FACULDADE DE INFORMÁTICA
PUCRS – Brazil
http://www.inf.pucrs.br

X-BDI: uma ferramenta para programação de agentes BDI

Alexandre de Oliveira Zamberlam, Lucia Maria Martins Giraffa e
Michael da Costa Móra

TECHNICAL REPORT SERIES

Number 009
April, 2001

Contact:

alexz@inf.pucrs.br

http://www.inf.pucrs.br/~alexz

giraffa@inf.pucrs.br

http://www.inf.pucrs.br/~giraffa

michael@inf.pucrs.br

http://www.inf.pucrs.br/~michael

Alexandre de Oliveira Zamberlam is a graduate student of UNIJUÍ – Universidade Regional do Noroeste do Estado do RS – Brazil. He is member of the GIE research group (Computer Science apply to Education Research Group) since 2000. He receives a federal graduate research grant from CAPES (Brazil) to support his research.

Lucia Maria Martins Giraffa works at PUCRS/Brazil since 1986. She is a titular professor and leader of the GIE Gr. She develops research in Multi-agent systems, Artificial Intelligence apply to Education, and design of educational software. She got her Ph.D. in 1999 at UFRGS (Brazil).

Michael da Costa Móra is an assistant professor and researcher of GIE Gr. He works with Multi-agent systems, and Artificial Intelligence apply to different fields. He got his Ph.D. in 2000 at UFRGS (Brazil).

Published by the Campus Global – FACIN – PUCRS
Av. Ipiranga, 6681
90619-900 Porto Alegre – RS – Brazil

X-BDI: uma ferramenta para programação de agentes BDI

Relatório Técnico 009/2001

Alexandre de Oliveira Zamberlam (mestrando)*
Lucia Maria Martins Giraffa (orientadora)†
Michael da Costa Móra‡

1 Introdução

O interesse pelo desenvolvimento de sistemas que incorporam técnicas de modelagem e implementação baseadas na abordagem orientada a agentes têm crescido nos últimos anos. Esta abordagem permite um nível de abstração que não é usual nas metodologias tradicionais, isto é, torna explícita certas funcionalidades no sistema que antes ficavam apenas subentendidas.

O termo agente possui uma grande diversidade de definições e vem sendo utilizado para denotar desde simples processos de hardware e/ou software até entidades sofisticadas capazes de realizar tarefas complexas. Esta diversidade reflete o estado atual da área, onde não há um consenso do que realmente seja um agente. Cada grupo de pesquisa segue uma determinada linha, de acordo com seus próprios objetivos, apresentando sua definição personalizada para o termo agente. Além disso, o termo agente apresenta-se em mais de um contexto, não sendo exclusivo da área de IA, fato que acarreta maior complexidade e necessidade de cuidado quando se estuda determinado texto. Neste trabalho será utilizado como base a definição de Russell e Norvig [RUS96], que definem um agente como um sistema capaz de perceber através de sensores e agir em um dado ambiente através de atuadores.

Apesar da falta de uma definição consensual, há certas propriedades que um sistema deve exibir, a fim de que seja considerado agente, principalmente para os agentes que utilizam estados mentais (foco deste trabalho). Estas propriedades são:

*alexz@inf.pucrs.br
†giraffa@inf.pucrs.br
‡michael@inf.pucrs.br

- autonomia - a capacidade de operar sem a intervenção de humanos, e de controlar suas próprias ações e estados internos;
- habilidade social - a capacidade de interagir com outros agentes, humanos ou não, através de algum tipo de linguagem;
- reatividade - a habilidade de responder em tempo a estímulos recebidos do ambiente;
- pró-atividade - a capacidade de, além de responder a estímulos do ambiente, exibir um comportamento orientado a objetivos. Ou seja, capacidade de prever como atingir ou evitar um determinado estado ou objetivo.
- adaptabilidade - a capacidade de se adaptar a modificações no ambiente, alterando planos previamente concebidos ou aprendendo através da interação com o ambiente.

Segundo Móra [MÓR00], as definições e propriedades que caracterizam a noção de agente têm por objetivo não meramente dividir o mundo entre entidades que são e que não são agentes, mas servir de ferramenta para analisar sistemas, bem como especificar, projetar e implementar sistemas cujos elementos básicos sejam agentes.

À medida que as aplicações consideradas tornam-se maiores e mais complexas, é necessário utilizar níveis de abstração que permitam representar os problemas e suas soluções de forma mais natural. Os agentes, como entidades autônomas com capacidade de planejar suas ações, reagir e interagir entre si em busca de soluções para problemas, parecem fornecer um passo em direção a esse nível de abstração mais alto. Mas, assim como acontece com a modelagem dos sistemas, também a modelagem dos agentes requerem níveis de abstração adequados. As abordagens utilizadas são as mais diversas. Uma dessas abordagens freqüentemente utilizadas é a que considera agentes como sendo **sistemas intencionais** (maiores detalhes podem ser obtidos em [DEN87] e [MÓR00]). Sistemas intencionais são aqueles aos quais são atribuídos **estados mentais** (ou **atitudes intencionais**), usualmente atribuídos a seres humanos, tais como **crenças**, **desejos** e **intenções**. Assim, é possível descrever mecanismos de resolução de problemas dos agentes em termos do que eles acreditam, de que planos eles possuem ou constrõem a fim de satisfazer seus desejos e intenções, que atributos eles utilizam para determinar que opções eles escolhem, e assim por diante. A abordagem intencional foi adotada no trabalho de Móra (X-BDI - *eXecutable BDI*) [MÓR00], que será utilizado como base para este trabalho.

Uma abordagem intencional considera que:

- os estados mentais, como abstração, possuem um forte apelo conceitual, pois são bastante naturais para os projetistas e os analistas que utilizam a abordagem de agentes;

- fornecem descrições sucintas para sistemas complexos, além de ajudar a entender e a explicar o comportamento desses sistemas;
- podem ser usados pelos próprios agentes para raciocinar sobre eles próprios e sobre outros agentes (reciprocidade, segundo a nomenclatura adotada por Dennet apud [MÓR00]).

Este trabalho tem por objetivo apresentar aspectos relacionados à modelagem e implementação de agentes BDI utilizando a ferramenta X-BDI desenvolvida por Móra [MÓR00]. Para tanto, o trabalho foi dividido em 8 seções. A seção 2 apresenta agentes modelados com estados mentais. A seção 3 apresenta ferramentas de modelagem e implementação para agentes BDI. A seção 4 apresenta os mecanismos lógicos utilizados na construção do X-BDI. Na seção 5 é apresentada uma visão geral da teoria utilizada no X-BDI. Na seção 6 é descrita a ferramenta X-BDI na visão do usuário final, bem como são apresentados dois trabalhos que utilizaram a ferramenta X-BDI. Nas seções 7 e 8 são apresentadas as considerações finais e as referências bibliográficas utilizadas, respectivamente.

2 Agentes modelados com estados mentais

O que caracteriza o agente são as interações que ele realiza com o mundo e os processos internos que possibilitam a realização destas interações. A especificação de quais e como são estes processos internos é chamada de arquitetura do agente [GIR99]. Diferentes arquiteturas têm sido propostas com o objetivo de caracterizar os agentes com um particular nível de inteligência e de autonomia, as quais podem ser classificadas de acordo com o mecanismo utilizado pelo agente para a seleção das ações em: não-deliberativas, deliberativas e híbridas. Segundo [COR94], para definir a arquitetura interna do agente é necessário saber, inicialmente, qual o tipo de tarefa que o agente irá realizar e o seu papel na sociedade. Uma vez considerado isto, o agente pode ser classificado como:

- reativo (ou não-deliberativo): a escolha da ação (resposta) está diretamente situada na ocorrência de um conjunto de eventos (estímulos) que ele percebe no e do ambiente, captados por seus sensores ou por mensagens enviadas por outros agentes.
- cognitivo (deliberativo): possui um processo explícito para escolher a ação a ser realizada. Esta ação é realizada através de uma representação simbólica do mundo, de um plano e de uma função de utilidade.

Um agente cognitivo é um agente racional que possui alguma representação explícita de seu conhecimento e objetivos. Um agente reativo não necessariamente é um agente racional: seu comportamento pode ser definido através de um padrão de estímulo-resposta. Um agente

pode ser 'mais cognitivo' do que outro, conforme o grau de racionalidade explícita de seu comportamento [OLI96].

Arquiteturas de agentes cognitivos, segundo [OLI96], podem ser divididas em:

- arquiteturas funcionais - onde o agente é composto por módulos que representam cada uma das funcionalidades necessárias para sua operação. O agente possui conhecimento, um conjunto de objetivos, capacidade de percepção, comunicação, decisão e raciocínio.

- arquiteturas baseadas em estados mentais - adotam uma perspectiva de inspiração psicológica para definir a estrutura dos agentes, que são entidades cujo estado é visto como consistindo de componentes mentais tais como crenças, capacidades, escolhas e compromissos. Por esta razão o estado de um agente é chamado de estado mental.

A utilização de estados mentais para modelagem de agentes cognitivos é chamada de abordagem mentalista [MÓR98]. Nesta visão mentalista o que faz qualquer componente de hardware ou software ser um agente é precisamente o fato de ele poder ser analisado e controlado em termos destes componentes mentais. Assim, a questão do que é um agente fica substituída pela questão de quais entidades podem ser vistas como tendo um estado mental, pois os diferentes estados mentais desempenham papéis distintos no comportamento dos agentes [MÓR00].

Dentro das arquiteturas baseadas em estados mentais, encontra-se a abordagem BDI (*belief, desire e intention*). As idéias básicas da abordagem BDI baseiam-se na descrição do processamento interno de um agente utilizando um conjunto básico de estados mentais (crença, desejo e intenções) e na definição de uma arquitetura de controle através da qual o agente seleciona racionalmente o curso de suas ações. Algumas abordagens de arquitetura BDI agregam as noções de planos e objetivos, como por exemplo os trabalhos de Bratman **et al**. [BRA84]; [BRA87]; [BRA89], Rao e Georgeff [RAO92].

Uma crença de um agente corresponde às informações que o agente tem sobre o mundo (é o conhecimento do ambiente de forma explícita) que poderia ser incompleta ou incorreta. Crenças podem ser vistas como simples variáveis (como por exemplo, na linguagem PASCAL), mas agentes modelados na arquitetura BDI representam crenças de forma simbólica (como por exemplo, fatos em PROLOG).

Um desejo de agente (ou objetivo em um sistema) intuitivamente corresponde à tarefas estabelecidas pelo próprio agente. Agentes BDI, assim como agentes humanos, não exigem que desejos sejam logicamente consistentes. Desejo é um estado mental intencional e com potencial motivador das ações do agente, apresentando as seguintes características:

- representa uma situação ou um conjunto de situações em que o agente gostaria que o mundo estivesse;

- pode estar em conflito com as crenças do agente;
- pode apresentar simultaneamente desejos conflitantes;
- não causa diretamente as ações, mas pode, potencialmente, gerar suas ocorrências, deixando por conta das intenções a realização de tais ações.

A intuição em sistemas BDI é que o agente não é capaz, geralmente, de realizar todos os seus desejos, mesmo sendo consistentes. Agentes devem, portanto, estabelecer alguns subconjuntos de desejos disponíveis e comprometer recursos para realizá-los. Esses desejos escolhidos são intenções e um agente continuará a tentar realizar uma intenção até que acredite que a intenção foi satisfeita, ou acredite que a intenção não conseguirá ser realizada. Intenções tem um papel na deliberação dos agentes, monitorando as ações que devem ser realizadas, e servindo de limitador das futuras escolhas do agente.

2.1 Modelos formais de agentes

A partir dos estudos de Bratman [BRA84]; [BRA87]; [BRA89], surgiram diversos trabalhos que têm por base os conceitos intencionais baseados na teoria BDI, objetivando descrever os elementos básicos do agente cognitivo, funcionalidades e funcionamento. Ou seja, modelar sistemas que possuem as propriedades básicas[1] que caracterizam este agente. Segundo Móra[MÓR00], ao se modelar agentes deve-se destacar:

- *modelos formais de agentes* - modelar um agente inteligente significa *especificar* o agente, utilizando alguma linguagem de especificação formal. Como é o usual na Ciência da Computação, o modelo é feito utilizando-se uma linguagem que forneça uma abstração sobre a qual é construída a descrição do sistema. No caso da modelagem de agentes, o usual é utilizar uma linguagem baseada em lógica, como lógica de primeira ordem ou uma lógica modal, e a abstração sendo considerada aqui são os estados ou atitudes mentais. Esses modelos de agentes, definidos com estas lógicas, chamam-se de *teorias de agentes como sistemas intencionais*;
- *arquiteturas de agentes* - são descrições informais dos elementos e processos que compõem os agentes. Novamente, como é usual na Ciência da Computação, as arquiteturas fornecem um esquema com lacunas que quando preenchidas com informações específicas do domínio da aplicação formam o sistema sendo descrito.

Um modelo baseado em estados mentais deve definir os estados mentais que devem compor o agente, qual o conteúdo proposicional destes estados mentais, qual o papel de cada um deles;

[1] Autonomia, habilidade social, reatividade, pró-atividade e adaptabilidade.

a relação entre os diferentes estados mentais, como um estado mental influencia o outro, quais as restrições que uma atitude mental impõe à outra; como estes estados mentais interagem entre si para produzir comportamentos dos agentes.

Tais modelos tem dois papéis a desempenhar:

- inicialmente, devem servir como ferramenta para definir agentes, definir propriedades que estes agentes devem possuir e demonstrar, formalmente ou empiricamente, que os agentes modelados de fato possuem as propriedades desejadas;
- devem, além disso, fornecer subsídios para a implementação de agentes.

Uma questão fundamental envolvendo a abordagem BDI é que a maioria dos sistemas formais se utilizam de lógicas modais[2] para especificação dos agentes e não possuem correspondência (ambientes ou ferramentas) que permitam a sua implementação. Isto ocorre, conforme [MÓR00], devido aos seguintes fatores:

- as lógicas modais utilizadas não são, em geral, tratáveis computacionalmente. Tais lógicas, não possuem procedimentos de derivação para manipulação de suas sentenças que sejam corretos e completos com relação a semântica destas linguagens. Isto faz com que estas lógicas modais possam ser usadas como linguagem de especificação, mas não como ferramenta de representação do conhecimento ou ferramenta básica para implementar agentes. Além disso, sua intratabilidade não permite que tais mecanismos sejam implementados;
- baseados nestes modelos formais, vários sistemas foram construídos. Nos casos em que sistemas baseados em agentes foram construídos segundo um determinado modelo formal, o que se observa é que a alegada relação especificação/implementação existente entre os modelos formais e as implementações é difícil de ser estabelecida, seja porque não há mecanismos de verificação de consistência entre a especificação que usa lógica modal e a implementação, seja porque os sistemas implementam simplificações do modelo de agentes devido a impossibilidade de tratar computacionalmente os mecanismos inerentes as lógicas modais.
- além das dificuldades inerentes às lógicas escolhidas, os modelos adotam uma *perspectiva de especificação*. Os modelos enfatizam a definição de propriedades que os agentes idealmente deveriam possuir, sem se preocupar em como tais propriedades seriam construídas nos agentes. Embora isto seja típico e aceitável em especificações, contribui bastante para a existência desta distância entre especificação e implementação.

[2]Lógica Modal pode ser descrita resumidamente como a lógica da necessidade e possibilidade do que "deve ser" e do que "pode ser" Costa apud [GIR99].

Ou seja, estes modelos formais guardam uma grande distância do que seriam implementações de agentes porque os elementos utilizados para descrição - os estados mentais - são de um alto nível de abstração, enfatizando as funções e as relações entre estes estados mentais, sem detalhar os processos a eles associados e que, em última instância, levam o agente à ação. As lógicas modais, enquanto bastante adequadas para descrever estes estados mentais e como ferramenta de análise, não dispõem de mecanismos que permitam aos agentes utilizá-la para representar e manipular a informação.

3 Ferramentas para agentes BDI

A busca em diminuir a distância entre a formalização e a implementação de agentes BDI é, hoje, um grande foco da pesquisa nesta área. Além desta distância dos aspectos de implementação, os agentes representados pelos modelos teóricos existentes nem sempre apresentam as características esperadas. Em particular, o que se deseja é possuir uma ferramenta que permita modelar agentes que exibam um comportamento autônomo e flexível. Em [MÓR00] é descrito uma situação que exemplifica bem o que se espera poder modelar (e implementar) se existir uma ferramenta adequada. O texto a seguir foi extraído do original de Móra [MÓR00].

"No edifício de gabinetes dos professores de uma Universidade, um robô autônomo tem por função transportar documentos e pequenas cargas entre os gabinetes. Este robô deve, ainda, conduzir os visitantes que chegam ao edifício até o gabinete do professor com quem este deseja falar.

Em um dado instante, o robô recebe um chamado do gabinete 310, no terceiro andar, para apanhar um envelope e entregá-lo à secretária, no andar térreo, onde o robô encontra-se naquele instante. O robô, então, planeja um curso de ações capaz de levá-lo a cumprir sua tarefa: ele deslocar-se-á até o elevador e, após acionar e aguardar a chegada do elevador, subirá ao terceiro andar e apanhará o envelope, levando-o de volta à secretaria.

Elaborado o plano, o robô passa a ação e começa a executá-lo. Sem nenhum percalço o robô chega ao gabinete 310, apanha o envelope e começa o caminho de volta. Ao chegar ao elevador, no entanto, o robô observa que há algum problema, pois ao pressionar o botão, a luz deste não se acende. Não podendo usar o elevador, o plano previamente traçado por ele não pode mais ser executado. Assim, o robô reconsidera suas opções e decide tomar a rampa de acesso aos andares e, através dela, chegar a secretaria e entregar o envelope.

Possuindo novamente um plano de ação, ele passa a executá-lo. Enquanto está a caminho, o robô recebe duas novas ordens: levar um envelope da secretaria para o gabinete 330 e conduzir um visitante ao gabinete 312 (ambos no terceiro andar). Analisando suas novas ordens, ele conclui ser mais adequado fazer a entrega do envelope que ele tem no momento

antes de atender as duas outras ordens (pois ele terá que ir à secretária de qualquer forma para encontrar o visitante e apanhar o novo envelope).

Enquanto se dirige a secretaria, o robô constrói um plano para atender suas novas ordens: ele apanhará o envelope, encontrará o visitante e conduzi-lo-á ao gabinete 330, para depois fazer a entrega do envelope. Existiria outra opção, qual seja entregar o envelope primeiro. Mas ele tem como prioridade atender aos visitantes, para não fazê-los esperar ou caminhar desnecessariamente.

Após cumprir sua ordem corrente, o robô passa a executar seu novo plano. No entanto, enquanto se dirigia ao gabinete 312, após conduzir o visitante ao gabinete 330, o robô sensora uma baixa carga em suas baterias. Imediatamente, ele dirige-se ao ponto de recarga mais próximo. Feita a recarga da bateria, ele retoma o plano interrompido, entrega o envelope e retorna a secretaria."

São agentes que exibam comportamentos com as características básicas do exemplo acima que se deseja modelar. Esse agente, para ser considerado inteligente deve possuir: autonomia (o robô decide por si que ordens priorizar, que ações tomar para satisfazer estas ordens), habilidade social (o robô recebe ordens dos professores e funcionários, comunica-se com os visitantes para conduzi-los ao seu destino), reatividade (o robô locomove-se sem esbarrar em obstáculos, imediatamente recarrega a bateria ao detectá-la com pouca carga), pró-atividade (o robô é capaz de construir planos para chegar nos gabinetes desejados, prioriza o atendimento dos visitantes para que estes não esperem ou caminhem em demasia) e adaptabilidade (o robô re-planeja quando constata que o elevador está com problemas, interrompe a entrega do envelope quando percebe que tem pouca carga na bateria e retoma a entrega depois de fazer a recarga).

As ferramentas apresentadas a seguir, conforme seus idealizadores, foram desenvolvidas com a intenção comum de diminuir a distância que existe entre teorias formais para especificação de agentes cognitivos e aplicações, tais como arquiteturas abstratas para construção de agentes de software e linguagens orientadas a agente.

3.1 AgentSpeak(L)

AgentSpeak(L) [RAO96] é uma linguagem que pode ser visualizada como uma abstração da arquitetura PRS (Procedural Reasoning System) [GEO86]; [GEO89] e que permite que programas baseados em agentes sejam escritos e interpretados de maneira similar à programas lógicos baseados em cláusulas de horn. É uma versão textual e simplificada da linguagem dMARS. AgentSpeak(L) como linguagem de especificação consiste de um conjunto de crenças

básicas (ou fatos, no sentido de programação lógica) e um conjunto de planos. Planos são sensíveis ao contexto, são receitas invocadas por eventos que permitem decomposição hierárquica de objetivos, bem como a execução de ações. Embora que sintaticamente planos pareçam cláusulas absolutas de linguagem de programação lógica, eles são bastante diferentes em seus comportamentos. Em tempo de execução um agente pode ser visto como consistindo de um conjunto de crenças, planos, intenções, eventos, ações e funções de seleção.

3.2 dMARS^{TM} distributed Multi-Agent Reasoning System

dMARS [d'IN98] é uma ferramenta integrante de um ambiente para desenvolvimento de agentes de software (sistemas orientados a agentes), baseada em C++. Foi desenvolvida no AAII (*Australian Artificial Intelligence Institute*), sendo uma ferramenta comercial voltada para aplicações em domínios dinâmicos, de conhecimento incerto e complexo, tais como telecomunicações, viagens espaciais, tráfego aéreo, gerência de negócios, etc. Projetada para configuração rápida e facilidade de integração, o ambiente facilita o projeto de sistema, manutenção e reengenharia. O modelo BDI é operacionalizado em agentes dMARS através de planos. Cada agente tem uma biblioteca de planos (receitas/fórmulas) especificando o curso da ação que pode ser tomado pelo agente para realizar suas intenções.

3.3 X-BDI: eXecutable BDI

O X-BDI é uma ferramenta (modelo) que permite a descrição formal de agentes cognitivos baseados no modelo BDI, sendo ao mesmo tempo, uma linguagem para implementação de agentes. Segundo Móra [MÓR00], sua proposta disponibiliza um sistema formal cuja linguagem é adequada para a representação de conhecimento. Este sistema fornece suporte para diversos tipos de raciocínios, tratados de forma computacional, além de fornecer as ferramentas necessárias para se modelar os estados mentais da tríade BDI.

O modelo possui duas características :

- serve como ambiente de especificação de agentes, em que é possível definir formalmente um agente através da descrição dos seus estados mentais (crenças, desejos e intenções) e de suas respectivas propriedades. Este ambiente pode, também, fazer a verificação destas propriedades;

- serve como um ambiente de implementação de agentes. O X-BDI é um ambiente onde se formaliza o agente e se o executa, verificando se o comportamento desejado ocorreu ou não.

Conforme em [MÓR00], a abordagem utilizada no X-BDI consiste em adotar uma perspectiva do agente na construção do modelo, ou seja, ao invés de enxergar o modelo simplesmente

como uma ferramenta a ser utilizada pelo projetista, vê-lo também como uma ferramenta a ser utilizada pelo agente para representar os estados mentais e raciocinar sobre eles.

O X-BDI possui um ambiente de implementação com alto nível de abstração (os estados mentais), o que reduz a complexidade no desenvolvimento de sistemas baseados em agentes. Isto significa que o projetista da área de domínio (aplicação) apenas tem de colocar suas heurísticas a respeito do processo a ser modelado, a fim de guiar o seu comportamento. A linguagem do X-BDI possui baixa complexidade no que diz respeito à sintaxe, facilitando sobremaneira o trabalho de especificação e implementação de agentes baseados na arquitetura BDI. A arquitetura BDI passa a ser um paradigma de implementação de agentes cognitivos, devido a possibilidade de se poder excetuar o modelo formal.

Segundo Móra [MÓR00], o X-BDI, como modelo formal de agentes, reduz a distância entre especificação e implementação de agentes. Adotando o ELP (*Extended Logic Programming*) como formalismo de apoio, preserva-se tanto as principais características dos modelos formais (habilidade para definir e verificar formalmente agentes), como fornece mecanismo para agentes raciocinarem. Além do que, o modelo X-BDI tem a vantagem de modelar aspectos estáticos e dinâmicos de estados mentais pró-ativos[3].

4 Os Mecanismos Lógicos

4.1 Programação em lógica estendida- ELP

O ambiente base utilizado para a construção do X-BDI usa programação em lógica estendida com negação explícita (ELP - *Extended Logic Programming with explicit negation*) com a semântica WFSX (*Well-Founded Semantics eXtended for explicit negation*) que vem a estender as possibilidades de programas em lógica normal com uma segunda negação chamada de negação explícita. Essa extensão permite representar explicitamente informações negativas e aumentar os recursos da linguagem [MÓR98].

Conforme [MÓR98], na programação em lógica normal uma proposição é falsa somente se ela não pode ser provada como verdadeira, enquanto que na lógica estendida existe um segundo tipo de negação que permite representar explicitamente informações negativas. Conforme [ALF96], informação negativa explícita apresenta uma importante função no discurso natural e no raciocínio de senso comum. A representação de alguns problemas na programação lógica seria mais natural se programas lógicos tivessem algumas maneiras de representar explicitamente a falsidade. Considere por exemplo o enunciado [ALF96]:

[3]Searle apud [MÓR00], quando define o que é intencionalidade, classifica os estados mentais em duas categorias: estados mentais de informação e estados mentais pró-ativos, que são aqueles que, de alguma forma, conduzem a ação do agente .

"*Pingüins não voam*"

Uma maneira de representar este enunciado dentro da programação lógica poderia ser:

$no\_fly(X) \leftarrow pinguim(X)$
ou equivalentemente:
$fly'(X) \leftarrow pinguim(X)$

Mas esta representação não captura a conexão entre o predicado $no\_fly(X)$ e a predicação de voar. Isto torna-se claro se for desejado representar o enunciado:

"*Pássaros voam*"

Este enunciado pode ser representado por

$fly(X) \leftarrow bird(X)$

Mas então, nenhuma conexão existe entre os predicados $no\_fly(X)$ e $fly(X)$. Intuitivamente, é desejável ter uma conexão estabelecida. A importância destas conexões aumenta se a informação negativa representar exceções para as regras [ALF96]. O primeiro enunciado, apresentado acima, pode ser tido como uma exceção para a regra geral, que pássaros, normalmente, voam. Neste caso se quer, realmente, estabelecer a conexão entre voar e não voar.

Assim, se por um lado o acréscimo desse segundo tipo de negação aumenta a expressividade da linguagem, ela também introduz a possibilidade de informações contraditórias nos programas. Quando se lida com estados mentais pró-ativos, como desejos e intenções, é necessário:

- em algumas situações, representar simultaneamente informações contraditórias, definindo uma semântica e procedimentos que representem a contradição e a levem em consideração;
- em outras, detectar e resolver contradições que surjam durante a manipulação dos estados mentais. Programas contraditórios não são usuais, pois introduzem problemas para quem deve lidar com eles. Surge então, a necessidade de definir-se uma estratégia para remoção das contradições, restaurando assim, a consistência do programa.

Portanto, quando as contradições surgem, o ambiente fornece mecanismos para removê-las. A semântica WSFX apresenta mecanismos de detecção e o SLX (*Selected Linear resolution for eXtended programs*), que é um procedimento de derivação descendente, completo e correto

em relação a semântica WFSX, apresenta mecanismos para remoção de contradições, computando as combinações possíveis das alterações de valores verdade, garantindo a remoção. Segundo [MÓR00], esse procedimento permite provar que determinado literal pertence ou não ao modelo do programa, sem exigir o cálculo do modelo completo deste programa. Através desse mecanismo de remoção é possível, além de manter a consistência, descrever diferentes tipos de raciocínio não-monotônicos. Para a situação, existe o interesse em duas formas de raciocínio não-monotônico:

- ao se fazer a seleção das intenções a partir dos desejos é necessário selecionar subconjuntos consistentes do conjunto de desejos do agente, o que é feito através de *raciocínio revogável*[4];
- também, a derivação de intenções secundárias a partir das intenções primárias é semelhante a um processo de planejamento, sendo feita por *raciocínio abdutivo* (abdução).

Um programa em lógica estendida é um conjunto de regras do tipo:

$$H \leftarrow B_1, ..., B_n, not\, C_1, ..., not\, C_m \quad (m, n \geq 0)$$

onde H, $B_1, ..., B_n$, $C_1, ..., Cm$ são literais objetivos. Um literal objetivo é também um átomo A ou sua negação explícita $\neg A$. O símbolo *not* significa a negação por omissão ou negação por falha, e $not\, L$ é uma literal por omissão. Literais podem ser literais objetivos ou literais omissão, sendo que $\neg\neg L \equiv L$.

A linguagem também permite tratar da consistência das restrições através de $A \Leftarrow B_1, ... B_n$, $not\, C_1, ..., not\, C_m \quad (m, n \geq 0)$ onde H, $B_1, ..., B_n$, $C_1, ..., Cm$ são literais objetivos, instanciando que A pode acontecer se o seu corpo acontece também. Particularmente, quando $A = \bot$, onde $\bot$é a *contradição*. Isso significa que uma contradição aparece quando o corpo da contradição é válida.

Por fim, a estrutura da ELP, além de prover um mecanismo para remover contradições, também provê procedimentos computacionais para prova de teorias que se expressam em sua linguagem. Os benefícios da utilização de um modelo com estas características são fornecidas pelo formalismo. O requisito básico para a lógica selecionada é que ela possa ser utilizada tanto para a especificação como para implementação do sistema. E, de fato, esta é a maneira usual como a linguagem é vista na programação em lógica.

Segundo Móra [MÓR00], quando se lida com estados mentais pró-ativos e comportamentos do agente a partir destes estados mentais, é preciso representar e manipular tanto ações que

[4]Do inglês *defeasible reasoning*.

o agente executa ao longo do tempo, quanto propriedades que devem se verificar ao longo do tempo como conseqüência das ações. A programação em lógica fornece uma linguagem de uso geral para representação do conhecimento e que, portanto, não possui primitivas específicas para representação de ações e tempo. Assim, o formalismo, adotado no X-BDI, que permite representar ações e tempo, bem como raciocinar sobre eles é o Cálculo de Eventos (*Event Calculus*).

4.2 Cálculo de eventos - EC

Inicialmente proposto por Kowalski e Sergot apud [MÓR00], para superar algumas limitações do Cálculo de Situações, o cálculo de eventos possui como primitivas na sua ontologia[5] os eventos, que são ocorrências de ações, as quais iniciam e terminam em períodos em que as propriedades se verificam. Para o modelo X-BDI, foi construído mais uma variação do Cálculo de Eventos, a partir da versão de Quaresma e Lopes [QUA97], que permite lidar com ações instantâneas concorrentes e com propriedades que ocorrem sem a intervenção do agente. Estas características permitem modelar eventos que ocorrem espontaneamente no ambiente, ou ações executadas por outros agentes e que não são percebidos pelo agente no momento de sua execução. Ou seja, esta proposta permite que eventos:

- tenham uma duração e uma identificação ao invés de serem instantâneos;
- e identificados no instante de tempo em que o evento ocorre.

A conseqüência disto, é que eventos podem ocorrer simultaneamente.

Descrita a base formal necessária para o entendimento da ferramenta X-BDI, passa-se à descrição dos estados mentais que a compõe. Inicialmente são definidos os aspectos estáticos do modelo: que são os três estados mentais básicos, seu conteúdo proposicional, e as condições e restrições que tais estados mentais devem satisfazer, como por exemplo, o conjunto de crenças deve ser consistente, o conjunto de intenções deve ser consistente e consistente com as crenças e assim por diante.

A seguir, definem-se os aspectos dinâmicos dos estados mentais, como por exemplo, como são criadas as intenções, como são produzidas as ações a partir das intenções, como são resolvidos os conflitos nos desejos, e assim por diante.

Conforme [MÓR00], é importante observar que o X-BDI preocupa-se com o comportamento do agente a partir dos estados mentais. Conseqüentemente, a preocupação central é com os estados mentais pró-ativos que estão diretamente ligados a produção de ações. Portanto, questões como métodos de manutenção e revisão de crenças não fazem parte do objetivo da

[5]Especificação de uma conceitualização.

ferramenta X-BDI, apesar das crenças serem definidas e os processos relacionados aos estados mentais terem a necessidade de tratar as crenças [MÓR00]. Estes são os elementos que formam o modelo do agente (maiores detalhes na seção 5.1).

5 Uma Visão Geral da Teoria do X-BDI

A seguir, são apresentadas definições do modelo BDI.

5.1 Modelo do agente

Para a definição do modelo BDI, parte-se da análise de Bratman [BRA89] sobre intenção, sua função no raciocínio lógico e como se relaciona com crenças e desejos. De acordo com Bratman apud [MÓR98], desde que agentes são tidos como limitados a recurso, eles não podem avaliar continuamente suas crenças e desejos contraditórios para agirem racionalmente. Depois de alguns raciocínios, agentes precisam se comprometer com um conjunto de escolhas. Estas escolhas seguidas por um comprometimento, são o que caracterizam as intenções. O modelo proposto não define um agente completo, mas somente a estrutura cognitiva que é parte do modelo do agente.

Neste trabalho, a formalização matemática (descrição formal) não será tratada. A ênfase se dará no mecanismo da teoria, que explica o funcionamento da ferramenta.

Definição 1 - Estrutura do agente cognitivo

A estrutura cognitiva contêm os estados mentais que compõem o agente e as regras que administram a interação destes estados mentais (e, conseqüentemente, o comportamento do agente).

Desejos são relacionados ao estado de mundos[6] que o agente quer provocar. Mas desejos, no sentido usualmente apresentado, não necessariamente levam o agente à ação. Isto é, o fato de um agente ter um desejo não significa que ele irá agir para satisfazê-lo. Significa que antes de um determinado agente decidir o que fazer, ele passa por um processo de racionalização e confronta os seus desejos (o estado de mundos que quer provocar) com suas crenças (as circunstâncias atuais e restrições que o mundo impõe). O agente escolherá os desejos que são possíveis de acordo com algum critério. Em outras palavras, desejos constituem o conjunto de estados entre os quais o agente escolhe o que fazer. Note que, desde que agentes não estejam comprometidos com seus desejos, eles não precisam ser consistentes, nem com outros desejos, nem com outros estados mentais [MÓR98].

Definição 2 - Conjunto de desejos

[6]Do inglês *state of affairs*.

A definição de desejos, nesse modelo, permite que o agente tenha um desejo com uma certa propriedade se verificando/acontecendo (ou não) em um instante específico do tempo. Cláusulas de desejos podem ser fatos, representando estados que o agente pode querer realizar/atingir quando possível, ou regras, representando estados a serem atingidos quando uma certa condição acontecer (ou se verificar). Os atributos associados para cada desejo, definem propriedades como urgência, importância ou prioridade, que são usadas pelo agente para selecionar o desejo mais apropriado [MÓR98].

Crenças constituem a informação sobre a atitude do agente. Elas representam as informações que o agente tem sobre o ambiente e sobre si próprio.

Definição 3 - Conjunto de crenças

Assume-se que o agente atualiza continuamente suas crenças para refletir mudanças que detecta no ambiente, e sempre que uma nova crença é somada ao conjunto de crenças a consistência é mantida.

Como descrito anteriormente, intenções são caracterizadas por uma *escolha* de um estado de mundos a atingir e um *comprometimento* com esta escolha. Assim, intenções são vistas como um compromisso que o agente assume com um futuro específico possível. Isto significa que, diferentemente dos desejos, uma intenção não pode ser contraditória com outras intenções. Além disso, significa que não seria racional para um agente agir para alcançar estados incompatíveis. Intenções deveriam ser apoiadas pelas crenças do agente, isto é, não seria racional para um agente pretender algo que não acredita ser possível. Uma vez que uma intenção é adotada, o agente procurará satisfazer esta intenção e planejará ações para realizá-la. O replanejamento de ações ocorre sempre quando um fracasso acontece. Estas ações, que são usadas para realizar intenções, também devem ser adotadas como intenções por parte dos agentes.

Definição 4 - Conjunto de intenções

A definição de intenções obriga a criação de restrições para a racionalidade:

- um agente não deveria pretender algo num tempo passado;
- um agente não deveria pretender algo que acredita já estar satisfeito ou que será satisfeito sem esforços por parte do agente;
- e um agente só pretende algo que acredita ser possível de ser alcançado, isto é, se acredita que exista um curso de ações que o conduza ao estado intencional de mundos.

Quando se projeta um agente, especifica-se suas crenças e os seus desejos. O agente é que vai selecionar apropriadamente suas intenções a partir de seus desejos. Essas restrições de racionalidade também devem ser garantidas durante esse processo de seleção [MÓR98].

5.1.1 Originando intenções

Uma vez caracterizada as intenções e os estados mentais relacionados, é necessário definir como estes estados mentais interagem para produzir o comportamento do agente, ou seja, como agentes selecionam intenções e quando e como agentes revisam as intenções selecionadas.

Agentes selecionam as intenções de duas fontes diferentes: de seus desejos e de um refinamento de outras intenções. Por definição, não há nenhuma restrição nos desejos do agente. Então, um agente pode ter desejos contraditórios (conforme já descrito anteriormente). Por outro lado, intenções são restringidas através de restrições de racionalidade. Assim, os agentes têm que selecionar só os desejos relacionados com aquelas restrições.

Definição 6 - Desejos elegíveis

Desejos elegíveis são aqueles desejos que o agente acredita não estarem satisfeitos. De acordo com as restrições de racionalidade na seção 5.1, não é racional para um agente pretender alguma coisa que acredita que já realizou ou que é impossível de realizar. Se um desejo é condicional, então o agente pode acreditar que esta condição é verdadeira.

Como no conjunto inicial de desejos, desejos elegíveis podem também ser contraditórios. Então, é necessário determinar subconjuntos de desejos elegíveis realizáveis através de um esforço comum. Assim, pode-se indicar quais desses subconjuntos são preferidos a serem adotados como intenções. Isso é feito através de uma relação de preferência [MÓR98].

Definição 7 - Relação de preferência de desejos

De acordo com a teoria definida em [MÓR98], o agente pode preferir satisfazer primeiro o desejo mais importante. Adicionalmente, o agente vai preferindo os mais importantes. Assim, o agente adota a maior quantidade de desejos que puder.

A relação de preferência é um relação de pré-ordem. Baseado nessa ordem de preferência, define-se o grafo de preferência que será usado para revisar os estados mentais e o processo de revisão.

Definição 8 - Grafo de preferência de desejos

A definição de grafo de desejos inicia pela definição do conjunto de literais revisáveis, isto é, o conjunto de literais que terá o valor verdade alterado quando acontecer a revisão para selecionar o subconjunto de desejos. De acordo com a definição [MÓR98], quando o conjunto de desejos elegíveis for revisado, as revisões preferidas serão aquelas que eliminarem, primeiramente, os desejos menos importantes e a quantidade mínima possível de desejos, como será descrito a seguir.

Definição 9 - Conjunto de desejos candidatos

No processo de revisão, há uma mescla de raciocínio abdutivo e revogável (ver seção 4.1), onde o literal $unsel(D)$ (ver [MÓR98] para maiores detalhes) é revogável. Seu significado intuitivo é "*Desejo D pode não ser selecionado como intenção*". Se o agente acredita que é possível satisfazer todos os seus desejos (se ele pode abduzir ações que satisfaçam todos os desejos e todas as restrições), ele encontrará uma revisão que contenha somente $happens/3$ e $act/2$. Quando restrições não podem ser todas satisfeitas, significa que a adoção de todos os desejos como intenções levam à contradição, isto é, elas não são aceitas através de um esforço comum.

Note que contradições não surgem se as ações necessárias para satisfazer duas intenções diferentes apresentarem efeitos contraditórios. Se uma ação necessária para satisfazer uma intenção cancela uma propriedade que é também uma intenção, uma restrição é violada e o curso daquela ação é rejeitado. De acordo com o grafo de preferência, as contradições que representarem os desejos menos importantes serão "destruídas" e os desejos mais importantes serão mantidos, se possível.

Como mencionado anteriormente, a relação de preferência de desejos não é uma relação de ordem. Então, é possível ter mais do que um conjunto candidato depois de uma revisão. Contudo, se for considerado a executabilidade e atributos dos desejos como critério de decisão, não faz diferença para o agente adotar qualquer dos conjuntos de desejos candidatos [MÓR98].

Definição 10 - Intenções primárias

Intenções como refinamentos de intenções: Assim que o agente adotar suas intenções, ele começará a planejar como realizá-las. Durante o planejamento, o agente formará as intenções que são relativas ou preexistentes, significando que as intenções existentes são refinadas. Este refinamento pode ser feito de vários modos, como por exemplo, um plano que inclui uma ação que não é diretamente executável, envolve um conjunto de ações com uma ordem temporal [MÓR98]. Desde que o agente se comprometa com as intenções adotadas, as intenções adotadas previamente restringem a adoção de novas. Ou seja, durante a elaboração de planos, é adotada somente uma nova intenção, em potencial, não contraditória com as intenções e com crenças existentes.

Definição 11 - Intenções relativas

A condição de não contradição reforça novamente a noção de comprometimento, isto é, uma vez adotada uma intenção, ela restringe a adoção de novas intenções.

5.1.2 Revisando intenções

Na subseção anterior, foi definido como o agente seleciona suas intenções. Como descrito anteriormente, pesar motivações e crenças significa:

- encontrar inconsistências em desejos concorrentes; checar a validade dos desejos de acordo com crenças e intenções;
- resolver restrições impostas por intenções e desejos. Isto é, atividades de raciocínio com alto custo.

O próximo passo é definir quando o agente deve executar o raciocínio sobre as intenções. Não é suficiente declarar que um agente deve revisar suas intenções quando acredita que acontece uma certa condição, assim como acreditar que uma intenção foi satisfeita ou que não é possível satisfazê-la. O que o agente precisa fazer é verificar constantemente suas crenças. Ao invés de instanciar as intenções, é necessário definir um mecanismo que ative/dispare o processo de raciocínio sem impor um fardo adicional ao agente. A abordagem disponibilizada por [MÓR98] permite definir as condições que fazem o agente começar a raciocinar sobre as intenções como restrições em cima de suas crenças. Sempre que fatos novos são incorporados, o agente tem que manter constantemente suas crenças consistentes. Da mesma forma, sempre que uma contradição aparece, o agente revisa suas crenças. O processo de revisão de intenções é disparado quando uma das restrições é violada.

Definição 12 - Gatilho de intenções[7]

As condições que foram definidas até aqui, são usualmente definidas por modelos formais de agentes. Como visto, essa caracterização de intenções pode levar para um comportamento extremo. Então, é preciso adotar restrições adicionais que evitarão comportamentos indesejáveis. Entende-se que as mesmas razões que originaram as intenções podem ser usadas para romper o comprometimento associado a elas. Se é aceito que uma intenção é originada a partir de desejos, é sensato declarar que não é racional persistir com uma intenção que suas razões são substituídas por razões mais urgentes ou importantes. O comportamento normal do agente seria determinar seus desejos e crenças concorrentes e selecionar suas intenções. O agente se comprometeria com estas intenções e constituiria um filtro de admissibilidade para outras intenções. Também, o agente tentaria satisfazer aquelas intenções, até a realização completa ou até a detecção de impossibilidade, ou até que alguns de seus outros desejos que não foram selecionados anteriormente tornar-se-iam elegíveis, ou até que os desejos que as originaram não fossem mais elegíveis, reativando o processo de revisão que determinaria (novamente) desejos e crenças concorrentes. Desde que a noção de comprometimento é preservada, o agente não inicia seu processo toda vez que existisse uma mudança nas crenças, mas somente se condições relevantes dispararem o processo de revisão de intenção, alterando o foco de atenção do agente. Estes disparos são determinados pelas pré-condições de desejos.

Definição 13 - Restrições de gatilho de desejos

[7]Do inglês *trigger from intention*.

O primeiro gatilho restringido é formado pelas pré-condições daqueles desejos que não foram elegíveis e que são mais importantes do que aqueles que foram avaliados. Isto é, se as pré-condições de tais desejos tornam-se verdadeiras, estes desejos (que não foram considerados durante o processo de racionalização) tornam-se elegíveis. Então, é necessário reavaliar desejos e crenças para checar se seus novos desejos podem ser produzidos. A segunda restrição é formada pela pré-condição dos desejos elegíveis que, embora mais importantes, não foram relevantes quando o agente fez sua escolha. Note que não existe gatilhos para desejos que foram elegíveis, mas que foram rejeitados durante a seleção de uma revisão. Isto se deve porque eles já tinham sido avaliados e por terem sido considerados menos importantes do que os outros desejos. Então, não é de utilidade disparar todo os processos novamente (isto é, para trocar a atenção do agente) para reavaliá-los.

6 A Ferramenta

Após a descrição textual de como as informações no modelo X-BDI são interpretadas, faz-se necessário uma descrição da ferramenta X-BDI, seguindo o ponto de vista do usuário final.

A ferramenta X-BDI tem como função implementar a parte cognitiva do agente; e o restante do agente (sensores, atuadores, interface) são implementados com qualquer linguagem. Assim, uma funcionalidade da ferramenta é a portabilidade, isto é, os sensores, atuadores e interface se mantêm integrados com a parte cognitiva, pois conseguem a comunicação através de um *socket*[8] (ver figura 1) .

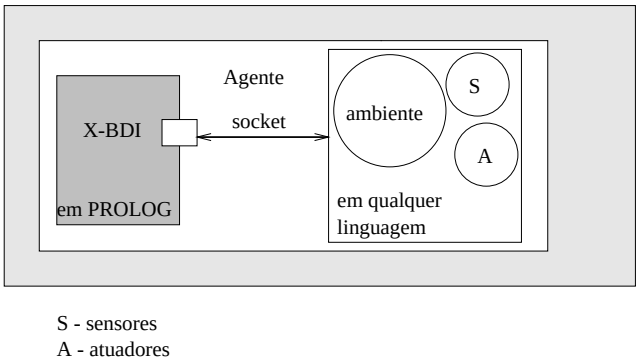

Figura 1: Interação do agente com o ambiente. Uma visão física.

As especificações de crenças (incluindo ações) e desejos devem ser feitas em um arquivo chamado **bdi.a** que é lido no momento da carga do X-BDI. No início deste arquivo deve ser feita a identificação do agente (*identity*(Agente)), sendo que essa informação não pode ser sensorada (percebida no ambiente).

A ferramenta apresenta dois modos de operação:

[8]Um *socket* é um ponto final de uma comunicação bi-direcional entre dois programas executados na rede, o qual está ligado a um número de porta [JAV01].

1. Normal: a conexão é via *socket*, por onde a ferramenta X-BDI envia os planos a serem executados e recebe informações percebidas no ambiente. (ver figura 2).

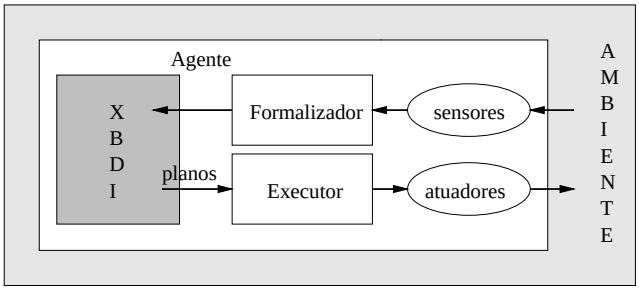

Figura 2: Interação do agente com o ambiente (figura extraída de [ROZ00]). Uma visão funcional.

2. Coreografia: onde a ferramenta lê o arquivo **coreografia.a** e simula o sensoramento do ambiente (ver figura 3) e os planos são gerados na tela. No arquivo coreografia.a ficam informações do tipo:

$$[sense(propriedade, tempo), sense(...), ...].$$

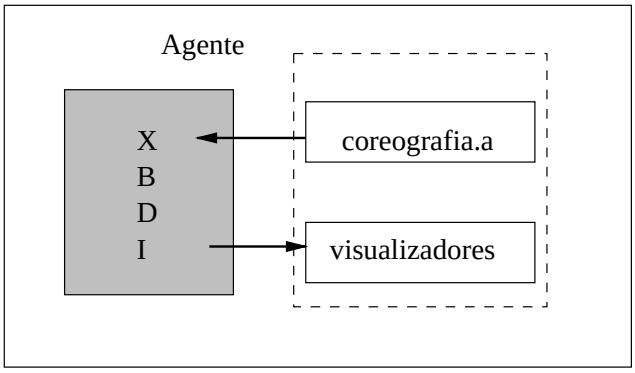

Figura 3: Atuação do arquivo que simula a interação.

6.1 O arquivo bdi.a

Ações

Ações devem ser representadas pelo predicado *act*, da forma:

act(agente, ação), onde a identificação do agente é opcional, podendo também ser representadas expressões do tipo:

act(Agente,Ação) CAUSES	$propriedade_1$, ..., $propriedade_n\ if$
	$propriedade_1$, ..., $propriedade_m$. ou
Ação CAUSES	$propriedade_1$, ..., $propriedade_n\ if$
	$propriedade_1$, ..., $propriedade_m$.

Crenças

Crenças são representadas pelo predicado *bel* :

bel(Agente,Propriedade,Tempo), onde a especificação do tempo é opcional, fato que nos permite representar crenças no futuro, ou seja, as expectativas do agente, ou acreditar em propriedades que se verificam em algum instante de tempo no passado.

Desejos

Desejos utilizam-se do predicado *des* :

des(Agente,Propriedade,Prioridade). A especificação de uma prioridade é facultativa e não é obrigatória e, se especificada deve representar valor entre um e zero inclusive.

Tempo

O tempo é controlado por vários predicados:

current_time(tempo), indica o tempo corrente do sistema, sendo que "tempo" não pode ser uma operação aritmética do tipo:

$$current\_time(t_1 + t_2).$$

Observação

current_time(tempo) é uma propriedade sensorada, ou seja, o X-BDI não faz a correção do tempo.

$$before(tempo)$$

$$after(tempo)$$

onde "tempo" é um tempo antes ou depois do tempo corrente.

Os predicados $before$ e $after$ podem ser representados de forma explícita e de forma implícita:

$current\_time(t_3)$.
bel(Agente,Propriedade,t_1). – forma implícita

$current\_time(t_3)$.
$before(t_1)$.
bel(Agente,Propriedade,t_1). – forma explícita

O predicado $after$ funciona de forma análoga.

6.2 Exemplo - Robô do armazém

Este é um exemplo para ilustrar o uso da ferramenta. Suponha que exista um robô rbt que possa carregar todos os objetos O colocados em uma entrada (como um balcão) e armazená-los em um armazém. Ele também recarrega suas baterias toda hora que elas descarregarem.

Assim,

1. $identity(rbt)$.
2. $des(rbt, bat\_carregada, 0.5)$.
3. $des(rbt, guardado(O), 0.3)$ if $bel(rbt, entrada(O))$.
4. $act(rbt, carregar)$ $causes$ $bel(rbt, bat\_carregada)$.
5. $bel(rbt, \neg bat\_carregada)$ if $bel(rbt, sinal\_carga\_bat, T)$.
6. $act(rbt, guardar(O))$ $causes$ $bel(rbt, guardado(O))$.
7. $bel(rbt, entrada(O))$ if $bel(rbt, no\_entrada(O), T)$.
8. $act(rbt, guardar(O))$ $causes$ $bel(rbt, \neg entrada(O))$.

O robô possui dois desejos:

- que a bateria esteja sempre carregada.
- que todo o objeto que chega seja guardado.

As crenças descrevem como estes desejos podem ser satisfeitos, ou seja, suas pré e pós condições. Imaginando as seguintes situações:

a) Um pacote chegando no balcão

Um pacote p_1 chegando, o robô (agente) sensora que o pacote está lá (linha 7).

Essa crença $bel(rbt, no\_entrada(p_1), T)$ torna-se verdadeira quando o agente percebe ($sense$) que um pacote está no balcão.

Essa é a pré-condição $bel(rbt, entrada(p_1))$, que por sua vez é um gatilho sinalizado que ativa o desejo da linha 3.

Uma vez selecionado o desejo a ferramenta tenta construir um plano que o satisfaça.

Neste caso, o agente passa acreditar $bel(rbt, guardado(p_1))$ após ter executado a ação $guardar(p_1)$.

Outra conseqüência de executar essa ação é que o agente passa acreditar que o objeto p_1 não está mais na entrada (linha 8).

b) Chegando um pacote e percebendo que o sinal da bateria está baixo

Neste caso, as crenças sensoradas ativam os dois desejos, no entanto apenas um pode ser satisfeito. É neste momento que se verifica a propriedade de prioridade do desejo.

Assim, o agente tentará satisfazer o desejo da linha 2 executando a ação da linha 4. Uma vez satisfeito esse desejo, o outro passa a ser candidato e pode ser satisfeito, como na primeira situação.

6.3 Utilizando o X-BDI

Os trabalhos que utilizaram o X-BDI como ferramenta para programação de agentes BDI foram o MCOE [GIR99] e MENSAGEN [ROZ00]. O MCOE utilizou o X-BDI exatamente para definir o *kernel cognitivo* do agente tutor (a parte deliberativa do agente), que é, justamente, o que a ferramenta permite construir. É necessário salientar que existe a necessidade, vide figura 4, de se construir o entorno ao kernel cognitivo do agente, isto é, as outras funcionalidades necessárias para que o agente cumpra seu papel dentro do sistema. É necessário dotar o agente de sensores e atuadores conforme a definição de [RUS96].

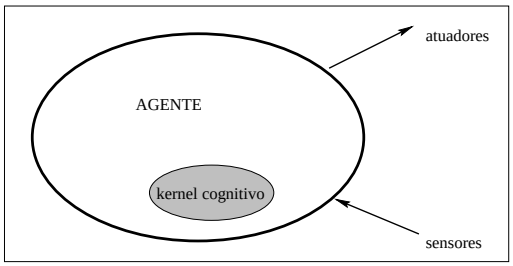

Figura 4: O kernel cognitivo e o X-BDI

A MENSAGEN propõe uma metodologia para modelagem dos estados mentais em agentes BDI. O trabalho utilizou o X-BDI por ser uma ferramenta apropriada para o processo de modelagem e programação do agente, permitindo, assim, que a metodologia pudesse ser validada em alguns aspectos.

7 Considerações Finais

O X-BDI é uma ferramenta em fase de consolidação, não sendo ainda um produto. É necessário que uma série de refinos e recursos sejam incorporados, afim de que sua utilização seja mais abrangente e acessível. Para usar o X-BDI não existe, neste momento, uma interface mais amigável e uma documentação organizada na forma de um manual que auxilie o projetista a utilizar a ferramenta.

Um agente modelado/implementado com o X-BDI requer um entorno (interface) para permitir a inserção de entradas (crenças, desejos e outras informações relevantes) e a visualização da saída (resultado do processo de deliberação), que são os planos a serem executados. No trabalho de [GIR99], pode-se observar isto claramente, quando é proposta uma arquitetura multiagente que trata a questão da seleção de múltiplas estratégias de ensino, levando em consideração a modelagem cognitiva de cada aluno que interage com o sistema.

Dada a importância da ferramenta X-BDI no contexto da pesquisa em agentes modelados com BDI e a continuidade do trabalho desenvolvido por [GIR99] e [MÓR00] é que surgiu a idéia de se construir um editor para programação orientada a agentes BDI associado à ferramenta X-BDI. A proposta deste editor, recentemente, foi apresentada como Plano de Estudo e Pesquisa [ZAM00b] no Programa de Pós-Graduação em Ciência da Computação desta universidade, bem como no XI Simpósio Brasileiro de Informática na Educação [ZAM00] e no II Workshop em Informática na Educação [ZAM00a].

O Editor de programação orientado a agentes BDI permitirá a visualização, organização e consistência das informações (estados mentais - desejos e crenças) a serem processadas no *kernel cognitivo*. Facilitará a descrição declarativa dos estados mentais dos agentes que serão necessários para a especificação das ações dos agentes e para a coreografia[9] existente. Além disso, o editor incorporará funções de auxílio para o usuário.

A necessidade de facilitarmos a edição dos estados mentais associados aos agentes reside na natureza da descrição dos estados mentais. A descrição dos agentes com estados mentais segue um paradigma declarativo, ou seja, ao invés de se descrever passo à passo, proceduralmente, o comportamento do agente, descreve-se os seus desejos e crenças, sendo seu comportamento gerado a partir desta descrição. Se por um lado descrições declarativas tendem a ter um nível de abstração mais alto, facilitando a modelagem, por outro, à medida que as descrições crescem, visualizar e depurar o modelo tende ser razoavelmente complexo.

Uma das grandes dificuldades dos projetistas que utilizam a abordagem BDI para modelagem dos agentes cognitivos é, justamente, a organização dos estados mentais e sua inter-relação. No caso da ferramenta X-BDI, o projetista necessita apenas descrever o conjunto

[9] O termo coreografia refere-se à dinâmica envolvendo os estados mentais, que ocorre durante a interação entre os agentes. Por exemplo, a mudança na base de crenças (revisão, remoção, etc.), a adoção de um desejo candidato à intenção e a confirmação ou não de uma expectativa. Estes exemplos representam alguns dos eventos que podem acontecer ao longo de uma interação entre agentes modelados nesta abordagem.

de crenças e desejos e as intenções são inferidas a partir das informações fornecidas à ferramenta (maiores detalhes vide [GIR99]; [MÓR00]; [ZAM00c]). Porém, esta descrição necessita da organização de um conjunto de crenças esteja associado a determinados desejos e, como isto está ligado com os planos e, conseguinte, que ações devem ser executadas pelo agente. Esta organização não é trivial e pode ser muito facilitada se o projetista possuir um editor onde ele escreva os estados mentais (utilizando a linguagem do X-BDI) e possa visualizar as inter-relações através da representação gráfica gerada (grafos ou árvores).

Assim, o objetivo deste trabalho foi compreender o X-BDI como ferramenta de programação e gerar um material teórico-prático que sirva de apoio para a construção do editor proposto, bem como uma documentação inicial para a utilização do X-BDI. Um fator a salientar é que poucos grupos de pesquisa, proporcionalmente, estão trabalhando com esta abordagem. Fato este que restringe o referencial teórico. Aliado a isso, a escassez de ferramentas para programação de agentes BDI, restringe mais o escopo da pesquisa.

No caso deste trabalho a restrição é intencional, pois escolhemos a ferramenta X-BDI como base para nosso trabalho, ficando o referencial teórico associado ao trabalho do seu criador Móra [MÓR00] e aos trabalhos que o utilizaram [GIR99] e [ROZ00].

Como todo o trabalho que cumpre seu ciclo, este termina ainda com algumas questões a serem tratadas como trabalhos futuros. Na seqüência do trabalho, será elaborado um "manual do usuário" do X-BDI, onde pretende-se descrever os passos, que vão desde a instalação da ferramenta e seus requisitos até a modelagem passo-a-passo de um agente, ou agentes.

Acredita-se que dada a importância da proposta do X-BDI para comunidade de IA, o trabalho realizado neste relatório técnico contribui para o esclarecimento de dúvidas daqueles que estão iniciando seus estudos nessa abordagem.

Referências

[ALF96] ALFERES, J.J.; PEREIRA, L.M. **Reasoning with logic programming**. Berlin, Heidelberg, New York, London, Paris, Tokyo, Hong Kong, Barcelona, Budapest: Springer-Verlag, 1996.

[BRA87] BRATMAN, M.; ISRAEL, D.; POLLACK, M. **Toward an architecture for resource-bounded agents**. Stanford: Stanford University, 1987.

[BRA84] BRATMAN, M. Two faces of intention. **The Philosophical Review, v.93, n.3**, p.275–405, 1984.

[BRA89] BRATMAN, M. What is intention? In: COHEN, P; MORGAN, J; POLLACK, M. (Eds.), 1989. **Anais...** MIT Press, 1989.

[COR94] CORRÊA, M. **A arquitetura de diálogos entre agentes cognitivos distribuídos**. Rio de Janeiro: UFRJ, 1994. Tese de Doutorado.

[DEN87] DENNETT, D. **The intentional stance**. Cambridge, MA: MIT Press, 1987.

[d'IN98] d'INVERNO, M.; KINNY, D.; LUCK, M.; WOOLDRIDGE, M. A formal specification of dMARS. In: INTELLIGENT AGENTS IV: 4th INTERNATIONAL WORKSHOP ON AGENT THEORIES, ARCHITECTURES AND LANGUAGES, 1998. **Proceedings...** Springer-Verlag, 1998.

[GEO89] GEORGEFF, M.; IGRAND, F. Decision-making in an embedded reasoning system. In: 11th INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE, 1989, Detroit, Michigan. **Proceedings...** Ames Research Center, 1989.

[GEO86] GEORGEFF, M.; LANSKY, A. Procedural knowledge. In: SPECIAL ISSUE ON KNOWLEDGE REPRESENTATION, 1986. **Proceedings...** IEEE, 1986. v.74, p.1383–1398.

[GIR99] GIRAFFA, L. M. M. **Uma arquitetura de tutor utilizando estados mentais**. Porto Alegre: CPGCC/UFRGS, 1999. Tese de Doutorado.

[JAV01] JAVASUN. **Custom networking**: all about sockets. Disponível por WWW em http://java.sun.com/docs/books/tutorial/networking/sockets/definition.html (2001).

[MÓR98] MÓRA, M.; LOPES, J.G.; COELHO, J.G.; VICCARI, R. BDI models and systems: reducing the gap. In: AGENTS THEORY, ARCHITECTURE AND LANGUAGES WORKSHOP, 1998, Canarias. **Proceedings...** Springer-Verlag, 1998.

[MÓR00] MÓRA, M. **Um modelo de agente executável**. Porto Alegre: CPGCC/UFRGS, 2000. Tese de Doutorado.

[OLI96] OLIVEIRA, F. Inteligência artificial distribuída. In: IV ESCOLA REGIONAL DE INFORMÁTICA, 3, 1996, Canoas, RS. **Anais...** Sociedade Brasileira de Computação, 1996.

[QUA97] QUARESMA, P. **Inferência de atitudes em diálogos**. Lisboa, Portugal: Faculdade de Ciências e Tecnologia, 1997. Tese de Doutorado.

[RAO92] RAO, A.S.; GEORGEFF, M.P. An abstract architecture for rational agents. In: INTERNATIONAL CONFERENCE ON PRINCIPLES OF KNOWLEDGE REPRESENTATION AND REASONING - KR, 3., 1992. **Proceedings...** Morgan Kaufman, 1992.

[RAO96] RAO, A.S. AgentSpeak(L): BDI agents speak out in logical computable language. In: 7th EUROPEAN WORKSHOP ON MODELING AUTONOMOUS AGENTS IN A MULTI-AGENT WORLD, MAAMAW'96, 1996, Eindhoven, The Netherlands. **Proceedings...** Springer, 1996. p.42–55.

[ROZ00] ROZA, M. P. **MENSAGEN**: uma metodologia para modelagem de estados mentais em agentes. Porto Alegre: PPGCC/PUCRS, 2000. Dissertação de Mestrado.

[RUS96] RUSSEL, S.; NORVIG, P. **Artificial intelligence**: a modern approach. New Jersey: Prentice Hall, 1996.

[ZAM00] ZAMBERLAM, A.O.; GIRAFFA, L.M.M.; MÓRA, M. C. Um editor para programação orientada a agentes BDI. In: XI SIMPOSIO BRASILEIRO DE INFORMATICA NA EDUCACAO, 2000, Maceio, Alagoas. **Anais...** SBC, 2000.

[ZAM00a] ZAMBERLAM, A.O.; GIRAFFA, L.M.M.; MÓRA, M. C. Agentes cognitivos modelados com BDI. In: I WORKSHOP EM INFORMATICA NA EDUCACAO, 2000, Passo Fundo, RS. **Anais...** Instituto de Educação/UPF, 2000.

[ZAM00b] ZAMBERLAM, A.O. **Um editor para programação orientada a agentes BDI**. Porto Alegre: PPGCC/PUCRS, 2000. Plano de Estudo e Pesquisa.

[ZAM00c] ZAMBERLAM, A.O. **Modelagem de agentes utilizando a arquitetura BDI**. Porto Alegre: PPGCC/PUCRS, 2000. Trabalho Individual I.