denota o propósito do plano e uma conjunção de literais de crença que forma um contexto que deve ser satisfeito para a execução do plano (o contexto deve ser conseqüência lógica da base de crenças do agente). Um plano também possui um corpo, que é formado por uma seqüência de ações básicas, objetivos e sub-objetivos de realização ou teste.

Definição de Plano: se e é um evento disparador; $b_1, \ldots, b_m$ são literais de crença; e $h_1, \ldots, h_n$ são objetivos ou ações; então [e `:` b_1 `&` ... `&` b_m `<-` h_1 `;` ... `;` h_n`.`] é um *plano*.

A expressão à esquerda da seta é referida como o *cabeçalho* do plano e a expressão à direita da seta é referida como o *corpo* do plano. A expressão à direita da ponto no cabeçalho do plano é referido como *contexto*. O corpo de um plano ou um contexto vazio é representado pela expressão "$true$".

Definição de Agente: Um agente AgentSpeak(L) é definido pela tupla $\langle E, B, P, I, A, \mathcal{S}_\mathcal{E}, \mathcal{S}_\mathcal{O}, \mathcal{S}_\mathcal{I} \rangle$, onde E é um conjunto de eventos, B é um conjunto de crenças iniciais, P é um conjunto de planos, I é um conjunto de intenções e A é um conjunto de ações. A função de seleção $\mathcal{S}_\mathcal{E}$ seleciona um evento do conjunto E; a função de seleção $\mathcal{S}_\mathcal{O}$ seleciona uma opção ou um plano aplicável de um conjunto de planos aplicáveis; e $\mathcal{S}_\mathcal{I}$ seleciona uma intenção do conjunto I.

Intenções são cursos de ações com as quais o agente se comprometeu a fim de realizar algum objetivo específico. No interpretador, cada *intenção* é uma pilha de *planos parcialmente instanciados*, ou seja, planos onde algumas variáveis foram instanciadas. Um evento, que pode iniciar a execução de planos, pode ser *externo*, quando originado da percepção do ambiente, ou *interno* quando gerado pela própria execução de um plano (a adição ou remoção de um objetivo ou crença dentro do corpo de um plano). Planos escolhidos por eventos internos serão empilhados sobre a intenção que os gerou. Enquanto planos escolhidos por eventos externos geram uma nova intenção em I, representando diferentes focos de atenção do agente no ambiente. As funções de seleção ($\mathcal{S}_\mathcal{E}, \mathcal{S}_\mathcal{O}, \mathcal{S}_\mathcal{I}$) que são consideradas partes específicas de cada agente, apesar de não terem sido definidas nas especificações da linguagem, são parte importante de um interpretador AgentSpeak(L).

A figura C.1, traduzida de Machado e Bordini (2002), descreve as estruturas do interpretador e o funcionamento geral do interpretador AgentSpeak(L) proposto por Rao. Nesta figura, conjuntos (de crenças, planos, eventos, planos e intenções) são representados por retângulos; círculos representam processamento envolvido; e losangos representam a seleção de um elemento em um conjunto.

A cada ciclo de interpretação a lista de eventos é atualizada, sendo os eventos introduzidos através da percepção ambiente ou na execução de intenções. Note que foi introduzido uma Função de Revisão de Crenças (FRC) que é implícita no interpretador de Rao, mas que normalmente é explícita na arquitetura BDI genérica. Isto assume que as crenças são atualizadas a partir das percepções e mudanças nas crenças dos agentes, o que implica na inserção de eventos no conjunto de eventos.

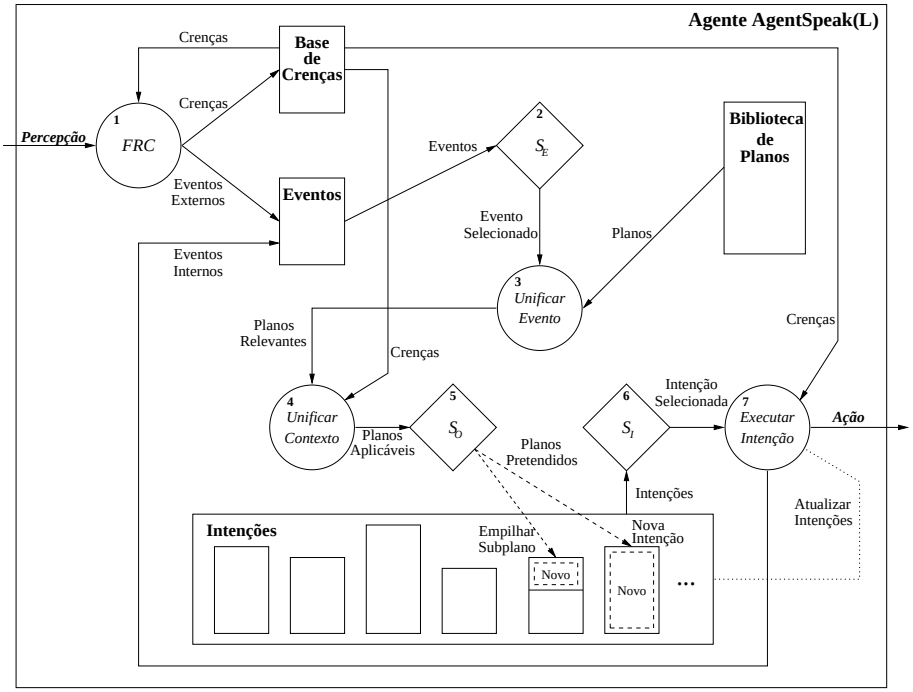

Figura C.1: Interpretação de Programa AgentSpeak(L)

Como visto anteriormente, agente AgentSpeak(L) é definido basicamente através de crenças iniciais e planos, porém as funções de seleção $\mathcal{S}_\mathcal{E}$,$\mathcal{S}_\mathcal{O}$ e $\mathcal{S}_\mathcal{I}$ também são partes relevantes da definição do agente que deveriam ser especificadas na execução do interpretador. Após o $\mathcal{S}_\mathcal{E}$ selecionar um evento, o interpretador tem que unificar este evento com os eventos disparadores dos cabeçalhos dos planos, gerando uma conjunto de todos os *planos relevantes*. Quando unificados o contexto do cabeçalho dos planos com a base de crenças, é determinado um conjunto de *planos aplicáveis*, que são planos que podem ser utilizados para o tratamento do evento. Deste conjunto, o $\mathcal{S}_\mathcal{O}$ seleciona um único plano aplicável e empilha o plano na intenção que gerou o evento, no caso de um evento interno, ou cria uma nova pilha de intenção, caso seja um evento externo. A seguir, o $\mathcal{S}_\mathcal{I}$ seleciona uma das intenções do agente e pega o comando que está no corpo do plano. Este comando poderá ser uma ação básica, ação do agente sobre o ambiente, ou a geração de um evento interno. O comando será executado, com todas as operações e atualizações envolvidas, finalizando um ciclo de interpretação AgentSpeak(L).

C.2 Jason

Jason é um interpretador para a linguagem AgentSpeak, com funcionalidades que permitem o desenvolvimento de SMA. Nesta seção são apresentadas as principais funcionalidades do interpretador, que podem ser encontradas em (BORDINI; HÜBNER; VIEIRA, 2005)

Uma das características mais importantes do ***Jason*** é implementar a semântica operacional de uma extensão do AgentSpeak. Tendo uma semântica formal, permitiu que fosse caracterizada uma definição para noções práticas de crenças, desejos e intenções existentes na execução de agentes AgentSpeak, que por sua vez sustenta os trabalhos na verificação formal de programas AgentSpeak, tais como (BORDINI; MOREIRA, 2004; MOREIRA; VIEIRA; BORDINI, 2004; VIEIRA et al., 2007).

A versão original do AgentSpeak, definida como linguagem abstrata, tinha como objetivo o trabalho teórico com lógica BDI (RAO; GEORGEFF, 1998) para implementação de sistemas de planejamento reativo (*reactive planning systems*). As extensões implementadas no ***Jason*** foram necessárias para tornar a linguagem abstrata original em uma linguagem de programação de uso prático para SMA. As extensões da linguagem apresentam as seguintes características:

Negação forte: como é de conhecimento na comunidade de linguagens de programação de agentes, não é o ideal assumir um *mundo-fechado* dentro de sistemas abertos onde a incerteza não pode ser evitada, a possibilidade dos agentes poderem referir a coisas que acreditam ser verdadeiras, falsas ou desconhecidas pode ajudar na modelagem destas aplicações.

Tratamento de falha de planos: devido a natureza dinâmica típica de ambientes multiagentes, planos podem falhar em atingir seus os objetivos propostos, ***Jason*** tem uma forma particular de mecanismo para tratamento de falhas de planos que consiste de planos que são disparados por tal falha.

Comunicação baseada em atos de fala: o fundamento filosófico para todo o trabalho na comunicação entre agentes é a teoria dos atos de fala. Como as atitudes mentais, que são usadas para dar semântica para a comunicação baseada em atos de fala, são formalmente definida para o AgentSpeak é possível especificar a semântica para como agentes interpretam as forças ilocucionárias básicas, implementadas em ***Jason***.

Anotação de Crenças: uma extensão da linguagem interessante é a possibilidade de crenças poder ter *anotações* que podem ser úteis em tarefas específicas em cada aplicação. Uma anotação padrão, automaticamente realizada pelo ***Jason***, é a marcação da *origem* de cada crença.

Anotação de Planos: da mesma forma que as crenças podem ter anotações, os programadores podem adicionar anotações nos rótulos de planos, que podem ser usada para elaborar funções de seleção. Funções de seleção são funções definidas pelo programador e que são utilizadas pelo interpretador, incluindo as que definem para quais planos devem ser dadas prioridade no caso de vários planos diferentes possam ser aplicáveis em um evento específico.

A implementação da plataforma ***Jason*** tem as seguintes funcionalidades:

Distribuição: a plataforma torna fácil a definição dos agentes que irão fazer parte do sistema e determinar em quais máquinas cada agente será executado, de acordo com a infraestrutura de execução selecionada. Na versão corrente, duas formas de infraestrutura são disponíveis, uma que executa todos os agentes em uma máquina e outra que permite a distribuição utilizando o SACI (HÜBNER; SICHMAN, 2000). As infraestruturas de execução permitem obter uma melhor performance de acordo com a forma de execução na comunicação dos agentes.

Ambientes: SMA podem ser executados em um ambiente *real*. Mesmo neste caso, durante o desenvolvimento, uma simulação do ambiente será necessária. O ***Jason*** suporta ambientes simulados que podem ser programados em Java. O interpretador ELMS (seção 3) faz uso deste suporte para permitir a definição dos ambientes na linguagem ELMS.

Customização: programadores podem alterar partes importantes da plataforma de agentes através da definição de métodos Java específicos para determinados aspectos de um agente e da arquitetura geral dos agentes. Um exemplo mais simples seria a redefinição das funções de seleção de planos, eventos e intenções.

Extensibilidade e suporte a código herdado: a extensão AgentSpeak disponível no ***Jason***, tem uma construção chamada "ações internas", no lugar de um literal pode haver uma chamada para uma "ação interna". Estas ações internas podem ser implementadas em Java como um método booleano, ou outra linguagem utilizando se a JNI (Java Native Interface). Isto fornece uma maneira direta para a extensão da linguagem através da definição de ações pelo usuário, o que permite a invocação de código herdado a partir da programação alto-nível do agente. Além das ações definidas pelo usuário, o ***Jason*** fornece uma biblioteca de ações internas pré-definidas que implementam operações úteis para a programação prática, possibilitando a implementação de programas inspirados no BDI, que não eram possíveis com a linguagem AgentSpeak original, tais como a verificação e desistência de desejos e intenções.

Ambiente de Desenvolvimento Integrado: o ***Jason*** é distribuído com um ambiente de desenvolvimento que fornece uma interface gráfica para o gerenciamento do projeto do sistema, que permite a edição dos códigos-fontes de cada agente. Outro ferramenta oferecida, permite a inspeção de estados internos do agente. Este ambiente de desenvolvimento pode ser utilizado como um plugin para o jEdit (`http://www.jedit.org/`).

Existem várias pesquisas relacionadas ao desenvolvimento do ***Jason***, mencionadas a seguir:

Modelos de Planos para objetivos declarativos: em trabalho recente, foram definidos padrões de planos AgentSpeak que podem ser utilizados para definir vários tipos de objetivos *declarativos* com estruturas temporais sofisticadas. Tais tipos de objetivos são importantes na literatura e característica essencial para programação orientada a agentes. Isto permite expressar, por exemplo, que um agente deverá persistir em um objetivo até surja alguma evidência de que será impossível de atingir determinado objetivo ou que não há mais necessidade de atingir o objetivo.

Troca de Planos: Trabalhos têm sido realizado s para permitir a troca de planos AgentSpeak entre agentes, o que pode ser bastante útil, especificamente para sistemas de agentes cooperativos, mas também para aplicação nas quais um grande número de planos não podem ser mantidos na biblioteca de planos de agente simultaneamente. Várias questões relacionadas na engenharia de sistemas onde tais trocas de planos podem acontecer devem ser consideradas, como as abordadas em (ANCONA; MASCARDI, 2004).

Raciocínio Ontológico: apesar de ainda não estar disponível no ***Jason***, em (MOREIRA et al., 2006) são abordadas questões sobre como a base de crenças deveriam ser formuladas como uma ontologia. De maneira concreta, no ***Jason*** é possível utilizar as anotações de planos para especificar qual ontologia cada crença pertence e utilizar alguma ferramenta existente para fazer o raciocínio ontológico quando necessário, o que poderá necessitar uma revisão de crenças mais apropriada.

Revisão de Crenças: no ***Jason***, a consistência de base de crenças é parte da responsabilidade do programador, devido ao fato que a revisão de crenças tem custo computacional bastante alto. Em (ALECHINA; JAGO; LOGAN, 2005), é apresentado um algoritmo para revisão de crenças em tempo polinomial. Este algoritmo, poderá ser adaptado para uso com o ***Jason***.

Nesta seção, foram abordadas de maneira resumida as principais funcionalidades do ***Jason***, maiores informações podem ser encontradas em `http://jason.sf.net`.