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

Um Modelo Formal e Executável de Agentes BDI

por

Michael da Costa Móra

Tese submetida à avaliação,
como requisito parcial para a obtenção do grau de Doutor
em Ciência da Computação

Prof. Dra. Rosa Maria Viccari
Prof. Dr. José Gabriel Pereira Lopes
Orientadores

Porto Alegre, Setembro de 1999.

CIP - CATALOGAÇÃO NA PUBLICAÇÃO

Móra, Michael da Costa

Um Modelo Formal e Executável de Agentes BDI / por Michael da Costa Móra. — Porto Alegre: PPGC da UFRGS, 1999.
185 f.: il.

Tese (doutorado) — Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 1999. Orientadores: Viccari, Dra. Rosa Maria; Lopes, Dr. José Gabriel Pereira.

1. Arquiteturas BDI. 2. Agentes Inteligentes. 3. Sistemas Multiagentes. 4. Programação em Lógica. I. Viccari, Dra. Rosa Maria II. Lopes, Dr. José Gabriel Pereira III. Título.

Conteúdo

Lista de Figuras

Lista de Tabelas

Resumo

Modelos BDI (ou seja, modelos **B**eliefs-**D**esires-**I**ntentions models) de agentes têm sido utilizados já há algum tempo. O objetivo destes modelos é permitir a caracterização de agentes utilizando noções antropomórficas, tais como estados mentais e ações. Usualmente, estas noções e suas propriedades são formalmente definidas utilizandos formalismos lógicos que permitem aos teóricos analisar, especificar e verificar agentes racionais. No entanto, apesar de diversos sistemas já terem sido desenvolvidos baseados nestes modelos, é geralmente aceito que existe uma distância significativa entre esta lógicas BDI poderosas e sistemas reais. Este trabalho defende que a principal razão para a existência desta distância é que os formalismos lógicos utilizados para definir os modelos de agentes não possuem uma semântica operacional que os suporte. Por "semântica operacional" entende-se tanto procedimentos de prova que sejam corretos e completos em relação à semântica da lógica, bem como mecanismos que realizem os diferentes tipos de raciocínio necessários para se modelar agentes.

Há, pelo menos, duas abordagens que podem ser utilizadas para superar esta limitação dos modelos BDI. Uma é estender as lógicas BDI existentes com a semântica operacional apropriada de maneira que as teorias de agentes se tornem computacionais. Isto pode ser alcançado através da definição daqueles procedimentos de prova para as lógicas usadas na definição dos estados mentais. A outra abordagem é definir os modelos BDI utilizando formalismos lógicos apropriados que sejam, ao mesmo tempo, suficientemente poderosos para representar estados mentais e que possuam procedimentos operacionais que permitam a utilizaçao da lógica como um formalismo para representação do conhecimento, ao se construir os agentes. Esta é a abordagem seguida neste trabalho.

Assim, o propósito deste trabalho é apresentar um modelo BDI que, além de ser um modelo formal de agente, seja também adequado para ser utilizado para implementar agentes. Ao invés de definir um novo formalismo lógico, ou de estender um formalismo existente com uma semântica operacional, define-se as noções de crenças, desejos e intenções utilizando um formalismo lógico que seja, ao mesmo tempo, formalmente bem-definido e computacional. O formalismo escolhido é a *Programação em Lógica Estendida com Negação Explícita* (**ELP**) com a semântica dada pela **WFSX** (*Well-Founded Semantics with Explicit Negation - Semântica Bem-Fundada com Negação Explícita*). **ELP** com a **WFSX** (referida apenas por **ELP** daqui para frente) estende programas em lógica ditos normais com uma segunda negação, a *negação explícita*[1]. Esta extensão permite que informação negativa seja explicitamente representada (como uma crença que uma propriedade P não se verifica, que uma intenção I não deva se verificar) e aumenta a expressividade da linguagem. No entanto, quando se introduz informação

[1]Em contraste com *negação por falha* usual de programas em lógica normais, que são chamadas *Negação Implícita* no contexto da ELP.

negativa, pode ser necessário ter que se lidar com programas contraditórios. A ELP, além de fornecer os procedimentos de prova necessários para as teorias expressas na sua linguagem, também fornece um mecanismo para determinar como alterar minimamente o programa em lógica de forma a remover as possíveis contradições. O modelo aqui proposto se beneficia destas características fornecidas pelo formalismo lógico.

Como é usual neste tipo de contexto, este trabalho foca na definição formal dos estados mentais em como o agente se comporta, dados tais estados mentais. Mas, constrastando com as abordagens até hoje utilizadas, o modelo apresentanto não é apenas uma especificação de agente, mas pode tanto ser executado de forma a verificar o comportamento de um agente real, como ser utilizado como mecanismo de raciocínio pelo agente durante sua execução. Para construir este modelo, parte-se da análise tradicional realizada na psicologia de senso comum, onde além de crenças e desejos, intenções também é considerada como um estado mental fundamental. Assim, inicialmente define-se estes três estados mentais e as relações estáticas entre eles, notadamente restrições sobre a consistência entre estes estados mentais. Em seguida, parte-se para a definição de aspectos dinâmicos dos estados mentais, especificamente como um agente escolhe estas intenções, e quando e como ele revisa estas intenções. Em resumo, o modelo resultante possui duas características fundamentais:(1) ele pode ser usado como um ambiente para a especificação de agentes, onde é possível definir formalmente agentes utilizando estados mentais, definir formalmente propriedades para os agentes e verificar se estas propriedades são satifeitas pelos agentes; e (2) também como ambientes para implementar agentes.

Abstract

Beliefs-**D**esires-**I**ntentions models (or BDI models) of agents have been around for quite a long time. The purpose of these models is to characterize agents using anthropomorphic notions, such as mental states and actions. Usually, these notions and their properties are formally defined using logical frameworks that allow theorists to analyze, to specify and to verify rational agents. However, despite the fact that many systems have been developed based on these models, it is a general concern that there is a gap between those powerful BDI logics and practical systems. We believe that the main reason for the existence of this gap is that the logical formalisms used to define the models do not have an operational model that support them. By an operational model we mean proof procedures that are correct and complete with respect to the logical semantics, as well as mechanisms to perform different types of reasoning needed to model agents.

There are, at least, two major approaches that may be used to overcome this limitation of BDI models. One is to extend existing BDI logics with appropriate operational models so that agent theories become computational. This may be achieved through the definition of proof procedures of the logic used to model the mental states. The other approach is to define BDI models using a suitable logical formalism that is both powerful enough to represent mental states and that has operational procedures that allow us to use the logic as a knowledge representation formalism, when building the agent. This is also the approach we follow in this work.

The purpose of this work is to present a BDI model that, besides being a formal model of agents, is also suitable to be used to implement agents. Instead of defining a new BDI logic or choosing an existing one, and extending it with an operational model, we define the notions of belief, desires and intentions using a logical formalism that is both well-defined and computational. The formalism we are using is *logic programming extended with explicit negation* (ELP) with the *Well-Founded Semantics eXtended for explicit negation* (WFSX). ELP with WFSX (simply ELP, from now on) extends normal logic programs with a second negation named *explicit*[2] *negation*. This extension allows us to explicitly represent negative information (like a belief that a property P does not hold, or an intention that a property Q should not hold) and increases the expressive power of the language. When we introduce negative information, we may have to deal with contradictory programs. The ELP framework, besides providing the computational proof procedure for theories expressed in its language, also provides a mechanism to determine how to minimally change a logic program in order to

[2]In contrast with the usual *negation as failure* or *negation by default* of normal logic programs, which is called *implicit negation* in the ELP context.

remove contradictions. Our model benefits from these features provided by the logical formalism.

As it is usually done, we focus on the formal definition of mental states and on how the agent behaves, given such mental states. But, contrasting with these former approaches, our model is not only an agent specification, but it may also be executed in order to verify the actual agent behavior, as well as it may be used as reasoning mechanism by actual agents. We depart from the traditional folk phycology analysis, where along with desires and beliefs, intentions is considered a fundamental mental state. Therefore, initially we define these three mental states and the static relations between them, namely constraints on consistency among those mental states. Afterwards, we advance with the definition of dynamic aspects of mental states, namely how the agent chooses its intentions, and when and how it revises its intentions. In fact, the resulting model has two defining characteristics: (1) it may be seen as a agent specification environment, where it is possible to formally define an agent using mental states, to formally define properties that the agents should comply with, and to verify if these properties are really present; and (2) also as an agent implementation environment.

1 Introdução

1.1 O Contexto da Tese

Nos últimos anos tem havido um grande interesse no desenvolvimento de sistemas que encorporem a noção de *agente* e em técnicas de desenvolvimento que adotem uma *abordagem orientada à agentes.* No entanto, apesar de tamanho interesse, a definição do que seja um agente é um ponto sobre o qual não há concenso. Isto acontece pois as definições de agente frequentemente são fortemente influenciadas pelo domínio onde são aplicados e pelo conjunto de exemplos de agente que quem produz a definição tem em mente[1]. Por outro lado, apesar da falta de uma definição concensual, há certas propriedades que se considera que um sistema, seja "software" ou "hardware", deve exibir a fim de que seja considerado um agente. Estas propriedades são [WOO 95]:

- *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;
- *pro-atividade* – a capacidade de, além de responder a estímulos do ambiente, ser capaz de exibir um comportmento orientado a objetivos. Ou seja, ser capaz 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.

Estas definições e propriedades que caracterizam a noção de agente tem por objetivo não meramente dividir o mundo entre entidades que são e que não são

[1]O leitor interessado em uma discussão sobre as várias definições de agentes pode referir-se à [FRA 97], onde os autores fazem uma coletânea de várias destas definições, bem como uma breve comparação entre as mesmas. Pode consultar ainda [SIC 92][V.D 95] [CAS 97][WOO 97].

agentes, mas servir de ferramenta para analizar sistemas [RUS 95], bem como especificar, projetar e construir sistemas cujos elementos básicos sejam agentes. A medida que as aplicações sendo 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, estas entidades autônomas com capacidade de planejar suas ações, reagir e interagir entre sí em busca de soluções para problemas parecem fornecer um passo em direção a este nível de abstração mais alto.

Mas, assim como acontece com a modelagem dos sistemas, também a modelagem dos agentes requer níveis de abstração adequados. As abordagens utilizadas tem sido bastante diversas. Uma destas abordagens frequentemente utilizadas, e que é adotada neste trabalho, é a que considera agentes como sendo *sistemas intencionais*. 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 os mecanismos de resolução de problemas dos agentes em termos do que ele acredita, de que planos ele possui ou constrói a fim de satisfazer seus desejos e intenções, que atributos ele utiliza para determinar que opções ele escolhe, e assim por diante.

O trabalho desenvolvido nesta tese insere-se neste contexto, da modelagem de agentes como sistemas intencionais. As seções que se seguem apresentam uma visão geral sobre a noção de sistemas intencionais (seção 1.1.1) e de estados mentais (seção 1.1.2), a distinção entre modelos formais e arquiteturas de agentes (seção 1.1.3) e, finalmente, a motivação, objetivos e abordagem utilizada neste trabalho (seção 1.2).

1.1.1 Sistemas Intencionais

Quando se descreve o comportamento humano, é comum utilizar termos como "acreditar", "querer", "precisar" e outros deste tipo. Tais termos são oriundos do entendimento que as pessoas tem de como explicar as propriedades observáveis da mente[2]. Dennet [DEN 87] refere-se a esta prática como *explicação intencional*, como nos exemplos abaixo:

- "Ele abriu a porta pois *pensava* (ou *acreditava*) que havia alguém a espera.";
- "João foi de avião pois *pretendia* (ou *tinha a intenção de*) chegar a São Paulo ainda hoje".

Na tentativa de fundamentar este processo de explicação intencional, Searle [SEA 84] desenvolveu uma *teoria da intencionalidade*, onde ele define intencionalidade como sendo aquela propriedade de vários estados mentais e eventos através da qual eles são dirigidos à ou sobre objetos e estados do mundo. Assim,

[2]Em inglês, este *entendimento comum* é chamado de *folk psychology*.

se alguém possui uma crença, deve ser uma crença de que algo é de determinada maneira; se alguém possui um desejo, deve ser um desejo de fazer algo ou de que algo seja de determinada maneira. É a esta direcionalidade que alguns estados mentais possuem (por exemplo, medo ou ansiedade seriam estados mentais mas não seriam intencionais) que se chama intencionalidade. Ainda de acordo com Searle, esta abordagem intencional segue uma longa tradição filosófica e é, portanto, bastante usual e aceita quando utilizada em relação a seres humanos. No entanto, poder-se-ia questionar a validade da atribuição de tais conceitos a máquinas e sistemas outros que não os humanos.

A crítica mais comum é que termos intencionais seriam apenas metáforas vazias e que seu uso generalizado, sem cuidados que estabeleçam uma relação entre o conceito e seu funcionamento, levam fatalmente ao inssucesso das técnicas baseadas em estados intencionais [SMI 91]. No entanto, MacCarthy [MCC 78] argumenta que, mais do que simples metáforas, atribuir estados mentais a sistemas complexos é legítimo e útil: legítimo quando se pode encontrar nos sistemas sendo descritos um correspondente funcional aos estados mentais nos seres humanos, e é útil quando ajuda a entender e a controlar a estrutura do sistema, auxiliando ainda na compreensão do seu comportamento passado e futuro. Dennet [DEN 87] afirma ainda que, apesar das explicações intencionais terem as ações das pessoas como seu domínio primordial, há situações em estas explicações intencionais, bem como as previsões feitas a partir delas, são não somente úteis mas indispensáveis para dar conta do comportamento de máquinas complexas. Ele acrescenta, ainda, que o que importa ao adotar esta abordagem não é se os sistemas são realmente intecionais, i.e., realmente formados por estados mentais, mas se eles podem ser coerentemente descritos como tal.

Apesar de amplamente utilizado e, como dito anteriormente, de sua longa tradição, esta abordagem intencional (mesmo quando adotada com relação a seres humanos) recebeu críticas de vários tipos. A maior parte delas, oriundas de pesquisadores de áreas ligadas a neurociência, ataca o princípio não-científico da concepção de noções como "acreditar", "pretender" e assim por diante. O argumento básico é que, eventualmente, tais teorias psicológicas intencionais seriam completamente substituídas por uma teoria *completa* da neurociência. De acordo com Churchland [CHU 81] e sua *abordagem materialista*, uma compreensão neuro-fisiológica do ser humano eventualmente eliminará a concepção mentalística que temos de nós mesmos. Ele observa, ainda, que de acordo com o que se depreende da evolução das ciências, é muito provável que os processo psicológicos sejam, na verdade, processos do cérebro, físicos, e não processos de uma mente ou alma não físicas.

Ainda assim, apesar destas críticas serem bem fundamentadas, discute-se se as teorias intencionais devem ser completamente abandonadas. De fato, segundo Haddadi [HAD 96], ao se examinar detalhadamente a teoria de Dennet, é difícil detectar contradições com as teorias materialistas, uma vez que o foco da mesma está na explicação e previsão de comportamentos, e não no argumento que, de fato, estados mentais sejam estruturas componentes dos sistemas. No que diz respeito a sistemas computacionais, ambos os domínios — intencional e materialista, baseado nas neurociências — podem ainda atingir seus objetivos, modelando os sistemas a partir de princípios distintos e não tendo em consideração a outra abordagem. O que parece inevitável é que, progressivamente, ambas as abordagens comecarão a se entrelaçar.

Assim sendo, este trabalho adota uma abordagem intencional, uma vez que:

- os estados mentais, como abstração, tem um apelo conceitual bastante forte. São, por conseguinte, bastante naturais para os projetistas e os analistas dos agentes;
- fornecem descrições sucintas de sistemas complexos, além de ajudar a entender e a explicar o comportamento destes sistemas;
- podem ser usados pelos próprios agentes para raciocinar sobre eles próprios e sobre outros agentes (reciprocidade, segundo a nomenclatura adotada por Dennet [DEN 87]).

É importante observar que, apesar da abordagem puramente intencional adotada no desenvolvimento deste trabalho, e como será discutido no capítulo, a integração das duas abordagens na forma de sistemas híbridos é considerada desejável. O desenho e o desenvolvimento de sistemas de alta complexidade exigirão ferramentas e métodos que levem em conta ambas as teorias.

1.1.2 Os Estados Mentais

Uma vez adotada esta abordagem intencional, a questão passa a ser que estados mentais devem ser utilizados para descrever o agente. Os diferentes estados mentais desempenham papeis diferentes no comportamento dos agentes. Searle [SEA 84], quando define o que é intencionalidade, classifica os estados mentais em duas categorias: *estados mentais de informação* e *estados mentais pró-ativos*, como pode ser visto na figura 1.1. Esta classificação está diretamente relacionada à noção de direção inerente ao conceito de intencionalidade, como descrito na seção 1.1.1, e vai definir o tipo de papel que o estado mental vai desempenhar. Estados mentais de informação são aqueles que representam as informações sobre o mundo onde o agente está inserido, ou seja, são aqueles estados mentais que tendem a ser modificados a fim de que seu conteúdo reflita o estado do mundo, como crenças e conhecimento. Estados mentais pro-ativos são aqueles que, de alguma forma, conduzem a ação do agente, ou seja, estão diretamente ligados ao processo de seleção, por parte do agente, de um entre vários cursos de ação possíveis. No caso dos estados mentais pro-ativos, o agente tende a modificar o mundo de forma que o mundo se adeque ao conteúdo dos estados mentais. São eles estados mentais como desejos, intenções, preferências, obrigações, e assim por diante.

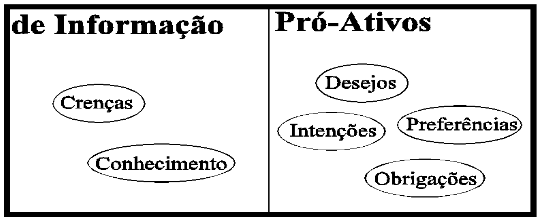

Figura 1.1 — Categorias de Estados Mentais - Searle

Esta não é a única classificação que se pode encontrar para os estados mentais. Kiss [KIS 92] classifica-os em três grupos: *cognitivos*, *conativos* e *afetivos*, conforme a figura 1.2. Os estados cognitivos referem-se aqueles ligados a questões epistêmicas, como os estados de informação do Searle. Os estados conativos referem-se à ação, ao controle e às tentativas de realização de ações. Já os estados afetivos, segundo Kiss, referem-se àquelas atitudes que estão ligadas à dinâmica do comportamento do agente, como objetivos e preferências. Apesar do nome, os estados mentais afetivos não tem nenhuma relação com emoções como medo, ansiedade, entre outros (que não são atitudes intencionais, como foi referido anteriormente na seção 1.1.1). O que distingue os conativos dos afetivos na classificação de Kiss é que os conativos de fato geram ações (como o comprometimento), enquanto que os afetivos tem o potencial de levar o agente a agir, mas não necessariamente o fazem. A união dos estados conativos e afetivos reduz esta classificação àquela proposta por Searle.

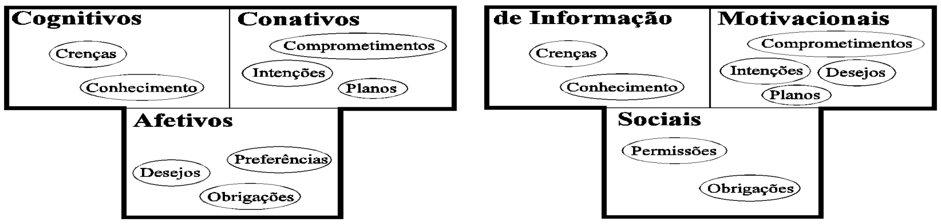

Figura 1.2 — Categorias dos estados mentais - Kiss e Shoham & Cousin

Shoham e Cousins [SHO 94] adotam como critério de classificação a relevância em aplicações computacionais das atitudes, e dividem-nas em atitudes *de informação*, *motivacionais* e *sociais*, como na figura 1.2. A classificação é bastante semelhante a de Searle, a não ser pelo fato de que eles distinguem aqueles estados motivacionais que tem papel social, como obrigações e permissões.

Como foi dito anteriormente, a questão que se põe é que combinação de estados mentais é adequada para descrever o agente. A classificação de Searle, baseada na direcionalidade do estado mental, pode servir de orientação nesta seleção. Na medida em que os estados mentais, de acordo com Searle, ou refletem o mundo, ou fazem o agente modificar o mundo, parece razoavel exigir que, pelo menos, uma atitude de cada tipo seja selecionada. Seguindo esta linha, Davidson [DAV 80] argumenta que com apenas dois estados mentais, *desejos* e *crenças*, é possivel descrever o comportamento de agentes inteligentes e que, se por um lado um número maior de estados mentais talvez fosse interessante para facilitar tal descrição, por outro todos os demais estados mentais intencionais podem ser reduzidos a estes dois últimos, o que torna os demais supérfluos na produção de explicações intencionais.

Segundo este modelo proposto por Davidson, o evento a seguir:

- "Pedro apanhou o avião das 7:30 para São Paulo"

  teria a seguinte explicação intencional:

- "Pedro *desejava* estar em São Paulo as 11:30, e apanhou o vôo das 7:30 pois *acreditava* que este vôo chegava ao destino antes daquela hora".

Ou seja, o que levou Pedro à ação foi este desejo de estar em São Paulo, fundamentado por sua crença no tempo de duração do vôo. A noção de desejo de Davidson tem um sentido mais amplo do que aquele empregado no senso comum. Ele argumenta que um agente somente age se o desejar. Mesmo quando este agente faz algo contra sua vontade, que não deseje legitimamente (mas por ser moralmente obrigado, ou coagido para tal, por exemplo), ele só passa à ação se o aceitar (ou, como ele diz, se passar a desejá-lo). De acordo com Davidson, este desejo poderia ser sub-categorizado como uma necessidade, uma vontade, uma obrigação ou outro estado mental pró-ativo que, de uma forma ou de outra, acabaria reduzido a um desejo.

O problema com este modelo baseado somente em desejos e crenças é que ele não contempla os processo de raciocínio e decisão envolvidos quando da ação de um agente. Bratman [BRA 87] argumenta que agentes, sejam humanos ou artificiais, são entidades de recursos limitados (memória, capacidade de raciocínio, e outros). Assim, eles não seriam capazes de, continuamente, avaliar seus desejos e crenças a fim de agir racionalmente. Após um certo tempo de raciocínio, os agentes devem decidir, *escolher* um subconjunto dos seus desejos e *se compremeter* em tentar realizá-los. Seriam estes desejos escolhidos, as *intenções* do agente, que lavariam o agente a ação. Assim, Bratman, ao contrário de Davidson e outros autores, afirma que intenções não são redutíveis a desejos e crenças, e que tem um papel distinto nos raciocínio dos agentes racionais. Enquanto que desejos (e os demais estados pro-ativos, como preferências, necessidades e assim por diante) são *potenciais influenciadores da conduta* dos agentes, intenções são *geradores e controladores de conduta* [BRA 87] [BRA 90].

Seguindo este argumento, Bratman identifica alguns papéis funcionais das intenções, que mostram como elas dão origem a ações e limitam a necessidade de raciocínio:

- intenções impõem problemas para o agente, pois este precisará determinar meios para satisfazer tais intenções. De acordo com Bratman, uma vez adotada uma intenção (que é um desejo que o agente escolhe e se compromete a satisfazer), um agente racional raciocinará a partir destas intenções e das crenças relevantes para derivar novas intenções que servirão como meio, passos intermediários ou cursos de ação específicos que levem a satisfação daquela intenção;

- uma vez que uma intenção leva o agente a realizar ações, elas devem ser mutualmente consistentes. Ou seja, um agente não deve se comprometer com opções que sejam conflitantes com intenções já existentes. Assim, intenções fornecem um *filtro de admissibilidade* [BRA 90], ou seja, elas restringem futuras intenções e limitam o número de opções a serem consideradas durante o raciocínio. Bratman afirma que, por estas características, as intenções, tanto aquelas adotadas a partir dos desejos como aquelas adotadas como refinamento de intenções anteriores, precisam possuir uma certa estabilidade. As intenções, como regra geral, resistem a serem reconsideradas ou abandonadas, embora não devam ser consideradas como irrevogáveis, já que isto também daria origem a comportamentos irracionais. Estes dois são papéis desempenhados pelo que Bratman denomina *intenções orientadas ao futuro*[3], ou seja, intenção de atingir um estado particular do mundo no futuro;

- uma vez adotada uma intenção, o agente deve monitorar o sucesso ou falha de suas intenções, de modo a persistir na execução ou a re-planejar seu curso de ação. Este é o papel desempenhado pelas intenções orientadas ao presente[4], ou seja, intenção de executar uma ação.

Assim, segundo a análise de Bratman, são três os estados mentais básicos necessários para modelar agentes seguindo uma abordagem intencional, bem como descrever e prever seu comportamento: Crenças, Desejos e Intenções (respectivamente **B***eliefs*, **D***esires* e **I***ntentions*, em inglês). As crenças formam o estado de informação do agente, representando as informações que o agente tem sobre o mundo e sobre sí próprios. Ou seja, elas constituem a representação da situação corrente e passada do mundo conforme percebido pelo agente. Os desejos estão relacionados com os estados do mundo que o agente eventualmente quererá atingir. No entanto, os desejos não necessariamente levam o agente a agir, ou seja, o fato de possuir um desejo não significa que o agente agirá para satisfazê-lo. Significa, ao invés, que antes que agente decida o que fazer, ela deliberará, confrontando seus vários desejos (os estados que ele deseja ver atingidos) com suas crenças (a situação corrente do mundo como percebido pelo agente). O agente

[3]Do inglês *future-directed intentions.*

[4]Do inglês *present-oriented intentions.*

escolherá, então, de acordo com algum critério, um subconjuntos dos desejos que ele tentará satisfazer. Em outras palavras, os desejos são um conjunto de estados possíveis de onde o agente escolhe o que fazer. As intenções são caracterizados por uma escolha de um estado a ser atingido e por um certo grau de comprometimento a esta escolha. Desta forma, as intenções são vistas como um compromisso que o agente assume com um determinado futuro possível. Isto significa que, ao contrário dos desejos, um intenção não pode ser contraditória com outras intenções, visto que não seria racional para um agente agir conscientemente com o objetivo de atingir estados incompatíveis. Uma vez que uma intenção é adotada, o agente tenta satisfazer esta intenção, planejando ações, re-planejando quando uma falha ocorre, e assim por diante. Tais ações, que são meios para satisfazer intenções, também são adotadas como intenções pelo agente. As três primeiras letras dos nomes dos estados em inglês — **BDI** — dão o nome a abordagem seguida em diversos trabalhos desenvolvidos sobre agentes que surgiram a partir do trabalho de Bratman, ou de variações deste ([COH 90] [RAO 91] [SHO 93] [WAI 94] [vL 96] [RAO 96] [HAD 96] [MÓR 98a] [BRA 88] [GEO 91] [COR 93] , entre outros). Alguns adotam exatamente os três estados mentais, outros acrescentam estados adicionais ou nomeiam os estados de maneira diferente, mas incorporam os conceitos básicos defendidos por Bratman. Também este trabalho parte da análise feita por Bratman e adota os três estados mentais — crenças, desejos e inteções — como elementos básicos utilizados para descrever os agentes e seus comportamentos. Antes de descrever como estes estados mentais são utilizados para este fim, é necessário fazer algumas colocações sobre modelos formais e arquiteturas de agentes.

1.1.3 Modelos Formais × Arquiteturas

Como foi referido anteriormente, a partir da análise feita por Bratman surgiram diversos trabalhos que tem por base os conceitos intencionais baseados na teoria BDI e por objetivo modelar sistemas que possuam aquelas propriedades — autonomia, habilidade social, reatividade, pro-atividade e adaptabilidade — que caracterizam o agente. Em outras palavras, estes trabalhos tem por objetivo descrever os elementos básicos do agente, sua funcionalidade e funcionamento. Nestes trabalhos, pode-se identificar duas linhas de atuação[5]:

- *modelos formais de agentes* – neste caso, modelar um agente inteligente significa *especificar*, utilizando alguma linguagem de especificação formal, o agente. Como é o usual na ciência da computação, o modelo é feito

[5]Esta divisão não por objetivo criar uma taxonomia dos trabalhos sobre agente. Na verdade, muitos trabalhos se sobrepõem em mais de uma das categorias, inclusive o que é desenvolvido nesta tese. No entanto, esta divisão é interessante pois permite situar o que é feito aqui neste trabalho.

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 [COH 90], e a abstração sendo considerada aqui são os estados ou atitudes mentais. A estes modelos de agente definidos com estas lógicas chama-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; a relação entre os diferentes estados mentais, como um estado mental influencia o outro, quais a restrições que uma atitude mental impõe a outra; como estes estados mentais interagem entre sí para produzir comportamentos dos agentes. Tais modelos tem dois papeis 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 disto, fornecer subsídios para a implementação de agentes.

No entanto, os modelos existentes falham em suprir totalmente estas necessidades. Por um lado, os modelos formais (como [COH 90] [RAO 91] [SIN 94] [KON 93] [HAD 96]) são bastante adequados para a descrição formal de agentes e suas propriedades, e também para serem usadas como ferramenta para verificar se agentes possuem determinadas propriedades. No entanto, estas são tão complexas e expressas em lógicas de nível de abstração tão altos que é difícil estabelecer sua relação com possíveis implementações. As arquiteturas de agentes (como [BRA 88] [GEO 91] [BUR 92] [COR 93] [BRA 88] [GEO 91]), ao contrário, estão muito próximas da implementação e fornecem subsídios bastante claros de como construí-la. No entanto, é bastante difícil estabelecer quais são os fundamentos teóricos que as justifiquem, ou de utilizar arquiteturas como ferramentas de análise de agentes. Mesmo quando há, alegadamente, um relação entre uma determinada arquitetura e um modelo teórico (como em [BRA 88] e [COH 90], ou em [GEO 91] e [RAO 91]), está relação é definida de maneira muito fraca e é difícil de ser verificada.

A existência desta distância entre modelos de agentes que permitam tanto a análise dos sistemas representados ao mesmo tempo que forneça subsídios para a implementação do mesmo é a principal motivação deste trabalho.

1.2 A Motivação e os Objetivos da Tese

O trabalho desenvolvido nesta tese tem uma característica dupla, uma vez que se trata de minorar o problema da distância entre modelos de agentes e sua implementação. Uma das razões principais para a existência desta distância entre especificação e implementação está na escolha do formalismo utilizado para construir tais modelos. Como foi referido anteriormente, a lógicas modais definidas para representar estados mentais tem sido o sistema formal mais utilizado para modelar agentes. No entanto, o que se nota é que há uma grande distância entre os modelos formais de agentes e as implementações de agentes que seguem tais modelos. Isto ocorre pois:

- as lógicas modais utilizadas não são, em geral, tratáveis computacionalmente. Tais lógicas, sejam elas lógicas multi-modais quantificadas (como em [COH 90] [RAO 91] [SIN 94]) ou lógicas modais não-normais (como em [KON 93]), não possuem procedimentos de derivação para manipulação de suas sentenças que sejam corretos e completos com relaç ão a semântica destas linguagens. Isto faz com que estas lógicas modais possam ser usadas como linguagem de especificação, mas não como ferrementa de representação do conhecimento ou ferramenta básica para implementar agentes. Além disto, 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 existentente 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 sistema implementam simplificações do modelo de agentes devido a impossibilidade de tratar computacionalmente os mecanismos inerentes as lógicas modais [CHE 80] [COS 92].

- além das dificuldades inerentes às lógicas escolhidas, os modelos adotam uma *perspectiva de especificação* [MÓR 95] [SIN 94]. 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.

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 dispõe de mecanismos que permitam aos agentes utilizá-la para representar e manipular a informação. A necessidade de se reduzir esta distância entre modelo teórico e implementação é a principal motivação deste trabalho.

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 comportamente autônomo e flexível como o descrito a seguir.

Example 1 *"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 vistante 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 obter quando se modela um agente inteligente: autonomia (o robô decide por sí 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 conduzí-los ao seu destino), reatividade (o robô locomove-se sem esbarrar em obstáculos, imediatamente recarrega a bateria ao detectá-la com pouca carga), pro-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). A dificuldade de construir, com os formalismos existentes, modelos de agentes que exibam este tipo de comportamento é a segunda motivação para este trabalho.

1.2.1 Os Objetivos da Tese

O objetivo deste trabalho é propor um modelo formal de agentes que, simultaneamente, apresente as vantagens dos modelos formais existentes, nomeadamente que possa ser usado como ferramenta de especificação e validação de agentes, e que ao mesmo tempo reduza a distância entre especificação e implementação de agentes. Além disto, a teoria proposta deve permitir descrever agentes e comportamentes autônomos e flexíveis, como aqueles mostrados no exemplo 1. Para tanto, dois requisitos tem que ser satisfeitos:

- uma vez que se deseja um modelo formal que possa ser usado para descrever agentes, o nível de abstração dos elementos básicos do modelo deve ser alto.

  Para satisfazer este requisito, a abordagem adotada no trabalho segue a abordagem intencional, ou seja, os agentes são descritos pelos seus estados mentais. Portanto, é necessário que o modelo formal defina quais são os estados mentais que compõem o agente, qual a relação entre estes estados mentais e como os comportamentos dos agentes são produzidos a partir destes estados mentais;

- visto que se deseja reduzir a distância entre a especificação e a implementação dos agentes, o modelo deve descrever os processos associados aos estados mentais e utilizados pelos agentes.

  A abordagem aqui utilizada consiste em adotar uma *perspectiva do agente* [MÓR 95] [SIN 94] na construção do modelo. Ou seja, ao invés de enxergar o modelo simplesmente como uma ferramenta de especificação 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.

1.2.2 A Abordagem Adotada

A construção de um modelo formal começa pela escolha ou definição de uma linguagem formal que será utilizada na descrição do modelo. Não basta, neste contexto, que a linguagem utilizada tenha a expressividade necessária para descrever os estados mentais. Esta linguagem deverá também ser utilizada pelo agente como ferramenta de representação e manipulação de conhecimento. Assim, outras características são também necessárias:

- a linguagem necessita de procedimentos de derivação que sejam corretos e completos com relação a sua semântica, a fim de que o agente possa manipular as sentenças da linguagem;

- a linguagem deve possibilitar a representação e a manipulação de informações contraditórias. Quando se lida com estados mentais pro-ativos, como desejos e intenções, é necessário, em algumas situações, representar simultaneamente informações contraditórias, em outras detectar e resolver contradições que surjam durante a manipulação dos estados mentais. Além disto, os procedimentos de manipulação devem ser computacionalmente tratáveis, de forma que o agente possa utilizá-los como ferramenta;

- a linguagem deve possuir mecanismos que permitam outras formas de raciocínio que não a dedução, notadamente abdução e algumas formas de raciocínio não-monotônico. Ao se fazer a seleção das intenções a partir dos desejos é necessário selecionar sub-conjuntos consistentes do conjunto de desejos do agente, o que é feito através de raciocínio revogável[6]. Também, a derivação de intenções secundárias a partir das intenções primárias é semelhante a um processo de planejamento, sendo feita por abdução.

Uma vez que estas características não são todas encontradas nas linguagens modais habitualmente utilizadas para construção de modelos de agentes, há pelo menos dois caminhos possíveis a seguir:

- selecionar uma das lógias modais habitualmente usadas e estendê-la com os mecanismos necessários. Esta é a via seguida por Rao em [RAO 96], onde ele define um precedimento de prova para a versão proposicional da sua lógica BDI [RAO 91]. Também Bell [BEL 95] desenvolve uma teoria computacional que é aplicada para formalizar a hierarquia entre objetivos de um agente [BEL 97];

[6]Do inglês *defeasible reasoning.*

- selecionar uma outra lógica ou linguagem formal que atenda a todas estas necessidades, e então construir o modelo de agentes nesta linguagem. Este é o caminho seguido por Corrêa et ali [COR 98], onde eles aplicam o modelo operacional para Teoria das Situações [DEV 91] definido por Nakashima [NAK 81] [COR 95] para tornar sua arquitetura de agentes SEM [COR 93], além de formal, executável. Este é também o caminho seguido neste trabalho [MÓR 98b] [MÓR 98a].

O formalismo utilizado para construir o modelo é a *programação em lógica estendida com negação explícita* com a semântica bem fundada com negação explícita (*WFSX* – *W*ell *F*ounded *S*emantics for e*X*tended programs), baseando-se no trabalho de Pereira et al. [ALF 96]. Esta semântica para programas em lógica estendida com negação explícita foi adotada devido essencialmente às suas propriedades (simplicidade, cumulatividade, racionalidade, relevância e avaliação parcial — ver o capítulo 2 para maiores detalhes) e devido a existência de procedimentos de prova corretos e completos com relação à *WFSX*, bem como de procedimentos que permitem a detecção e a manipulação de contradições e a implementação de diversas formas de raciocínio não monotônico.

O requisito básico para a lógica selecionada é que ela possa ser utilizada tanto para especificação como para implementação do sistema. E, de fato, esta é a maneira usual como a linguagem é vista na *programação em lógica*. Kowalski, em [KOW 86], afirma que o que distingue uma especificação de uma implementação em programação em lógica são questões de performance, visto que ambas são executáveis.

Uma vez selecionada a linguagem, passa-se a definição de quais os estados mentais básicos compõem o agente. O modelo desenvolvido será um modelo BDI, onde os três estados mentais que compõem o agente são crenças, desejos e intenções. Utilizando a linguagem escolhida, os estados mentais e suas propriedades são definidas.

Quando se lida com estados mentais pro-ativos e comportamentos do agente a partir destes estados mentais, é preciso representar e manipular tanto ações que o agente executa ao longo do tempo como propriedades que devem se verificar ao longo do tempo como consequência 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. Consequentemente, antes de partir para a definição dos estados mentais proprieamente ditos, é necessário construir, utilizando a programação em lógica extendida, um formalismo que nos permita representar ações e tempo, bem como raciocinar sobre eles.

O formalismo adotado neste trabalho é o Cálculo de Eventos (*Event Calculus*) [MES 92]. Inicialmente proposto por Kowalski e Sergot para superar algumas limitações do Cálculo de Situações, o cálculo de eventos possui como primitivas na sua ontologia os eventos, que são ocorrências de ações, as quais iniciam e terminam períodos durante os quais propriedades se verificam. Seguindo esta definição inicial, muitas variações do Cálculo de Eventos foram criadas para tratar diversos tipos de situação: planejamento por abdução e linearização de ações no planejamento [MES 92], ações concorrentes [MÓR 95] [QUA 95], ações não-instantâneas [QUA 97], entre outros. Neste trabalho, constroi-se mais uma variação do Cálculo de Eventos, a partir das versões de [MES 92] e [QUA 97], que permite lidar com ações instantaneas concorrentes, e com propriedades que ocorrem sem a intervenção do agente. Esta características permitem modelar eventos que ocorrem expontaneamente no ambiente, ou ações executadas por outros agentes e que não são percebidos pelo agente no momento de sua execução.

Uma vez construída a base formal necessária, passa-se a definição dos estados mentais. Inicialmente são definidos os aspectos estáticos do modelo: quais 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. É importante observar que este trabalho preocupa-se com o comportamento do agente a partir dos estados mentais. Consequentemente, a preocupação central é com os estados mentais pro-ativos que estão diretamente ligados a produção de ações. Portanto, questões como métodos de manutenção e revisão de crenças [GÄR 88] [DOY 79] não fazem parte do escopo deste trabalho, apesar das crenças serem definidas e dos processos relacionados aos estados mentais terem em linha de conta a necessidade de tratar as crenças.

Estes são os elementos que formam o modelo do agente. De fato, o modelo acaba por ter uma natureza dupla:

- serve como um ambiente de especificação de agentes, onde é possível definir formalmente um agente apenas em termos de seus estados mentais (crenças, desejos e intenções), definir formalmente propriedades que se deseja que o agente possua, e verificar se tais propriedades se verificam;

- serve como um ambiente de implementação de agentes. Embora questões como performance e eficiência não sejam consideradas, a partir das mesmas definições utilizadas para formalizar o agente é possível executá-los e verificar se o seu comportamento é o esperado.

É importante observar que um ambiente de implementação como o que resulta deste trabalho possui um alto nível de abstração (os estados mentais), o que reduz a complexidade no desenvolvimento de sistemas baseados em agentes.

1.3 Contribuições da Tese

Como visto na seção anterior, a abordagem utilizada neste trabalho resulta na construção de um modelo do agente que, ao mesmo tempo é formal e executável. Este modelo, além de possuir um alto nível de abstração, faz uso de diversas técnicas, como raciocínio não-monotônico, abdução e outras que permitem modelar situações que exigem do agente um comportamento racional e flexível.

A principal contribuição desta tese é a definição de uma teoria BDI de agente formal e executável que permite tanto definir e analisar formalmente agentes e propriedades destes agentes, bem como executar estes agentes, a partir de sua definição BDI. O desenvolvimento desta teoria deu origem a outras contribuições:

- extensão do cálculo de eventos para suportar ações concorrentes e eventos expontâneos, o que permite utilizar este modelo para representar ambientes dinâmicos e onde múltiplos agentes atuem;
- definição formal e computável de processos associados aos estados mentais, nomeadamente: criação de intenções a partir de desejos, criação de ações a partir das intenções, resolução de conflitos entre desejos. Estes processos não são comtemplados nos modelos formais existentes;
- integração de diversas formas de raciocínios na definição dos agentes, nomeadamente dedução, abdução e raciocínio revogável;
- implementação de um ambiente de teste de agentes onde o modelo formal é executado, de modo que a arquitetura BDI passa a ser um paradigma de implementação de agentes.

Cada um destes tópicos será abordado em detalhes nos capítulos que se seguem.

1.4 Organização do Texto

O texto a seguir está organizado da seguinte forma:

- o capítulo 2 descreve o ambiente lógico utilizado para construir o modelo do agente. Uma vez que nos artigos esta descrição é apresentada de forma resumida, este capítula apresenta a lógica em detalhes para facilitar a compreensão dos artigos que se seguem;
- o artigo apresentado no capítulo 3[MÓR 95] descreve as noções básicas sobre BDI e que o modelo deve respeitar, bem princípios básicos a serem adotados no modelo;.

- o artigo apresentado no capítulo 4[MÓR 96] descreve o papel dos estados mentais motivacionais na produção de comportamento inteligente por parte do agente;
- o artigo apresentado no capítulo 5[MÓR 97] descreve como modelar a reconsideração de intenções usando programação em lógica estendida, eliminando a carater "fanático" do comportamento do agente;
- os artigos apresentados nos capítulo 6 e 7[MÓR 98b][MÓR 98a] apresentam o modelo formal e executável do agente, utilizando as noções BDI como primitivas básicas;
- os artigos apresentados nos capítulos 7, 8, 9 e 10 apresentam diversos aspectos de uma aplicação real desenvolvida utilizando a ferramenta derivado do modelo, nomeadamente o tutor inteligente MCOE e o "kernel" BDI X-BDI.

2 O *Framework* Lógico

Como foi referido no capítulo anterior, o objetivo principal deste trabalho é definir uma teoria de agentes que reduza a distância entre a especificação e a implementação de agentes. Para tanto, a abordagem aqui adotada é utilizar um formalismo lógico que, além de fornecer as ferramentas necessárias para se modelar os vários estados mentais — um sistema formal cuja linguagem seja adequada para representação de conhecimento e que suporte diversos tipos de raciocínio — também seja tratável computacionalmente e possa ser utilizado pelo próprio agente para representar conhecimento e raciocinar. Este capítulo apresenta o ambiente lógico utilizado para este fim.

Inicialmente, na seção 2.1, é feita uma análise das lógicas modais e da sua utilização para representar estados mentais. A seguir, na seção 2.2, define-se o formalismo lógico adotado neste trabalho, nomeadamente a programação em lógica estendida com negação explícita, e semântica bem fundada estendida com negação explícita — *WFSX*. Além da linguagem e da semântica, a seção mostra como utilizar este ambiente para lidar com as contradições que naturalmente surgem na linguagem, bem como realizar outras formas de raciocínio além do dedutivo, notadamente raciocínio revogável e raciocínio abdutivo.

2.1 Visão Geral das Lógicas Modais

Os modelos formais de agentes são normalmente definidos utilizando-se uma *lógica modal* [CHE 80]. Lógicas modais são estensões feitas às lógicas proposicionais ou de primeira ordem pelo acréscimo de *operadores modais*, operadores estes que não são função dos valores verdade. Na lógica clássica, proposicional ou de primeira ordem, o significado de uma fórmula é uma função do valor verdade das suas sub-fórmulas. Assim, por exemplo, $(p \wedge q)$ será verdadeiro se tanto p como q o forem, e será falso caso contrário. Já o significado de uma sentença com um operador modal, por exemplo, $\Box p$, não depende de p ser verdadeiro ou falso.

As lógicas modais foram originalmente desenvolvidas para formalizar argumentos envolvendo as noções de *necessidade* e *possibilidade*. Uma proposição necessária é uma proposição verdadeira que não pode ser falsa, e uma proposição possível é uma que pode vir a ser verdadeira. Sintaticamente, isto foi resolvido estendendo-se a linguagem da lógica clássica com dois operadores modais: $\Box$, o operador de necessidade, e $\Diamond$, o operador de possibilidade. Já a semântica das sentenças foi dada utilizando-se as estruturas propostas por Kripke, e depois reformuladas para o formato, hoje usual, dos *mundos possíveis* [CHE 80] [HIN 62]. A idéia é que existe um conjunto de mundos possíveis de serem atingindos a partir de um mundo corrente. Que mundos pode-se atingir a partir de um determinado mundo é especificado por uma *relação de acessibilidade* arbitrária. Cada mundo possível é visto como um conjunto de configurações possíveis, ou seja, um conjunto de valores verdades possíveis para as sentenças.

Como um exemplo do conceito de mundos possíveis [HAD 96], suponha que Ana esteja em uma conferência onde ela conhece José, vindo do México. Mais tarde, ela decide entrar em contato com ele, mas dá-se conta que não sabe seu sobrenome, nem de que instituição ele veio. Procurando na lista de afiliações fornecida pela organização da conferência, ela encontra duas pessoas com o nome José, ambos do México. Dado o que Ana acredita, portanto, as únicas possibilidades são estas duas pessoas, cada uma sendo considerada um mundo possível. As sentenças consideradas verdadeiras são aquelas que sejam verdadeiras em todos os mundos possíveis. Neste exemplo, a sentença "José vem do México".

Formalmente, então, o significado das sentenças é dado em termos de um modelo $M = \langle W, R, \alpha \rangle$, onde:

1. W é um conjunto de mundos possíveis;

2. R é uma relação binária definida sobre W, a relação de acessibilidade, onde (w_1, w_2) indica que o mundo w_2 pode ser acessado a partir do mundo w_1;

3. α é a *função de atribuição*, que determina para cada mundo $w \in W$, o valor das proposições naquele mundo w.

Ou seja, a avaliação das sentenças é dada em um determinado mundo. Uma proposição necessária, como $\Box p$, é verdadeira se p for verdadeiro em todos os mundos possíveis acessíveis a partir do mundo em questão. Já uma proposição possível, como $\Diamond p$, é verdadeira se p for verdadeira em pelo menos um dos mundos possíveis acessíveis a partir do mundo em questão. Formalmente, as fórmulas são interpretadas de acordo com um par $\langle M, w \rangle$, onde $M = \langle W, R, \alpha \rangle$ é um modelo e w um mundo pertencente ao conjunto de mundos de M, usando uma relação de satisfação $\models$ definida por:

1. $\langle M, w \rangle \models \mathbf{t}$ (a constante verdadeiro é satisfeita em qualquer mundo);

2. $\langle M, w \rangle \models p$ sse $\alpha(w) = \mathbf{t}$ (a proposição p é verdadeira no mundo w sse o valor verdade dado pela função de atribuição para p em w for verdadeiro);

3. $\langle M, w \rangle \models \neg q$ sse $\langle M, w \rangle \not\models q$ (a negação de q é verdadeira em w sse q for falso em w);

4. $\langle M, w \rangle \models p \vee q$ sse $\langle M, w \rangle \models p$ ou $\langle M, w \rangle \models q$;

5. $\langle M, w\rangle \models \Box p$ sse $\forall w' \in W.$se $(w, w') \in R$ então $\langle M, w'\rangle \models p$ (p é necessariamente verdadeiro sse for verdadeiro em todos os mundos acessíveis a partir de w);

6. $\langle M, w\rangle \models \Diamond p$ sse $\exists w' \in W.(w, w') \in R$ e $\langle M, w'\rangle \models p$ (p é possivelmente verdadeiro sse for verdadeiro em pelo menos um dos mundos possíveis acessíveis a partir de w').

Uma fórmula é, então, satisfatível se for satisfeita em pelo menos um par modelo/mundo, e é válida se for verdadeira em pelo menos um par modelo/mundo.

As lógicas modais aplicadas a conceitos intencionais surgiram da necessidade de superar alguns problemas apresentados pelas lógicas clássicas quando eram utilizadas para representar a idéia de conhecimento ou crença. Por exemplo, para representar a seguinte afirmação [GEN 87] [WOO 95]:

Maria acredita que Cronos é o pai de Zeus.

uma tradução direta (e equivocada) para lógica de primeira ordem resultaria na seguinte fórmula:

$$BEL(maria, pai(Cronos, Zeus)) \tag{2.1}$$

Tal tradução é equivocada por duas razões: a fórmula $pai(Cronos, Zeus)$ está sendo usada como um termo, o que acarreta num erro de sintaxe. Mais sério do que isto, há também um erro semântico. A lógica de primeira ordem é *referenciamente transparente*, ou seja, o significado de uma fórmula depende unicamente do significado de suas sub-fórmulas. Assim, ignorando o erro sintático, se juntamente com a fórmula 2.1 for declarado

$$Zeus = Júpiter \tag{2.2}$$

que estabelece que *Zeus* é a mesma entidade que *Júpiter* seria possível deduzir-se que

$$BEL(maria, pai(Cronos, Júpiter))$$

o que não é necessariamente verdade. O fato de Zeus e Júpiter serem realmente a mesma entidade não significa que Maria saiba (ou acredite) que eles o são.

A solução para estes problemas foi inicialmente proposta por Hintikka, utilizando as estruturas propostas por Kripke [HIN 62] como ferramenta. A idéia de Hintikka era que a noção de crenças poderia ser formalizada como um conjunto de mundos possíveis. Cada mundo possível é visto como um conjunto de configurações possíveis (valores verdades para as sentenças) dado o que o agente

Nome do Axioma	Fórmula doAxioma	Propriedade da Relação	Sistema Axiomático	Aximatização
K	$\models \Box(p \Rightarrow q) \Rightarrow (\Box p \Rightarrow \Box q)$	*Qualquer*	*Todos*	—
T	$\Box p \Rightarrow p$	Reflexiva	T	**KT**
D	$\Box p \Rightarrow \Diamond p$	Serial	S4	**KD4**
4	$\Box p \Rightarrow \Box\Box p$	Transitiva	S5-fraco	**KD45**
5	$\Diamond p \Rightarrow \Box\Diamond p$	Euclidiana	S5	**KT5**
NEC	se $\models p$ então $\Box p$	*Qualquer*	*Todos*	—

Tabela 2.1 — Relações de Acessibilidade e Axiomas

sabe. Segundo Hintikka, estes mundos possíveis seriam alternativas epistêmicas derivadas do que o agente sabe. Algo que seja verdadeiro em todas as alternativas epistêmicas seria algo que o agente saberia.

Se por um lado o conceito de mundo possíveis resolve o problema da transparência referencial, algumas desvantagens surgem quando se incorpora a idéia de mundos possíveis a semântica de uma lógica modal epistêmica. Como foi visto anteriormente, o significado dos operadores modais está ligado a uma relação de acessibilidade. Esta relação define que mundos são considerados acessíveis a partir de cada mundo possível, sendo que a validade das fórmulas é verificada com relação aos mundos acessíveis a partir do mundo corrente (parafraseando o que foi dito anteriormente, a validade de uma fórmula é verificada nos *mundos possíveis derivados a partir do que o agente sabe*).

Esta estrutura definida pelas relações de acessibilidade é responsável por uma grande parte das propriedades que se verificam nas lógicas modais. Por exemplo, se a relação de acessibilidade é *reflexiva* (um mundo possível x é acessível a partir do próprio mundo x), a lógica possui o axioma $\Box\phi \Rightarrow \phi$. Da mesma forma, existem uma série de axiomas que correspondem a determinadas propriedades da relação de acessibilidade, e que são estudados e demonstrados pela *teoria de correspondência* associada a lógica. Esta teoria estabelece uma relação entre as propriedades da relação de acessibilidade e os axiomas da lógica. A existência desta teoria de correspondência torna possível derivar resultados relativos a completude das diversas lógicas modais, resultados estes que fornecem pontos de comparação entre as diferentes lógicas modais com a semântica dada pelos mundos possíveis ou *lógicas modais normais*, como são chamadas. A correspondência entre axiomas, sistemas axiomáticos e propriedades das relações é mostrada na tabela abaixo.

Assim, graças a esta teoria de correspondência, pode-se observar que o axioma **K** é uma fórmula válida e, portanto, é um teorema em qualquer axiomatização possível (ou, em outras palavras, seja qual for a relação de acessibilidade escolhida) para as lógicas modais normais. Este axioma **K** indica que um conjunto de sentenças de lógica modal normal é fechado para a implicação. Também o axioma **NEC**, a *regra da necessitação*, que indica que todas as fórmulas válidas são necessariamente válidas, faz parte de qualquer axiomatização das lógicas modais normais, visto ser uma regra de inferência válida. Juntos, estes dois axiomas, que estão presentes em todas a lógicas modais normais, constituem um dos grandes problemas do uso das lógicas modais normais para modelar estados mentais.

Quando usadas para representar conhecimentos, as lógicas modais tem o operador $\Box$ substituido por um operador de conhecimento K, dando origem as lógicas epistêmicas. A interpretação intuitiva do operador $K\phi$ é "*o agente sabe que* ϕ". A regra de necessitação e o axioma **K,** que se verificam em qualquer lógica modal normal, são interpretados da seguinte forma:

- **K**: o conhecimento do agente é fechado segundo a implicação lógica, ou seja, o agente conhece todas as consequências de seus conhecimentos e crenças;
- **NEC**: o agente conhece todas as fórmulas válidas e, portanto, todas as tautologias.

Esta característica das lógicas modais epistêmicas é conhecida como *onisciência lógica* [HIN 62] e implica que o agente conhece todas as consequências de seus conhecimentos, conhece todas as fórmulas válidas, que se o agente tem conhecimentos inconsistentes ele acredita em qualquer coisa e que proposições equivalentes são crenças idênticas. Uma vez que se parte do princípio que os agentes possuem recursos limitados (recursos como tempo de computação e espaço de memória), estas são características que os agentes, na prática, não possuem, e que os modelos incluem. Já quando usadas para representar um estado pro-ativo, como as intenções, as lógicas modais tem o operador $\Box$ substituido por um operador de intenções I. Se a interpretação de $I\phi$ for "*o agente tem a intenção de* ϕ", aqueles dois axiomas implicam que o agente sempre tem por intenção todas as conseqüências lógicas de suas ações, mesmo aquelas que ele não foi capaz de antever, o que novemente não é uma característica desejada para os agentes.

Assim, estas características inerentes as lógicas modais constituem uma grande desvantagem na sua utilização como ferramenta para modelar agentes[1]. Mesmo que se aceite estas discrepâncias introduzidas pelas lógicas modais normais, há ainda o problema do tratamento computacional destas lógicas. Halpern and Moses [HAL 92] [HAL 96] apontam que, ainda que a provabilidade destas lógicas seja um problema decidível, a determinação da satisfatibilidade e da validade são problemas de alta complexidade (PSPACE-completos). Ou seja, no caso

[1]Há outras abordagens alternativas para formalizar e racioacinar sobre estados mentais utulizando lógicas modais, tais como a utilização de outras estruturas semânticas [KON 93] [GOL 93], abordagens sintáticas [KON 86] ou abordagens híbridas, que misturam mundos possíveis e outras estrututas tentando evitar os problemas já mencionados. Uma análise detalhada destas alternativas foge ao escopo deste trabalho.

geral, a automação não é possível, o mesmo sendo generalizado para as lógicas multi-modais quantificadas (como em [COH 90] [RAO 91] [SIN 94]) que são usadas para definir as teorias de agentes. O fato destas lógicas não disporem de procedimentos de derivação para manipulação de suas sentenças que sejam corretos e completos com relação a semântica destas linguagens faz com que mesmas possam ser usadas como linguagem de especificação, mas não como ferrementas de representação do conhecimento ou ferramenta básica para implementar agentes.

2.2 A Programação em Lógica Estendida

Como foi referido no capítulo 1, a abordagem adotada neste trabalho para construir uma teoria de agentes de possa servir tanto de ferramenta de análise como linguagem de representação e raciocínio por parte dos agentes consiste selecionar uma outra lógica ou linguagem formal que atenda a ambas as necessidades, e então construir o modelo de agentes nesta linguagem. Portanto, o requisito básico para a lógica selecionada é que ela possa ser utilizada tanto para especificação como para implementação dos agentes. De fato, esta é a maneira usual como a linguagem é vista na *programação em lógica*. Kowalski, em [KOW 86], afirma que o que distingue uma especificação de uma implementação em programação em lógica são questões de performance, visto que ambas são executáveis. Assim sendo, a programação em lógica será a base para a construção da teoria de agentes neste trabalho.

A linguagem utilizada na construção da teoria deve, também, ter a expressividade necessária para representar e manipular conhecimento de diversas formas:

- a linguagem necessita de procedimentos de derivação que sejam corretos e completos com relação a sua semântica, a fim de que o agente possa manipular as sentenças da linguagem;

- a linguagem deve possibilitar a representação e a manipulação de informações contraditórias. Como foi referido no capítulo 1, quando se lida com estados mentais pro-ativos faz-se necessário ora representar simultaneamente informações contraditórias, ora detectar e resolver contradições que surjam durante a manipulação dos estados mentais;

- a linguagem deve possuir mecanismos que permitam outras formas de raciocínio que não a dedução, notadamente abdução e algumas formas de raciocínio não-monotônico.

O ambiente escolhido neste trabalho é a *programação em lógica estendida com negação explícita* (**E**xtended **L**ogic **P**rogramming ou **ELP**) com a *semântica bem-fundada estendida com a negação explícita* (**W**ell-**F**ounded **S**emantics e**X**tended for the explicit negation ou **WFSX**). A programação em lógica estendida acrescenta à programação em lógica normal uma segunda forma de negação, a *negação explícita*. Por programação em lógica normal entende-se a programação utilizando cláusulas de Horn aumentadas com operador *not*. Esta forma de negação é dita *negação por omissão* ou *negação por falha*, no sentido de *falha em provar ser verdadeiro*. Assim, uma conclusão negativa é obtida sempre "por default" ou implicitamente apenas se a correspondente conclusão positiva não pode ser obtida em um número finito de passos. A semântica da linguagem, neste caso, é dada utilizando-se a hipótese do mundo fechado[2], que estabelece que tudo aquilo que não está declarado como verdadeiro é falso. Para esta semântica, o procedimento de derivação SLDNF é completo e correto, e define uma semântica operacional para a linguagem.

No entanto, programas que apresentem computações infinitas não recebem significado por aquela semântica. Para resolver este problema (atribuir significado para qualquer programa em lógica) surgiram uma série propostas de novas semânticas, entre elas a *semântica bem-fundada* (**W**ell-**F**ounded **S**emantics ou **WFS**). A WFS lida semanticamente com computações "top-down" infinitas e é capaz de atribuir significado para qualquer programa, assinalando a estas computações, como valor verdade, ou indefinido ou falso.

Com a evolução da programação em lógica, e dada a importância e a necessidade em áreas como bancos de dados dedutivos, representação do conhecimento e raciocínio não monotônico, introduziu-se na linguagem aquela nova forma explícita de negação, em adição à forma implícita fornecida pela negação por omissão. Com a mudança da linguagem, surgiram novas semânticas que tinham em conta a extensão da programação em lógica com a negação explícita. A WFSX, uma estensão feita a partir da WFS, é particularmente interessante pois fornece uma expressividade maior que captura uma grande variedade de formas de raciocínio, bem como serve de instrumento para programá-los. Além destas vantagens, a WFSX possui um procedimento de derivação correto e "top-down", nos mesmos moldes do SLDNF para programação em lógica normal, notadamente a *derivação linear seletiva para programas estendidos* (**S**elected **L**inear resolution for e**X**tended programs ou **SLX**).

Quando se introduz alguma forma de negação explícita na linguagem, torna-se necessário lidar com a possibilidade de contradições na linguagem. O ambiente definido em [ALF 95] e [ALF 96], sobre a ELP com WFSX, permite lidar com estas situações de forma genérica. Assim, quando as contradições surgem, o ambiente fornece mecanismos para removê-las alterando o valor verdade de um subconjunto pré-definido de literais. Este mecanismo de remoção de contradições é obtido através de uma variante da SLX que computa as combinações possíveis de alterações de valores verdade que garantem a remoção. Através deste mecanismo

[2]Do inglês *closed world assumption* (**CWA**).

de remoção é possivel realizar os diferentes tipos de raciocínio não-monotônicos já mencionados.

Vários fatores, portanto, fazem da ELP com a WFSX um formalismo adequado para satisfazer os requisitos necessários mencionados anteriormente:

- possui uma linguagem formalmente definida. Assim, permite definir formalmente os agentes e as propriedades que se deseja verificar nos agentes, bem como verificar se um determinado agente satisfaz uma determinada propriedade;
- possui um procedimento de derivação construtivo e efetivo para verificar a satisfatibilidade de fórmulas de linguagem — o SLX — o qual é completo e correto em relação a **WFSX**[ALF 96]. Isto permite provar se um determinado literal pertence ou não ao modelo do programa, sem exigir que se calcule o modelo completo deste programa;
- possui um procedimento efetivo de *revisão de programas contraditórios*, o qual permite realizar com as sentenças da linguagem outros tipos de raciocínio que não a dedução, nomeadamente *raciocínio revogável*[3] e *raciocínio abdutivo*[4]. Estas formas de raciocínio são necessárias para definição dos agentes.

As seções a seguir apresentam o sistema formal, bem como sua utilização para definir diversos tipos de raciocínio não-monotônico.

2.2.1 A Linguagem e sua Semântica

Como é usual na lógica, a definição da linguagem compreende a definição da sua sintaxe, bem como das estruturas semânticas que lhe dão significado. Estas definições são apresentadas nesta seção.

A programação em lógica estendida acrescenta a linguagem da programação em lógica normal um segundo tipo de negação, a negação explícita, representada pelo símbolo $\neg$, em adição a negação por omissão ou negação implícita, como é chamada neste contexto.

Definition 1 ***(Linguagem)****Um programa em lógica estendido é um conjunto de regras na forma*

$$H \leftarrow L_1, \ldots, L_m, not\ L_{m+1}, \ldots, not\ L_n \qquad (0 \leq m \leq n)$$

onde cada L_i é um literal objetivo*. Um literal objetivo pode ser tanto um átomo A ou sua negação explícita $\neg A$, e conjunto de todos os literais objetivos de*

[3]Do inglês *defeasible reasoning.*

[4]Do inglês *abductive reasoning.*

um programa P é chamado de base de Herbrand estendida *de P, sendo denotada por* $\mathcal{H}(P)$. *O símbolo not representa a* negação por omissão, negação implícita *ou* negação por falha, *sendo que um literal not L é um* literal por omissão. *Literais podem ser somente literais objetivos ou por omissão, sendo que $\neg\neg L \equiv L$.*

A inclusão da negação explícita aumenta a expressividade da linguagem. Na programação em lógica normal, informação negativa só pode ser representada implicitamente, ou seja, um literal é considerado falso se não razão para considerá-lo verdadeiro. Embora isto seja suficiente em muitos casos, a representação explícita de informação negativa é importante em áreas como processamento de língua natural e raciocínio de senso-comum [ALF 95]. Considere o seguinte exemplo [ALF 96]:

Example 2 *Para representar a sentença "Pinguins não voam" utilizando programação em lógica normal seria necessário recorrer a algum recurso do tipo:*

$$\texttt{nao\_voa(X)} \leftarrow \texttt{pinguim(X)}$$

O problema deste tipo de representação, no entanto, perde a conexão entre os predicados `nao_voa` *e* `voa`. *Assim, ao representar a sentença "Pássaros voam" como*

$$\texttt{voa(X)} \leftarrow \texttt{passaro(X)}$$

seria desejável preservar a relação entre os predicados.

Ainda, segundo [ALF 95], a importância destas conexões crescem quando a informação negativa é utilizada para representar excessões a regras, como no exemplo acima. Há, ainda, certas situações em que as duas negações assumem significados bastante diferentes.

Example 3 *Dado a sentença "Um ônibus escolar pode cruzar trilhos de trem desde que não haja um trem se aproximando". Sem a negação explícita, esta sentença seria representada pela regra*

$$\texttt{cruza} \leftarrow \texttt{not trem}$$

No entanto, isto permite que se cruze o trem quando não há nenhuma informação sobre a presença de um trem (visto que o not A é verdadeiro quando nada se sabe sobre A). Já se a representação for

$$\texttt{cruza} \leftarrow \neg\texttt{trem}$$

o ônibus somente cruza os trilhos se há informação afirmando que não há nenhum trem se aproximando.

Definida a sintaxe da linguagem, é necessário atribuir significado as sentenças bem-formadas. Como é usual, o significado das sentenças é dado por uma estrutura matemática conjunto teorética, que define os elementos representados pela linguagem. No caso da programação em lógica estendida, o significado de um conjunto de sentenças ou programa em lógica estendido é dado por um conjunto de literais. Como foi visto anteriormente, a semântica adotada, a $WFSX$, estende a WFS que é definida para programas em lógica normal para programas em lógica estendidos com a negação explícita. As duas formas de negação são relacionadas pelo *princípio da coerência* [ALF 96], que estipula que a negação explícita implica na negação implícita, ou seja, se um literal objetivo $\neg A$ for verdadeiro então a negação por omissão *not* A também será verdadeira. Além disto, como a WFS para programas em lógica normal, a $WFSX$ atribui significado também para programas com computações infinitas, atribuindo o valor verdade indefinido aos literais envolvidos. Tem-se, então, uma lógica a três valores: verdadeiro, falso e indefinido.

Para definir formalmente a semântica dos programas, segue-se a sequência de definições mostrada na figura 2.1.

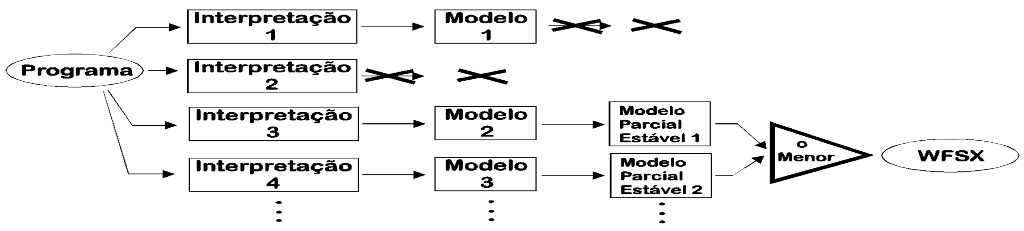

Figura 2.1 — Definição da WFSX.

A cada programa pode-se associar uma ou mais interpretações, que são atribuições de valores verdades a cada um dos literais que fazem parte do programa. Esta atribuição de valores verdades impõe como restrição apenas que, se um literal objetivo L pertence a uma interpretação I, então o literal por omissão *not* L não pertencerá a I. No entanto, as interpretações que interessam são aquelas que satisfazem (tornam verdadeiras) todas as regras que compõem os programas. Estas interpretações são os modelos do programa em questão. Novamente, nenhuma restrição adicional é feita aos modelos. No entanto, interessam apenas aqueles modelos que respeitem o princípio da coerência e que sejam não-contraditórios[5]. Estes modelos, os modelos parciais estáveis, são definidos através do ponto fixo de um operador sobre o programa em lógica. Um programa em lógica estendida pode possuir mais de um modelo parcial estável (ou seja, vários modelos que satisfaçam as condições de coerência, não-contradição e atribuam o

[5]Isto parece não satisfazer os requesitos enumerados anteriormente, visto ser desejado lidar também com programas contraditórios. No entanto, embora a WFSX não atribua significado a programas contraditórios, como será visto adiante, e a partir de sua definição que são construídas as ferramentas para aceitar, detectar e remover contradições.

significado desejado ao programa, como na WFS). A semântica do programa, de acordo com a $WFSX$, é dada pelo menor dos modelos parciais estáveis.

Inicialmente, define-se a noção de *interpretação* de um programa em lógica estendido. Basicamente, uma interpretação de um ELP é um conjunto de literais. Um literal L é verdadeiro em uma interpretação I se $L \in I$, L é falso em I se *not* $L \in I$, e L é indefinido caso contrário.

Definition 2 ***(Interpretação)*** *Uma* interpretação *I de um programa em lógica extendido P é denotado por $T \cup not\ F$, onde T e F são subconjuntos disjuntos de $\mathcal{H}(P)$. Para $F = \{a_1, \ldots, a_n\}$, $not\ F$ denota $\{not\ a_1, \ldots, not\ a_n\}$. O conjunto T contém todos os literais objetivos ground que sejam* verdadeiros *em I, o conjunto F contém todos os literais objetivos ground que sejam* falsos por omissão *em I e o valor verdade dos literais objetivos em $\mathcal{H}(P) - I$ é* indefinido*. O valor verdade de um literal not L é o complemento a três valores de L, ou seja:*

Valor Verdade	***Complemento***
t *(*true*, verdadeiro)*	**f** *(*false*, falso)*
f	**t**
u *(*undefined*, indefinido)*	**u**

É importante observar que esta definição de interpretação não garante que os dois tipos de negação, implícita e explícita, estejam relacionadas pelo princípio da coerência bem como não garante que a interpretação não seja contraditória. Portanto, não são todas as interpretações que interessam. Em particular, interessam aquelas interpretações que tornem as regras do programa verdadeiras, o que leva aos conceitos de satisfação e de modelo.

Definition 3 ***(Satisfação de um Programa em Lógica Estendido)*** *Seja I uma interpretação de um dado programa em lógica estendido. Diz-se que:*

- *I* satisfaz *um literal objetivo ou um literal por omissão, denotado por $I \models L$, se e somente se $L \in I$;*
- *I satisfaz a conjunção $L_1, \ldots, L_n, not\ G_1, \ldots, not\ G_m$,*

  denotada por $I \models L_1, \ldots, L_n, not\ G_1, \ldots, not\ G_m$ se e somente se

  $I \models L_1, \ldots,$ e $I \models L_n$, e $I \models not\ G_1,, \ldots,$ e $I \models not\ G_m$;
- *I satisfaz a regra $H \leftarrow L_1, \ldots, L_n, not\ G_1, \ldots, not\ G_m$,*

  denotado por $I \models (H \leftarrow L_1, \ldots, L_n, not\ G_1, \ldots, not\ G_m)$,

  se e seomente se sempre que o corpo da regra for satisfeito por I, a cabeça também for satisfeita por I.

Definition 4 ***(Modelo)*** *Uma interpretação I é um modelo de um programa em lógica extendido P se e somente se toda regra de P é satisfeita por I.*

Ou seja, um modelo de um programa é uma interpretação que faz com que todas as regras do programa serem verdadeiras. Observa-se que, de acordo com esta definição, um programa em lógica estendido pode ter vários modelos.

Example 4 *Considere o programa P_4 abaixo:*

```
¬b
b    ←  a
a    ←  not a, not b
c    ←  not ¬c
¬c   ←  not c
```

Tanto $I_1 = \{\neg b, not\ b\}$ e $I_2 = \{\neg b, not\ b, not\ \neg c\}$ são modelos para o programa P_4.

Novamente, nenhuma retrição é feita à noção de modelo, de modo que podem existir modelos que não respitem o princípio da coerência ou que sejam contraditórios. A definição da semântica $WFSX$ estipula algumas condições adicionais que atribuem um único modelo para cada programa. Isto pode ser feito de diversas maneira. A definição aqui adotada baseia-se no funcionamento do pontos fixos alternados de um operador semalhante ao operador Γ definido por Gelfond-Lifschitz e extraído de [ALF 95]. Dados um programa em lógica estendido e uma interpretação, o operador Γ simplesmente transforma este programa em lógica estendido em um programa em lógica definido (sem negação, implícita ou explícita) a partir das informações da interpretação e computa o modelo mínimo do programa resultante.

Definition 5 ***(Operador** Γ**)*** *Seja P um programa em lógica extendido, I uma interpretação.*

1. *Seja P' (respectivamente I') obtido a partir de P (respectivamente I) representando-se cada literal $\neg A$ por um novo átomo, por exemplo $\neg\_A$. A* tranformação-GL $\frac{P'}{I'}$ *é o programa obtido a partir de P' removendo-se todas as regras contendo um literal por omissão not A tal que $A \in I'$ e removendo-se todos os demais literais por omissão em P'.*

2. *Seja J o modelo mínimo de $\frac{P'}{I'}$. $\Gamma_P(I)$ é obtido a partir de J substituindo-se os átomos $\neg\_A$ introduzidos inicialmente pelos átomos $\neg A$ correspondentes.*

O função deste operador é definir o modelo mínimo para um programa, dada uma interpretação. Isto é feito de acordo com a noção de modelo mínimo para programas em lógica definidos. Assim, inicialmente (no passo 1) o operador Γ remove a negação explícita transformando todo literal objetivo negativo em um novo literal objetivo positivo. A seguir, remove todas as negações por omissão, da seguinte forma: se um literal *not A* é falso de acordo com a interpretação (ou seja, $A \in I'$) as regras onde *not* A ocorrem são removidas, visto que não poderiam ser satisfeitas; já os literais *not* A que sejam verdadeiros (ou seja, $A \notin I'$) são removidos, pois sabe-se de antemão que são verdadeiros, logo a setisfaçam das regras onde eles ocorrem não dependem destes literais. Obtido o programa em lógica definido, computa-se o seu modelo mínimo (no passo 2) O resultado da aplicação do operador é o modelo mínimo do programa em lógica definido recuperando-se as negações explícitas.

Antes de definir o que são os modelos parciais estáveis, é preciso garantir que o princípio da coerêcia seja respeitado no modelo do programa. Para tanto, utiliza-se a noção de versão semi-normal do programa.

Definition 6 ***(Versão Semi-Normal de um Programa)*** *A* versão semi-normal de um programa P *é o programa P_S obtido a partir de P adicionando-se ao corpo de cada regra (inclusive regras cujo corpo seja vazio) $L \leftarrow Body$ o literal por omissão not $\neg L$, onde $\neg L$ é o complemento (com respeito a negação explícita) de L.*

O princípio da coerência estipula que se um literal objetivo $\neg A$ for verdadeiro então a negação por omissão *not* A também será verdadeira. Ou seja:

"Dada uma interpretação $I = T \cup not\ F$, se $\neg L \in T$ então $L \in F$."

Na versão semi-normal de um programa, um determinado literal só é verdadeiro se não houver nenhuma regra que indique que seu complemento por omissão é verdade, garantindo a coerência. A definição a seguir impõe as condições necessária a definição de modelo de um programa em lógica estendido.

Definition 7 ***(Modelo Parcial Estável)*** *Um conjunto de literais objetivos T gera um* modelo parcial estável[6] (**PSM**) *de um programa em lógica extendido P se e somente se:*

1. $T = \Gamma_P \Gamma_{P_S}(T)$; *e*
2. $T \subseteq \Gamma_{P_S}(T)$.

O PSM gerado por T é a interpretação $T \cup not(\mathcal{H}(P) - \Gamma_{P_S}(T))$.

Isto quer dizer que os modelos parciais estáveis são determinados pelos pontos fixos de $\Gamma_P \Gamma_{P_S}$ (o operador Γ definido sobre o programa P aplicado sobre o resultado do operador Γ definido sobre a versão semi-normal de P). Dado um ponto fixo T, os literais objetivos em T são verdadeiros no PSM, os literais objetivos que não estão em $\Gamma_{P_S}(T)$ são falsos por omissão, e todos os outros são indefinidos. Assim sendo, os literais objetivos em $\Gamma_{P_S}(T)$ são todos aqueles que

[6] *Do inglês,* Partial Stable Model.

são ou verdadeiros ou indefinidos. É preciso notar que a condição 2 faz com que um literal qualquer não possa ser, simultaneamente, verdadeiro e falso por omissão, i.e., se tal literal pertence a T ele não pertence a $\mathcal{H}(P) - \Gamma_{P_S}(T)$ (e vice-versa). Já o uso de Γ_{P_S} impõe o princípio da coerência, ou seja, se $\neg L \in T$ ($\neg L$ é verdadeiro) então em $\Gamma_{P_S}(T)$ (a aplicação do operador Γ sobre a versão semi-normal do programa P) todas as regras para L são removidas e, consequentemente, $L \notin \Gamma_{P_S}(T)$ (L é falso por omissão).

É importante observar que programas contraditórios (onde um literal L e seu complemento $\neg L$ são verdadeiros) não possuem PSMs. Como já foi referido, isto parece não satisfazer os requisitos enumerados anteriormente, visto ser desejado lidar também com programas contraditórios. No entanto, como será visto adiante, a partir da definição da $WFSX$ é que são construídas as ferramentas para aceitar, detectar e remover contradições.

Example 5 *(extraído de [ALF 95])O programa $P = \{a; \neg a\}$ não possui modelos parciais estáveis. De fato, o único ponto fixo de $\Gamma_P\Gamma_{P_S}(T)$ é $\{a, \neg a\}$ e $\{a, \neg a\} \nsubseteq \Gamma_{P_S}(T) = \{\}$.*

A semântica $WFSX$ é dada pelo menor dos PSMs de um programa.

Definition 8 ***(Semântica WFSX)*** *Todo programa não-contraditório P possui um modelo parcial estável mínimo (em relação a inclusão de conjuntos $\subseteq$) chamado de modelo bem-fundado de P e denotado por $WFM(P)$. Este menor modelo pode ser obtido iterativamente, através da aplicação dos operadores Γ_P e Γ_{P_S}.*

A definição "bottom-up" iterativa para o $WFM(P)$ é dada pela sequência transfinita $\{I_\alpha\}$:

$$\begin{array}{lcl} I_0 & = & \{\} \\ I_{\alpha+1} & = & \Gamma_P\Gamma_{P_S}(I_\alpha) \\ I_\delta & = & \bigcup\{I_\alpha \ / \ \alpha < \delta\} \text{ para um ordinal limite } \delta \end{array}$$

Existe um menor ordinal λ para a sequência acima, tal que I_λ é o menor ponto fixo de $\Gamma_P\Gamma_{P_S}$ e o $WFM(P) = I_\lambda \cup not(\mathcal{H}(P) - \Gamma_{P_S} I_\lambda)$.

Nesta definição construtiva, os literais obtidos após a aplicação de $\Gamma_P\Gamma_{P_S}$ (ou seja, em algum I_α) são verdadeiros no $WFM(P)$ e os literais que não são obtidos após uma aplicação de Γ_{P_S} (ou seja, que não estão em $\Gamma_{P_S}(I_\alpha)$ para algum α) são falsos por omissão no $WFM(P)$.

Example 6 *(extraído de [ALF 96]) Considere o programa P_6 abaixo:*

```
a ← not b.      ¬a.
b ← not a.
```

Para aplicar a definição "bottom-up" da $WFSX$, é preciso construir a versão semi-normal de P_6. Assim, acrescentando os literais por omissão apropriados, temos P_{6_S}:

$$\begin{array}{lll} \mathtt{a} & \leftarrow & \mathtt{not\ b, not\ \neg a.} \\ \mathtt{b} & \leftarrow & \mathtt{not\ a, not\ \neg b.} \end{array} \qquad \begin{array}{lll} \mathtt{\neg a} & \leftarrow & \mathtt{not\ a.} \end{array}$$

Aplicando a definição iterativa da $WFSX$, temos:

1. $I_0 = \{\}$

2. $I_1 = \Gamma_{P_6}\Gamma_{P_{6S}}(I_0)$.

   Ora, $\Gamma_{P_{6S}}(I_0) = \{a, b, \neg a\}$, pois como I_0 é vazio, todos os literais por omissão são verdadeiros, logo nenhuma regra é removida de $\Gamma_{P_{6_S}}$, e todos os literais por omissão são removidos. O programa resultante é

   $$\begin{array}{ll} \mathtt{a.} & \mathtt{\neg a.} \\ \mathtt{b.} & \end{array}$$

   sendo seu modelo mínimo $\{a, b, \neg a\}$. Aplicando-se Γ_{P_6} sobre este resultado temos $\Gamma_{P_6}(\{a, b, \neg a\}) = \{\neg a\}$, pois como a e b pertencem ao conjunto, as duas regras com literais por omissão sobre a e b são removidas. O programa resultante é

   $$\mathtt{\neg a.}$$

   e seu modelo mínimo $\{\neg a\}$. Assim, temos que $I_1 = \{\neg a\}$;

3. $I_2 = \Gamma_{P_6}\Gamma_{P_{6_S}}(I_1)$. *Temos que $\Gamma_{P_{6S}}(I_1) = \{b, \neg a\}$ e $\Gamma_{P_6}(\{b, \neg a\}) = \{b, \neg a\}$. Assim, $I_2 = \{b, \neg a\}$. Como o próximo passo mostrará, este o ponto fixo da sequência;*

4. $I_3 = \Gamma_{P_6}\Gamma_{P_{6_S}}(I_2)$. *Temos que $\Gamma_{P_{6_S}}(I_2) = \{b, \neg a\}$ e $\Gamma_{P_6}(\{b, \neg a\}) = \{b, \neg a\}$.*

Logo, a $WFSX$ é dada por $WFM(P_6) = I_3 \cup not(\mathcal{H}(P_6) - \Gamma_{P_{6_S}} I_3) = \{b, \neg a\} \cup \{not\ a, not\ \neg b\}$.

De fato, como há uma regra $\mathtt{\neg a}$ o literal $\mathtt{\neg a}$ deve ser verdadeiro; pelo princípio da coerência, também *not a* deve ser verdadeiro. Uma vez que *not a* é verdadeiro, a regra $\mathtt{b} \leftarrow \mathtt{not\ a.}$ torna b verdadeiro e, pelo princípio da coerência, *not* $\neg b$ também é verdadeiro. Exatamente como cálculado pela semântica.

2.2.2 Detectando Contradições

Como foi mencionado anteriormente, ao se extender a linguagem para aceitar negação explícita, podem surgir programas contraditórios. É importante, então, ser-se capaz de não somente executar programas livres de contradição, mas também de detectar tais contradições, a fim de removê-las. Para remover contradições, além de detectá-las, é preciso saber-se as causas destas contradições, ou seja, quais as pressuposições feitas no programa suportam as contradições e são responsáveis por elas.

A definição da $WFSX$, no entanto, não é suficiente para apontar tais pressuposições[7]. Desta forma, a fim de se possuir uma ferramenta que possibilite apontar as causas das contradições, generaliza-se a $WFSX$ de modo a se definir a noção de *modelo bem fundado paraconsistente* para programas contraditórios, e consequentemente uma $WFSX$ paraconsistente ($WFSX_P$). A idéia básica da $WFSX_P$ é calcular, sempre respeitando o princípio da coerência, todas as consequências do programa, mesmo aquelas que originam contradições e aquelas que derivam de contradições, como no exemplo a seguir:

Example 7 *[ALF 95]Considere o programa P_7:*

$$\begin{array}{llll} \texttt{a} \leftarrow \texttt{not b} & \textit{(i)} & \texttt{d} \leftarrow \texttt{not a} & \textit{(iii)} \\ \neg\texttt{a} \leftarrow \texttt{not c} & \textit{(ii)} & \texttt{e} \leftarrow \texttt{not } \neg\texttt{a} & \textit{(iv)} \end{array}$$

1. *not b e not c são verdadeiros, visto que não há regras para b e c;*

2. *a e ¬a valem a partir de 1 e das cláusulas (i) e (ii);*

3. *not a e not ¬a valem a partir de 2 e do princípio da coerência;*

4. *d e e valem a partir de 3 e das regras (iii) e (iv);*

5. *not d e not e valem a partir de 2 e das regras (iii) e (iv) (são as únicas regras para d e e);*

6. *not ¬d e not ¬e valem a partir de 4 e do princípio da coerência.*

Logo, o conjunto de todos os literais que são consequência do programa P_7 é

$$\{not\ b, not\ c, a, \neg a, not\ a, not\ \neg a, d, e, not\ d, not\ e\}$$

Neste contexto, a $WFSX_P$ é uma *pseudo-semântica*[ALF 95], i.e., não uma semântica propriamente dita mas uma ferramenta necessária para detectar as contradições e suas causas nos casos em que a $WFSX$ não atribui significado ao programa. A semântica assinalada ao programa original será a $WFSX$ do programa resultante após a remoção das contradições.

É preciso, então, extender a definição da $WFSX$ para que esta se comporte como no exemplo acima. A $WFSX$ não está definida para programas

[7]De fato, de acordo com a definição de $WFM(P)$, um programa contraditório P não possui um modelo bem fundado.

contraditórios porque tais programas não possuem PSMs. Pela sua definição, um programa não tem PSMs se ele não possui pontos fixos para $\Gamma_P\Gamma_{P_S}$ ou se todo ponto fixo T de $\Gamma_P\Gamma_{P_S}$ não satisfaz a condição $T \subseteq \Gamma_{P_S}T$. Mas prova-se que todo programa, contraditório ou não, possui pontos fixos para $\Gamma_P\Gamma_{P_S}$(teorema 5.1, em [ALF 95]). Portanto, se para algum programa P, o menor ponto fixo de $\Gamma_P\Gamma_{P_S}$ respeita a condição (da definição) $T \subseteq \Gamma_{P_S}T$, então o programa é não-contraditório e o menor ponto fixo é o seu WFM. Caso contrário, o programa é contraditório e, desta forma, o teste para se saber se um programa é contraditório dado o seu menor ponto fixo de $\Gamma_P\Gamma_{P_S}$ pode ser reduzido a:

"O programa é não-contraditório se e somente se o seu menor
ponto fixo de $\Gamma_P\Gamma_{P_S}$ não possue pares de literais complementares quanto
à negação explícita (teorema5.3,em[ALF 95])."

Logo, a definição de WFM poderia ser simplificada removendo-se a condição $T \subseteq \Gamma_{P_S}T$ e substituindo-a pelo teste se T tem literais complementares quanto a negação explícita, o que garante que os literais não sejam tanto verdadeiros como falsos por omissão. Sem esta condição, no entanto, tal garantia não existe, e isto é exatamente o que se quer para definir a versão paraconsistente da $WFSX$. Assim sendo, a definição da $WFSX_P$ é dada construindo-se o WFM através do menor ponto fixo de $\Gamma_P\Gamma_{P_S}$ sem a exigência que $T \subseteq \Gamma_{P_S}T$. Daí, tem-se que programas contraditórios são aqueles que contém pares complementares de literais no seu WFM.

Definition 9 ***(WFSX Paraconsistente)****Seja P um programa extendido cujo menor ponto fixo de $\Gamma_P\Gamma_{P_S}$ é T. O modelo bem-fundado paraconsistentede P é o conjunto $WFM_P = T \cup not\ \{\mathcal{H} - \Gamma_{P_S}T)\}$. P é não-contraditório se e somente se para nenhum literal objetivo L, $\{L, \neg L\} \subseteq T$ e, neste caso, $WFM_P(P) = WFM(P)$.*

Esta definição garante tanto que todas as consequências do programa são calculadas no modelo, tanto aquelas que derivam contradições como aquelas que se originam de contradições, ao mesmo tempo que preserva o princípio da coerência, como pode-se verificar no exemplo 7 acima.

Portanto, com a $WFSX_P$, já é possível detectar-se contradições, bem como determinar a origem destas contradições, o que permite removê-las.

2.2.3 Revisando o Programa

Uma vez que se detecte uma contradição em um programa, é necessário revisá-lo. Para efetuar esta remoção e restaurar a consistência dos programas, estes são submetidos a um processo de revisão que se baseia na possibilidade de alterar o valor verdade de um conjunto de literais, os *literais revisíveis*. Os literais revisíveis são escolhidos entre aqueles considerados básicos, ou seja, aqueles que não dependem de nenhum outro literal (para os quais há somente fatos ou não há regra nenhuma).

Assim, inicialmente, define-se uma estrutura que divide explicitamente o programa em um par chamado *estado de programa*[ALF 95]: um dos elementos contém um subprograma que não pode ser modificado; o outro elemento contém um subprograma que pode ser modificado de uma maneira pré-determinada, notadamente adicionando-se regras de um formato simples e fixo somente para o conjunto de literais revisíveis. Esta segunda parte é definida de tal forma que ela contém somente regras para literais que não dependem de nenhum outro. Um estado é transformado em outro estado simplesmente modificando esta segunda parte.

A definição da $WFSX_P$ considera que as contradições surgem apenas da inexistência de um modelo consistente (ou seja, que pelo menos um par L, $\neg]L$ esteja presente no modelo bem-fundado do programa). Em geral, no entanto, podem haver outras razões para considerar um estado do programa como não-aceitável, resultante de restrições sobre o significado pretendido para o programa. Para englobar este tipo de situação, estende-se a linguagem definida com retrições de integridade.

Definition 10 ***(Restrições de Integridade)*** *Uma* restrições de integridade *de um programa em lógica estendido P possui o seguinte formato*

$$L_1 \vee \cdots \vee L_n \Leftarrow L_{n+1} \wedge \ldots \wedge L_{n+m} \qquad (n+m \geq 0)$$

Definition 11 *onde L_i são literais (objetivos ou por omissão) pertencentes a linguagem de P.*

Uma restrição de integridade nesta forma denota que pelo menos um dos literais da cabeça (o consequente da regra) deve ser verdadeiro se o seu corpo (o antecedente da regra) for verdadeiro.Caso a cabeça seja vazia, associa-se a ela o símbolo **f**, e a ao corpo vazio associa-se o símbolo **t**. Uma *teoria de integridade* é um conjunto de restrições de integridade, representando a conjunção de todas as restrições.

Definition 12 ***(Satisfação de uma Restrição de Integridade)*** *Dado um conjunto de literais (contraditórios ou não) I, uma restrição de integridade* ground *é violada por I sse cada um dos literais $L_{n+1}, \ldots, L_{n+m}$ pertencer a I e se nenhum dos literais $L_1, \cdots, L_n$ pertencer a I. Caso contrário, a restrição é* satisfeita *por I.*

Ou seja, uma restrição de integridade é satisfeita se e somente se a implicação clássica $(L_{n+1} \in I) \wedge \ldots \wedge (L_{n+m} \in I) \Rightarrow (L_1 \in I) \vee \cdots \vee (L_n \in I)$ for satisfeita.

Assim, um programa é contraditório se e somente se seu modelo bem-fundado for contraditório ou se violar alguma de suas restrições de integridade. Esta definição pode ser simplificada removendo-se a primeira condição e acrescentando, para todos os literais L do programa, uma restrição na forma $\mathbf{f} \Leftarrow L \wedge \neg L$.

A remoção das contradições que surgem é feita alterando-se o valor verdade dos literais revisíveis. Quando não há restrições de integridade (e contradições surgem apenas pela inexistência da $WFSX$ consistente), as contradições podem sempre ser removidas trocando-se o valor verdade de literais somente para indefinido. Mas, no caso geral, há necessidade, eventualmente, de mudar o valor verdade de literais para verdadeiro ou falso. A abordagem adotada aqui (conforme definido em [ALF 95]) é de realizar a remoção das contradições introduzindo ou removendo regras para os literais revisíveis, de maneira que se possa trocar seus valores verdades de um valor qualquer para outro valor qualquer. Se nenhuma restrição exigir, o valor verdade é modificado minimamente para indefinido. Caso se deseje impor a revisão somente de verdadeiro para falso e vice-versa, basta introduzir uma restrição na forma $L \vee \mathit{not}\ L \Leftarrow \mathbf{t}$.

A definição do ambiente de revisão proposto inicia pela definição formal de *programas abertos* e *estados de programas.*

Definition 13 ***(Programas Abertos)*** *Um* programa aberto *é uma tripla $\langle P, IC, O_P\rangle$, onde P é um programa em lógica estendido, IC é um conjunto de restrições de integridade e O_P, o conjunto de* literais abertos, *é um subconjunto de literais objetivos de $\mathcal{H}(P)$, tal que $L \in O_P$ sse $\neg L \in O_P$, ou seja, se um literal L pertence ao conjunto, então seu complemento pela negação explícita também pertence. Adicionalmente, não há regras em P para nenhum dos literais abertos.*

Os fatos básicos, sujeitos a variação, são representados no programa aberto pelos literais abertos, os quais não tem seu valor verdade determinado *a priori*. A parte fixa do programa aberto é um programa em lógica estendido arbitrário, que deriva informações adicionais a partir dos literais abertos, uma vez que seus valores sejam conhecidos. As restrições de integridade eliminam combinações não desejadas de literais.

Dado um programa aberto, um estado para este programa é estabelecido introduzindo-se, para cada literal aberto, uma regra definindo seu valor verdade.

Definition 14 ***(Estado de Programa)*** *Um estado de programa de um dado programa aberto $\langle P_{Fix}, IC, OP\rangle$ é uma tupla $\langle P_{Fix}, P_{Var}, IC, O_P\rangle$. Para cada literal $L \in O_P$, a parte variável contém apenas uma das seguintes regras:*

1. $L \leftarrow \mathbf{t}$, *ou*
2. $L \leftarrow \mathbf{f}$, *ou*
3. $L \leftarrow \mathbf{u}$.

Interpretação	Parte Variável
$\{not\ L, not\ \neg L\}$	$\{L \leftarrow \mathbf{f}, \neg L \leftarrow \mathbf{f}\}$
$\{not\ L\}$	$\{L \leftarrow \mathbf{f}, \neg L \leftarrow \mathbf{u}\}$
$\{not\ \neg L\}$	$\{L \leftarrow \mathbf{u}, \neg L \leftarrow \mathbf{f}\}$
$\{\}$	$\{L \leftarrow \mathbf{u}, \neg L \leftarrow \mathbf{u}\}$
$\{not\ L, \neg L\}$	$\{L \leftarrow \mathbf{f}, \neg L \leftarrow \mathbf{t}\}$
$\{L, not\ \neg L\}$	$\{L \leftarrow \mathbf{t}, \neg L \leftarrow \mathbf{f}\}$

Tabela 2.2 — Relação entre Interpretação e Estado de Programa

Ainda, se $(L \leftarrow \mathbf{t}) \in P_{Var}$ *então* $(\neg L \leftarrow \mathbf{f}) \in P_{Var}$ *e vice-versa (se* $(L \leftarrow \mathbf{f}) \in P_{Var}$ *então* $(\neg L \leftarrow \mathbf{v}) \in P_{Var}$*). Nenhuma outra regra pertence a* P_{Var}*.*

Por definição, os símbolos **t** e *not* **f** pertencem a todos os modelos, e nem **u** nem *not* **u** pertencem a qualquer modelo.

Example 8 *(extraído de [ALF 95]) O programa* P_8 *abaixo*

```
a   ←  not c.     c  ←  not d.
¬a  ←  not b.     d.
```

é contraditório, pois possui no modelo tanto ¬a (pois not b é verdadeiro, visto que não há regras para b) e a (pois d é verdadeiro, logo not d é falso e c é falso, logo not c é verdadeiro e a é verdadeiro). Para revisá-lo, inicialmente é preciso caracterizá-lo pelo estado $s = \langle P_{Fix}, P_{Var}, IC, O_P \rangle$ *onde*

$$\begin{array}{lcl} P_{Fix} & = & \{a \leftarrow not\ c;\ \neg a \leftarrow not\ b;\ c \leftarrow not\ d\} \\ P_{Var} & = & \{b \leftarrow \mathbf{f};\ \neg b \leftarrow \mathbf{f};\ d \leftarrow \mathbf{t};\ \neg d \leftarrow \mathbf{f}\} \\ IC & = & \{f \Leftarrow L, \neg L / L \in \mathcal{H}\} \\ O_P & = & \{b, \neg b, d, \neg d\} \end{array}$$

Para remover contradições, as alterações são feitas nas regras em P_{Var}*.*

É importante salientar que a componente variável do estado determina uma interpretação, e vice-versa, como mostrado na tabela 2.2 abaixo.

O princípio da coerência é garantido pelas restrições impostas pela definição de P_{Var}. O modelo de um programa aberto $P = \langle P_{Fix}, IC, O_P \rangle$ determinado por um estado associado $s = \langle P_{Fix}, P_{Var}, IC, O_P \rangle$ é dado pelo modelo bem-fundado $WFM_P(P_{Fix} \cup P_{Var})$ e é denotado por $\mathcal{M}(s)$. Daí sai a noção de estado de programa contraditório. Além do modelo, cada estado tem associado a sí também a noção de *crenças básicas*[8] $\mathcal{B}(s)$, definido por $\mathcal{B}(s) = \mathcal{M}(s) \cap \mathcal{H}(P_{Var})$.

[8]Esta noção de *crenças básicas* não tem nenhuma relação com a noção de *crenças* do modelo BDI.

Definition 15 ***(Estado de Programa Contraditório)*** *Um* estado de programa $s = \langle P_{Fix}, P_{Var}, IC, O_P \rangle$ *é* contraditório *sse* $P_{Fix} \cup P_{Var}$ *é contraditório ou se existe pelo menos uma restrição de integridade em s que seja violada por* $\mathcal{M}(s)$*. Caso contrário, é não-contraditório.*

Novamente, como já foi referido, a condição "$P_{Fix} \cup P_{Var}$ é contraditório" pode ser desconsiderada se for acrescido ao conjunto IC restrições na forma $\mathbf{f} \Leftarrow L, \neg L$ para todos os literais L.

Uma vez que um estado é contraditório, é preciso revisá-lo para eliminar as contradições. Um estado contraditório corresponde a uma situação onde as crenças básicas são não satisfatórias. Nesta situação, a idéia intuitiva é alterar aqueles literais revisíveis de forma a atingir um novo estado não-contraditório. Esta alteração não deve ser arbitrária, mas deve ser feita de modo a realizar o mínimo de modificações possíveis. Assim, o novo estado de programa atingido deve ser tão próximo quanto possível do estado de origem. Isto é feito pela definição de *diferença entre interpretações* e por uma *relação de proximidade* entre interpretações.

A abordagem adotada é que se um literal não é indefinido, então é preferível revisá-lo primeiramente para indefinido antes de passá-lo para o valor verdade oposto; se ele for indefinido, então a sua revisão é feita ou para verdadeiro ou para falso. A motivação para esta escolha vem das ordens existentes entre os valores verdade. Numa ordem de acordo com o valor verdade, tem-se que $\mathbf{f} < \mathbf{u} < \mathbf{t}$; já numa ordem de acordo com a "quantidade de conhecimento" tem-se que $\mathbf{u} < \mathbf{f}$ e $\mathbf{u} < \mathbf{t}$ (uma ordem parcial, onde $\mathbf{f}$ e $\mathbf{t}$ não se relacionam). Em ambos os casos, passar de verdadeiro para falso e vice-versa sempre ocorre via o valor indefinido. As definições a seguir capturam esta idéia.

Definition 16 ***(Diferença entre Interpretações)*** *Seja* $\mathcal{I}$ *o conjunto de todos os conjuntos coerentes de literais com respeito a uma linguagem finita, e* $I_1, I_2 \in \mathcal{I}$ *com* $I_1 = T_1 \cup not\ F_1$ *e* $I_2 = T_2 \cup not\ F_2$*. A* diferença entre as interpretações I_1 *e* I_2 *é definida por*

$$Diff(I_1, I_2) = \begin{cases} (F_2 - F_1) \times \{\mathbf{f}\} \cup \\ (F_1 - F_2) \times \{\mathbf{u}\} \cup \\ (T_1 - T_2) \times \{\mathbf{u}\} \cup \\ (T_2 - T_1) \times \{\mathbf{t}\} \end{cases}$$

A diferença entre interpretações registra as mudanças que ocorrem, quando de passa de uma interpretação I_1 para outra I_2. Assim:

- os literais que são falsos em I_2 e não são falsos em I_1 (pertencem a F_2, mas não a F_1) são rotulados com $\mathbf{f}$ (ou seja, de uma interpretação I_1 para a outra I_2 passaram a ser falsos);

- os literais que são falsos em I_1 e não são falsos em I_2 (pertencem a F_1, mas não a F_2) são rotulados com **u** (ou seja, de uma interpretação I_1 para a outra I_2 passaram a ser indefinidos);
- os literais que são verdadeiros em I_1 e não são verdadeiros em I_2 (pertencem a T_1, mas não a T_2) são rotulados com **u** (ou seja, de uma interpretação I_1 para a outra I_2 passaram a ser indefinidos);
- os literais que são verdadeiros em I_2 e não são verdadeiros em I_1 (pertencem a T_2, mas não a T_1) são rotulados com **t** (ou seja, de uma interpretação I_1 para a outra I_2 passaram a ser verdadeiros).

Alguns destes conjuntos podem não ser disjuntos. Por exemplo:

Example 9 *(extraído de [ALF 95]) Sejam $I1 = \{a\} \cup not\{b, c, d\}$ e $I2 = \{b, e\} \cup not\{d, f\}$. A diferença entre estas duas interpretações é dada por*

$$Diff(I_1, I_2) = \{(a, \mathbf{u}), (b, \mathbf{t}), (b, \mathbf{u}), (c, \mathbf{u}), (e, \mathbf{t}), (f, \mathbf{f})\}$$

Neste caso, b era falso em I_1 e passou a verdadeiro em I_2. A noção de diferença captura esta passagem pois possui tanto o par $(b, \mathbf{u})$ como o par $(b, \mathbf{t})$.

A relação de proximidade é definidia entre interpretações, e não entre estados de programas. Como foi referido anteriormente, a partir destes estados duas interpretações diferentes podem ser consideradas: o modelo determinado pelo estado do programa — $\mathcal{M}(s)$ — ou o seu conjunto de crenças básicas — $\mathcal{B}(s)$. Uma vez que se parte do princípio que as revisões devem ser feitas sobre as crenças básicas (que derivam da parte variável do estado do programa), é esta a interpretação adotada para definir a noção de revisão de um estado de programa.

Definition 17 ***(Relação de Proximidade)*** *Seja $\mathcal{I}$ o conjunto de todas as interpretações com respeito a uma linguagem finita, e $M, A, B \in \mathcal{I}$. Uma interpretação A é mais próxima a M do que B sse $Diff(M, A) \subset Diff(M, B)$.*

Quando em face de contradições, a revisão deve alterar minimamente as crenças básicas.

Definition 18 ***(Revisão de um Estado de Programa)*** *Seja s o estado de programa inicial de um programa aberto P, e S_P o conjunto de estados de programas não-contraditórios. Uma revisão de s é um elemento $t \in S_P$ tal que*

$$\forall r \in S_P . Diff(B(s), B(r)) \subseteq Diff(B(s), B(t)) \Rightarrow r = t.$$

São revisões de s aqueles estados de programa que tem sua parte variável alterada minimamente em relação a parte variável de s. Ou seja, as revisões que são feitas a partir de s são minimais.

Example 10 *(extraído de [ALF 95]) Considere as seguintes interpretações, que correspondem as partes variáveis de 4 estados de programas* s_0,s_1,s_2 *e* s_3*:*

$$\begin{array}{rclrcl} I_0 & = & \{a\} \cup not\ \{\neg a, b, \neg c\} & I_1 & = & \{a, c\} \cup not\ \{\neg a, b, \neg c\} \\ I_2 & = & \{a, d\} \cup not\ \{\neg a, \neg c, \neg d\} & I_3 & = & \{a, c, d\} \cup not\ \{\neg a, \neg c, \neg d\} \end{array}$$

Supondo que s_0 *é o estado inicial (contraditório) e que* s_1*,* s_2 *e* s_3 *sejam os únicos estados não-contraditórios possíveis a partir de modificações nas crenças básicas de* s_0*. Dadas as interpretações acima, tem-se as seguintes diferenças:*

$$\begin{array}{rcl} Diff(I_0, I_1) & = & \{(c, \mathbf{t})\} \\ Diff(I_0, I_2) & = & \{(b, \mathbf{u}), (d, \mathbf{t}), (\neg d, \mathbf{f})\} \\ Diff(I_0, I_3) & = & \{(b, \mathbf{u}), (c, \mathbf{t}), (d, \mathbf{t}), (\neg d, \mathbf{f})\} \end{array}$$

Logo, as revisões possíveis para o estado inicial s_0 *são* $\{s_1, s_2\}$ *pois* $Diff(I_0, I_1) \subseteq Diff(I_0, I_3)$*,* $Diff(I_0, I_2) \subseteq Diff(I_0, I_3)$*; e* $Diff(I_0, I_1)$ *e* $Diff(I_0, I_2)$ *não estão contidos nem contém uma a outra.*

Observe que a diferença de s_3 para s_0 mostra que há modificações desnecessária e que, apesar de removerem a contradição, não são minimais.

Retomando o exemplo 8 anterior, temos:

Example 11 *(extraído por [ALF 95]) As revisões do estado s são os dois estados resultantes substituindo-se, em s,* P_{Var} *por* P^1_{Var} *ou* P^2_{Var} *abaixo:*

$$\begin{array}{rcl} P^1_{Var} & = & \{\mathtt{b} \leftarrow \mathbf{u}; \neg\mathtt{b} \leftarrow \mathbf{f}; \mathtt{d} \leftarrow \mathbf{t}; \neg\mathtt{d} \leftarrow \mathbf{f}\} \\ P^2_{Var} & = & \{\mathtt{b} \leftarrow \mathbf{f}; \neg\mathtt{b} \leftarrow \mathbf{f}; \mathtt{d} \leftarrow \mathbf{u}; \neg\mathtt{d} \leftarrow \mathbf{f}\} \end{array}$$

Ou seja, a contradição é removida tornando-se b ou d indefinidos.

A capacidade de representar informação negativa, juntamente com um procedimento bem-definido para restaurar a consistência de programas em lógica faz com que a programação em lógica extendida seja adequada para representar e realizar diferentes formas de *raciocínio não-monotônico*[ALF 96]. Em particular, interessam-nos duas formas de raciocínio não-monotônico, nomeadamente *raciocínio revogável* e *raciocínio abdutivo.*

2.2.4 Raciocínio Revogável

Uma *regra revogável* é uma regra na forma *Normalmente se A então B.* A programação em lógica estendida permite que se represente regras revogáveis e atribui um significado a um conjunto de regras (revogáveis ou não) quando a aplicação de uma destas regras revogáveis dá origem a contradições.

Example 12 *Considere as cláusulas abaixo (extraídas de [ALF 96]):*

1. Normalmente aves voam..

2. Pinguins não voam..

3. Pinguins são aves.

4. c é um pinguin.

representadas pelo programa P_{12} abaixo:

```
v(X)    ←  a(X), not ab(X).    (i)
¬v(X)   ←  p(X).               (ii)
a(X)    ←  p(X).               (iii)
p(c).                          (iv)
```

Visto que não existem regras para $ab(c)$, *not* $ab(c)$ é verdadeiro e deriva $v(c)$. Ainda, $\neg v(c)$ é derivado a partir de $p(c)$, na segunda regra. Logo, o programa é contraditório. Neste caso, a regra (i) é revogável e dá origem a uma contradição devido a pressuposição de que $ab(c)$ é falso. Assim, $v(c)$ não deve ser obtido como uma conclusão e o significado pretendido para P requer $\neg v(c)$ and *not* $v(c)$ (por coerência) sejam verdadeiros.

Considerando o programa aberto definido a partir de P_{12} com

Se se revisa P_{12} com um conjunto de revisíveis $R = \{ab(X)\}$, obtêm-se uma única revisão $Rev = \{ab(a) \leftarrow \mathbf{t}\}$. A semântica do programa revisado obtido a partir desta revisão é

$$\{p(c), not\ \neg p(c), a(c), not\ \neg a(c), \neg v(c), not\ v(c), not\ \neg ab(c)\}$$

como era desejado.

Neste trabalho, o reciocínio revogável é utilizado para se obter um subconjunto consistente dos desejos dos agentes que constituirão sua intenções.

2.2.5 Raciocínio Abdutivo

O *raciocínio abdutivo* consiste em, dada uma teoria T e um conjunto de observações O, encontrar-se uma teoria Δ tal que $T \cup \Delta \models O$ e $T \cup \Delta$ seja consistente.

No contexto da programação em lógica extendida, um "framework" abdutivo P' é uma tupla $\langle P, Abd, IC \rangle$, onde P é um ELP, Abd é um conjunto de literais abduzíveis e IC é um conjunto de restrições de integridade. Uma observação O possui uma explicação abdutiva Δ se e somente se $P' \cup \Delta \models_P O$ e $P \not\models_P \Delta$. é possível abduzir-se uma teoria Δ que explica tais observações tornando-se cada uma destas observações O uma nova restrição de integridade $O \Leftarrow$ e revisando-se o programa com um conjunto de revisíveis Abd (é importante lembrar que um programa é contraditório se e somente se para algum literal L em P, $P \models L$ e $P \models \neg L$, ou se *uma restrição não é satisfeita*).

Example 13 *Considere as cláusulas a seguir:*

1. *Normalmente aves voam.*

2. *Pinguins não voam.*

3. *Pinguins são aves.*

4. *Normalmente tigres não voam.*

5. *Normalmente leão são mamíferos.*

   representado pelo programa P_{13} abaixo:

$$\begin{array}{rcl} v(X) & \leftarrow & a(X), not\ ab1(X) \\ \neg v(X) & \leftarrow & p(X) \\ a(X) & \leftarrow & p(X) \\ \neg v(X) & \leftarrow & t(X), not\ ab2(X) \\ m(X) & \leftarrow & l(X), not\ ab3(X) \end{array}$$

Existe uma explicação possível para a observação $\neg f(c)$?

O "framework" abdutivo derivado do programa P_{13} e observação $\neg f(a)$ é $P' = \langle P_{13}, \{t(X), p(X), l(X)\}, \{\neg v(a) \Leftarrow\}\rangle$. Visto que não existem regras para $p(c)$ e $t(c)$, P não deriva $\neg v(a)$. Portanto, a restrição não é satisfeita e o programa é contraditório. Para um conjunto de revisíveis Abd, existem duas revisões que restauram a consistência em P', notadamente $Rev_1 = \{t(a)\}$ e $Rev_2 = \{p(a)\}$. Estas são as duas explicações possíveis para $\neg v(a)$.

Neste trabalho, o raciocínio abdutivo é utilizado para determinar se um agente acredita que há um possível curso de ações para satisfazer um desejo, bem como no tratamento do planejamento.

2.2.6 Revisões Preferidas

Em algumas ocasiões, será necessário definir-se que não se deseja obter, durante o processo de revisão, as revisões mínimas fornecidas pelo formalismo, mas que prefere-se obter revisões que tenham um certo literal apenas se não houver nenhuma outra revisão incluindo outros literais. O formalismo que está sendo utilizado neste trabalho permite que se defina preferências sobre a ordem das revisões grafo e/ou rotulado direto acíclico, definido pelas regras na forma[DAM 94]:

$$Nível_0 \ll Nível_1 \wedge Nível_2 \wedge \ldots Nível_n (\text{n} \geq 1)$$

Nodos $Nível_i$ são identificadores de níveis de preferência. A cada um destes níveis está associado um conjunto de revisíveis denotado por $\mathcal{R}(Nível_i)$. Regras como esta para o $Nível_0$ definem que se deseja considerar revisões para o $Nível_0$ somente se, para algum corpo da regra, seus níveis forem considerados e não existirem revisões em qualquer um destes níveis. A raíz do grafo de preferências é o nodo denotado por **bottom**.

Se ao exemplo 13 associar-se o seguinte grafo de preferências:

$$\begin{array}{lll} 1 \ll bottom & & 2 \ll 1 \\ \mathcal{R}(\mathbf{bottom}) = \{t(X)\} & & \\ \mathcal{R}(1) = \{p(X)\} & \mathcal{R}(2) = \{l(X)\} & \end{array}$$

então, ao invés de duas revisões naquela abdução, ter-se-ia somente $Rev_1 = \{t(a)\}$, visto que revisões para $p(c)$ (no nível 1) somente seriam consideradas se não existissem revisões para os níveis mais baixos ($t(c)$ no **bottom**, neste caso).

Neste trabalho, usa-se as preferências para estabelecer a ordem de escolha dos subconjuntos dos desejos a serem adotados como intenções.

De posse desta maquinaria lógica, é agora possível definir-se o modelo de agentes, como será visto no capítulo a seguir.

3 SBIA95 - “Modelling Intentions with Extended Logic Programming”

Este artigo[MÓR 95], o primeiro escrito durante o doutorado, apresenta as idéias iniciais da teoria que viria a ser desenvolvida, notadamente:

- parte da revisão bibliográfica que embasa o trabalho, notadamente as teorias de agentes existentes, com uma perspectiva orientado ao agente e baseadas em lógicas não-tratáveis computacionalmente que apresentam uma série de problemas na modelagem dos agentes;
- a mudança de perspectiva levantava novas questões não tratadas pelas teorias existentes.

Após esta exposição, o artigo apresenta um modelo preliminar, já utilizando programação em lógica, das intenções em um agente proposto no trabalho. Apenas propriedades estáticas das intenções são descritas neste ponto. O trabalho aqui descrito apresenta as definições iniciais do modelo formal do agente proposto na tese.

Modeling Intentions with Extended Logic Programming

Michael da Costa Móra[1]* and José Gabriel Lopes[2]** and Helder Coelho[3]

[1] CRIA UNINOVA/Portugal - CPGCC UFRGS/Brazil
2825 Monte da Caparica, Portugal
mdm@fct.unl.pt
[2] CRIA UNINOVA/Portugal
2825 Monte da Caparica, Portugal
gpl@fct.unl.pt
[3] INESC/Portugal
R. Alves Redol, 7, 1000 - Lisbon
hcoelho@eniac.inesc.pt

Abstract. As far as we are concerned, the existing theories of intentions adopt a designer's perspective, i.e., intentions and related mental states are approached from a designer's point-of-view. Such theories are logic-based and enable one to reason about the agents. However, they are not adequate to be used by the agent, to reason about itself. We argue that adopting an agent perspective may simplify some aspects of such theories, and enable the definition of a logic that agents may use to reason. Also, this change of perspective raises some questions that are, in general, ignored by traditional theories, namely the relations among other attitudes and intentions and between intentions and actions. We present preliminary definitions of a theory of intentions that adopt this different perspective. We also discuss some of the traditionally ignored questions, unveiling potential extensions to the definitions presented here.

Keywords: theories of intentions; mental states modeling; event calculus; distributed artificial intelligence.

1 Introduction

Intentions, among all the mental states that characterize an intelligent agent, are considered fundamental building blocks, as they cannot be reduced to other more basic mental states. Since agents are resource bounded, it is not possible for them to continuously weigh their competing motives and beliefs. Intentions play

* Supported by CNPq/Brazil, contract number 200636/93-6(NV).
** Work supported by project IDILIO, contract number STRDA-C-TIT-139/92, funded by JNICT.

their role as a compromise with a specific possible future which, once chosen, is abandoned only in certain circumstances, thus avoiding such continuous weighing of other attitudes. According to Bratman [2], intentions play three functional roles in a rational agent: (1) intentions pose problems for the agent, who needs to determine ways to overcome them; (2) intentions constitute a *screen of admissibility* for adopting other intentions, i.e., an agent cannot adopt an intention which conflicts with previously adopted ones; and (3) the agent must track the success of its intentions and persist on its execution, in case of failure. Following Bratman's analysis, many formal theories of rational agency[1] have appeared [4] [10] [6] [13]. Such theories are meant to be formal specifications of a rational agent in terms of its mental states and the relations among them. However, none of these theories are suitable to be used by the agent as a formal basis for it to reason. According to our view, while allowing agent designers to reason about properties of agents, such theories of intentions ought to be expressed in a logic the agent may use to reason, in order to decrease the gap between specifications and real agents. In this paper, we make the preliminary definitions of a theory of intentions that adopts an *agent perspective*, instead of a designer's one. As the underlying logic, we use Logic Programming extended with explicit negation (or ELP – Extended Logic Programming) with its semantics being given by the paraconsistent version of the Well-Founded Semantics augmented with Explicit Negation (WFSX) [1].

Intentions are temporal attitudes, as they guide future planning activities and track present actions. Therefore, we need some formalism to deal with time. Most of the existing theories use a variation of dynamic logic [Eme 80], defined in their modal logic environment by the standard modal operators. However, in order to allow the previously mentioned change of perspective, we need a formalism with which the agent may reason, as discussed before. So, on top of the logical basis provided by ELP, we define a variation of the Event Calculus (EC) [7], inspired on previous extensions by Messiaen [8] and Shanaham [11]. The ELP, together with the EC, constitute the logical framework which will be used for building our theory of intentions.

This paper is organized as follows: in section 2, we present ELP and its paraconsistent WFSX semantics; in section 3, we define the EC and justify our choice; in section 4, we present the preliminary definitions of our theory of intentions; finally, in sections 5 and 6, we discuss some related questions and draw some conclusions.

[1] Also, many non-formal models of intentions, related some way to Bratman's ideas, have appeared. See, for instance, [**?**] [14].

2 Extended Logic Programs

An extended logic program is a set of rules of the form $L \leftarrow L_1, \ldots, L_n, not\ \ L_{n+1}, \ldots, not\ \ L_m$ $(0 \leq n \leq m)$ where L and L_i are objective literals. An objective literal is either an atom A or its explicit negation $\neg A$. $not\ L$ is a default literal, where *not* stands for negation as failure. Literals are either objective or default literals. By $not\ \{a_1, \ldots, a_n, \ldots\}$ we mean $\{not\ a_1, \ldots, not\ a_n, \ldots\}$. $\mathcal{H}(\mathcal{P})$ denotes the set of all ground objective literals of a program P and is called extended Herbrand base of P. We present here the definition of the paraconsistent version of WFSX, that allows for contradictory programs.

Definition 1 ((Interpretation)). An interpretation I of an extended logic program P is any set $T \cup not\ F$, where T and F are subsets of $\mathcal{H}(\mathcal{P})$ which verify the coherence principle: if $\neg L \in T$ then $L \in F$. Set T contains all *true* ground objective literals in I, set F contains all $false$ ground objective literals in I. The truth value of the remaining objective literals is $undefined$ (the truth value of a default literal $not\ L$ is the 3-valued complement of L).

Through some program transformations, which produce (possibly more than one) non-negative programs, it is possible to get all the consequences of the original program, even those leading to contradictions, as well as those arising from contradictions. It can be shown that the operator that is used to obtain such a set of consequences (name Φ_P) is monotonic under set inclusion of interpretations, for any program P[2]. Hence, it has a least fixpoint, which can be obtained by iterating Φ_P starting from the empty set.

Definition 2 ((Paraconsistent WFSX)). The paraconsistent WFSX of an extended logic program P, denoted by $WFSX(P)$, is the least fixpoint of Φ applied to P. If some literal L belongs to the paraconsistent WFSX of P we write $P \models_p L$. A program P is said to be *contradictory* iff, for some positive literal L, both $P \models_p L$ and $P \models_p \neg L$ hold. P is said to be consistent iff it is not contradictory.

ELP with the WFSX semantics presents some characteristics that are very useful when we are to model intentions and related mental

[2] Due to lack of space, the formal definitions of such operators are not presented here. For details, refer to [1].

states, namely its ability to deal with contradictory programs and to allow the detection of such contradiction, through the notion of contradictory program. Also, this concept is based on the definition of $\models_p$, which is computable and may be used not only by the designer, when (s)he is proving properties, but also by the agent, when it is reasoning.

3 The Event Calculus

In order to represent and reason about action and time, we use the Event Calculus. Initially proposed by Kowalski and Sergot [7] to overcome some drawbacks of the Situation Calculus, this formalism has inspired many variations. Here, We present a definition of the Event Calculus derived from the ones proposed by [8] and [11]. Although we do not make extensive use of the EC in this paper, it is necessary when we deal with topics related to action planning (see section 5). It is presented here because our purpose is to settle the logical framework used to develop our theory of intentions.

As in [7], the ontological primitives are *events*, which initiate and terminate periods during which *properties* hold. We use the ELP language to define them. Later on, we will extend this basic ontology to deal with concurrent actions. The core clauses of our EC are:

$$\begin{aligned}
holds\_at(Prop, T) &\leftarrow initially(Prop), persists(0, Prop, T). && (1)\\
holds\_at(Prop, T) &\leftarrow happens(Evt, Act, TEvt), initiates(Evt, Prop),\\
&\qquad TEvt < T, persists(TEvt, Prop, T).\\
persists(TEvt, Prop, T) &\leftarrow not\ clipped(TEvt, Prop, T). && (2)\\
clipped(TEvt, Prop, T) &\leftarrow happens(IntEvt, Act, TIntEvt), && (3)\\
&\qquad terminates(IntEvt, Prop), not\ out(TIntEvt, TEvt, T).\\
out(TIntEvt, TEvt, T) &\leftarrow (T \leq TIntEvt); (TIntEvt < TEvt). && (4)
\end{aligned}$$

The predicate *holds_at*(P, T) represents that property P holds at time T. The predicate *happens*(E, A, T) represents that an uniquely identified event E has occured or will occur, at time T. This event denotes a specific instance of action A. By now, time points are to be interpreted as natural numbers, being 0 the initial time point. We assume the usual definition for the relations $\neq$, $<$ and $\leq$. Predicates *initiates*(E, P) and *terminates*(E, P) indicate, respectively, that an event E initiates or terminates the interval during which property P holds. Predicate *persists*(TE, P, T) represents that a property P holds in an interval between TE and T if that interval is not interrupted (i.e., *clipped*) by an event which terminates the holding property. Predicate *initially*(P) states that property P holds from "the beginning of times" (there is no event that initiates it). Predicates *initiates*/2 and *terminates*/2 capture a problem domain. Predicate *happens*/3 will describe courses of events, meaning that action A is performed when event E occurs.

3.1 Extending the EC to Deal with Concurrent Actions

We need concurrent actions because we are setting the logical foundations for a theory of intentions, and at some moment it will be necessary to model an agent that interacts with some other agent. So there may be situations where their joint action will be required. In this formulation of the EC it is easy to represent concurrent occurrences of actions, as suggested in [11]: it is enough to state that they happen at the same time point, such as $\{happens(e1, A1, x)., happens(e2, A2, x).\}$. Problems arise when the concurrent execution of such actions produce results different from those expected if the actions were performed independently. In order to overcome this problem, we need to modify the EC formulation in two aspects: (1) it is necessary to state the properties that the concurrent execution of two or more actions initiates/terminates, and how to represent such a concurrent occurrence, as well; (2) we need some way to express that an event can cancel the effects of another one occurring at the same time. For instance, consider the supermarket trolley problem [11]. If one *push*es a supermarket trolley, the effect of such an action is to move the trolley forward; if one *pull*s such a trolley, the effect is to move it backward. But, if one *push*es and *pull*s it at the same time, the trolley will spin around, or stay still. Thus, in this kind of situation, we should be able to state that the concurrent occurrence of *push* and *pull* on the same object produces a different effect, and does not necessarily initiate/terminate their usual properties.

In order to solve problem (1), we introduce a new term *conc*/2, which relates 2 types of events that, together, form a new one, along with a new axiom for the predicate *happens*, which we will call, from now on, *occurs*, to avoid some looping problems[3]. The *compound event* $conc(E1, E2)$ happens whenever $E1$ and $E2$ happen concurrently.

$$\begin{aligned} occurs(conc(Evt1, Evt2), TConc) \leftarrow\ & happens(Evt1, A1, TConc), \\ & happens(Evt2, A2, TConc), Evt1 \neq Evt2. \\ occurs(Evt, TEvt) \leftarrow\ & happens(Evt, A, TEvt). \end{aligned} \tag{5}$$

It is also necessary to represent, through *initiates*/2 and *terminates*/2, the properties that are affected by compound events. To overcome problem (2), we introduce a new predicate *cancels*/2. A formula $cancels(Act1, Act2)$ states that if an action of type $Act1$ occurs, it cancels the effects of an action of type $Act2$ occurring at the same time. It is also necessary to modify some of the EC axioms, namely axiom 3.

$$\begin{aligned} clipped(TE, P, T) \leftarrow\ & occurs(C, TC), terminates(C, P), \\ & not\ cancelled(C, TC), not\ out(TC, TE, T). \end{aligned} \tag{6}$$

$$\begin{aligned} cancelled(Evt, T) \leftarrow\ & happens(Evt2, Act2, T), \\ & happens(Evt, Act, T), cancels(Act2, Act). \end{aligned} \tag{7}$$

[3] The predicate *occurs*/2 will be true if a simple event or a compound event happens.

Note that, if we do not introduce the *cancels* predicate and the above modifications, we may get incorrect conclusions, since the effects brought about by the events that form the compound event would still be caused.

Example 3. Consider the following representation of the trolley car problem [11]

$$\begin{aligned}
initiates(conc(E1,E2), spinning) &\leftarrow happens(E1, push, \_), happens(E2, pull, \_).\\
terminates(conc(E1,E2), forward) &\leftarrow happens(E1, push, \_), happens(E2, pull, \_).\\
terminates(conc(E1,E2), backward) &\leftarrow happens(E1, push, \_), happens(E2, pull, \_).\\
initiates(E, forward) &\leftarrow happens(E, push, T).\\
terminates(E, backward) &\leftarrow happens(E, push, T).\\
terminates(E, spinning) &\leftarrow happens(E, push, T).\\
initiates(E, backward) &\leftarrow happens(E, pull, T).\\
terminates(E, forward) &\leftarrow happens(E, pull, T).\\
terminates(E, spinning) &\leftarrow happens(E, pull, T).
\end{aligned}$$

and the following possible course of events

$$\begin{array}{l}
happens(e1, push, 2).\quad happens(e2, pull, 5).\\
happens(e3, pull, 15).\ happens(e4, push, 15).
\end{array}$$

No predicate $holds\_at(P,T)$, when $T = 20$, belong to the model. Informally, this happens because the effects of *push* and *pull* are still individually considered, and since $clipped/2$ tests for events that terminate properties, each one is terminated by the other. If we add

$$cancels(push, pull).\ cancels(pull, push).$$

then we will have $\{holds\_at(spinning, 20)\}$, as desired, since the effects of *pull* and *push* are not individually accounted. After these modifications, the final EC axioms are the basic ones, (1) – (4) extended with axioms (5) – (7)[4].

4 Towards a Theory of Intentions

A theory of intentions is meant to formalize the properties of intentions and its relation to other mental states. Following Bratman's analysis [2] of what intentions and their roles are, we may point some of the desired properties of intentions. Intentions must be mutually consistent, i.e., an agent is not supposed to adopt intentions that preclude each other. Exactly what to do when an agent is faced with a new intention that is contradictory with a previously existing one is yet to be decided. Cohen and Levesque [4] suggest that the newly arriving intention should be discarded if it introduces a contradiction, although

[4] It would also be necessary to introduce a standard event *start*, which happens at the initial time 0; and the definition of $\neg hold\_at(P,T)$. Notice that we are not (yet) benefiting from the ELP to define the EC. Its properties are used only when the mental states are defined.

they do characterize such a behavior as fanatical[5]. Intentions are believed to be satisfiable or, at least, they are not believed to be impossible. It is not rational that an agent intends something it believes to be impossible, since it must try to overcome the problems posed by its own intentions. It is, indeed, rational to intend something that we do not know if it is possible or not. Agents do not need to intend all the consequences of their intentions, even those that they are able to foresee – the so-called *side-effect* problem[6][10]. For example, an agent who owns an out-of-order car will formulate (suppose) an intention to repair it, and will decide to take the car to a mechanic. As a consequence, it will have to spend some money to pay the service. Although spending money is a consequence of his intention to repair the car, he probably does not intend to spend money. Suppose that, when it arrives at the garage, it is told that they are not able to fix the car. Since the agent's intention is not satisfied, it would take the it to another garage (remember that an agent must track the success of the accomplishment of his intentions, and persist acting in order to achieve them). But now, suppose that our agent is told that, since he is the $1{,}000{,}000^{th}$ client, the repair will be for free. Although spending money was a foreseen consequence of his intentions, it was not an intention by itself. So, the agent may (and certainly will) accept the free repair. In summary, the side-effect (spending money for repairing the car) was not intended, although it was *intentionally chosen*[7]. An agent must commit to its intentions, i.e., an agent who has an intention must persist in trying to achieve it through out changing circumstances. This is a delicate question: the notion of commitment is at the core of the idea of intentions, and it is certainly necessary, since agents are resource bounded and cannot continuously weigh their competing motives and beliefs[9]. Also, it is very useful to allow agents to infer other agent's intentions, to coordinate their actions more easily. However, the degree of such commitment varies according to many factors, namely changes in beliefs, changes in motives, action's priorities and so. For instance, an agent cannot persist forever on an intention, since this intention could no longer be compatible with its motives and beliefs, what would lead to quite irrational behaviors. As we will see further, in section 5, commitment may be achieved through adequate policies with respect to adoption and abandonment of intentions.

Some work has been devoted to theories of intentions (or theories of rational agency). Cohen and Levesque [4] were the first ones to develop a formal theory of intentions, based on Bratman's analysis. Although it has been used as a starting point for a very fruitful work ([5][3], among others), it does not meet some of the desired properties we have presented, namely it suffers from the side-effect problem. Also, its notion of commitment is deeply connected to the semantics of intentions, preventing its adequate treatment [12]. Konolige and Pollack [6]

[5] Later, in the same work [4], they consider relativizing intentions to its cause, an arbitrary condition. But, it has been argued that this approach contradicts Bratman's analysis [12].

[6] Side-effects are the non-intended consequences of one's intentions.

[7] As Bratman states, intentions are a subset of the agent's choices. See [2] for a more detailed discussion.

solve some of these problems presenting a formalism also based in modal logics but with a different semantics (not based in the accessibility relation among possible worlds). However, as the authors state, it is limited to static aspects of intentions, not encompassing questions such as adoption and abandonment of intentions. In common, those theories share a *designer's perspective*[8], i.e., they are conceived to be used by agent designers to prove properties about them. We argue that adopting an *agent's perspective*, where the theory is to be used by the agent when reasoning about the world (i.e, the agent builds the world model while reasoning) is more adequate: it simplifies some traditional problems, namely the side-effect problem, and induces us to deal with some questions that are, in general, ignored by traditional theories. We present here the preliminary definitions of such a theory, up to the point where these "ignored question" emerge[9].

Definition 4 ((Agent Structure)). An agent structure is a tuple $< \mathcal{B}, \mathcal{I}, \mathcal{A}, \mathcal{T} >$ such that (i) $\mathcal{B}$ is a set of extended logic programming sentences[10] – it is the set of agent's beliefs; (ii) $\mathcal{I}$ is a not deductively closed set of extended logic programming sentences – it is the set of agent's intentions; (iii) $\mathcal{A}$ is a set of *initiates*/2 and *terminates*/2 clauses – it is the description of the actions the agent is capable of performing; (iv) $\mathcal{T}$ is the set of EC axioms – the time axioms.

This agent structure is the core definition of an agent, in our theory. It is not an architecture, in the sense of [**?**], since it defines only some of the agent's mental states and the relation among them, and not the whole agent. But, unlike most of the existing theoretical approaches, this theory supplies, additionally to those specifications, a logical framework that enables the agent to reason and behave. By the above definition, an agent $\mathcal{G}$, with an agent structure $\mathcal{G}_{\mathcal{E}} = < \mathcal{B}, \mathcal{I}, \mathcal{A}, \mathcal{T} >$ *believes* in a sentence b iff $b \in \mathcal{B}$ and *intends* a sentence i iff $i \in \mathcal{I}$. The side-effect problem is avoided, since the only intentions an agent has are those belonging to $\mathcal{I}$ (remember that it is not closed under deduction). Its logical consequences, both with respect to other intentions and with respect to its beliefs, are excluded.

According to the language definition presented so far (ELP + EC), an intention will be one of the three different kinds of sentences: (i) $happens(Evt, Act, Time)$ which states that an agent intends an action Act to be performed at time $Time$; (ii) $holds\_at(Prop, Time)$ which states that an agent intends to be in a world state where property $Prop$ holds, at time $Time$; (iii) *time independent sentences* properties, that are meant to be maintained, as long as they are valid intentions. These are called *maintenance intentions* [4]. New intentions cannot contradict them and modifications in the agent's beliefs, stating that such property does not hold anymore, should make the agent act on that intention[11].

[8] This term, as long as the term *agent's perspective*, is borrowed from [13].
[9] We shall discuss them in section 5.
[10] Whether this set is closed under deduction or not depends on the approach we use to deal with beliefs. In general, it is not necessarily closed under deduction.
[11] This kind of intention is out of our scope, in this work.

Example 5. Suppose the car repair problem described in section 4. The agents beliefs, action description and intentions would be, respectively $\mathcal{B} = \{holds\_at(bad\_car, t1)\}$, $\mathcal{A} = \{initiates(spend\_money, E) \leftarrow act(E, take\_car\_mec)\}$, $\mathcal{I} = \{holds\_at(car\_is\_repaired, t2), happens(e1, take\_car\_mec, t3)\}$.

According to the definitions, logical consequences of the intentions are not intentions by themselves. So, although we are able to derive $holds\_at(spend\_money, t4)$, in any $t4 > t3$, from $\mathcal{B} \cup \mathcal{I} \cup \mathcal{A} \cup \mathcal{T}$, it does not belong to $\mathcal{I}$, so it is not an intention, and the agent does not need (in fact, he will not) to persist on it.

The definition of agent structure is still too generic, and does not assure that intentions will have the desired properties, as stated in section 4. Some restrictions must apply. Thus, we extended our definition to *rational agent structure*.

Definition 6 ((Rational Agent Structure)). Let *now* be a function of type $now :\rightarrow \mathcal{N}$ that returns the current time point. A rational agent structure is a tuple $< \mathcal{B}, \mathcal{I}, \mathcal{A}, \mathcal{T} >$ such that (i) $\mathcal{B}$ is a set of extended logic programming; (ii) $\mathcal{I}$ is a not deductively closed set of extended logic programming sentences, such that

1. its elements are not mutually contradictory, nor contradictory with the agent's beliefs, i.e., $\not\exists i \in \mathcal{I}\ .[(\mathcal{B} \cup \mathcal{I} \cup \mathcal{A} \cup \mathcal{T} \models_p i) \wedge (\mathcal{B} \cup \mathcal{I} \cup \mathcal{A} \cup \mathcal{T} \models_p \neg i)]$
2. $\forall (holds\_at(P, T)) \in \mathcal{I}.(now < T)$; and
3. $\forall (happens(E, P, T)) \in \mathcal{I}.(now < T)$;

$\mathcal{A}$ is a set of $initiates/2$ and $terminates/2$ clauses; $\mathcal{T}$ is the set of EC axioms.

Notice that, in constraint number 1, in the above definition, it is not enough to state just $\not\exists i \in \mathcal{I}.(\mathcal{B} \cup \mathcal{I} \cup \mathcal{A} \cup \mathcal{T} \models_p \bot)$, because we would be considering logical consequences of intentions as reasons to invalidate them. Also, constraint number 1 guarantees the satisfaction of the last basic requirement of intentions with respect to beliefs, namely that they are not believed to be impossible (if we include in $\mathcal{I}$ some sentence i which is believed to be impossible, i.e., $\neg i \in \mathcal{B}$, a contradiction would raise, offending restriction number 1).

The different perspective adopted — an agent-based one, instead of a designer-based one — may be noticed in many aspects, namely the definitions presented so far take into account not what is possible to derive with this framework - as it would be in a design tool - but what the agent in fact generates, while reasoning. Also, the definitions and restrictions are based on the notion of $\models_p$, which is clearly computable[12] and may be used by the agent while reasoning.

5 Further Work

As stated in section 4, this change of perspective also introduces some problems that are usually ignored in traditional theories. Mainly, these problems are connected with the dynamic aspects of intentions: how do they appear? how do they

[12] See [1] for more details.

vanish? Remember that intentions guide the agents behavior. Consequently, it is not enough to define intentions and relating mental states just as sets of logical sentences with some constraints, because we are only able to determine valid states of the mental attitudes. Those dynamic aspects are ignored[13].

We may classify such aspects in two types: *motive-to-intentions process* and *intention-to-action process*. Motive-to-intention process relates to the origin of intentions. During decision making, agents adopt intentions that will guide their behavior. Such intentions are restricted by its beliefs and are formed from other pro-attitudes, such as obligations, necessities, desires and so. Most of the existing theories do not focus this aspect[14]. Intention-to-action process relates to the production of actions. When the time comes, the agent has to transform his intentions into actions in order to fulfill those intentions. This has long been studied in the planning field, but no connection has been established with theories of agency[15]. These questions will influence two major aspects of intentions, namely intention revision (adopting and abandoning intentions) and commitment.

Suppose that an agent tries to adopt a new intention that is contradictory with existing ones. Must he ignore the newly arriving intention in favor of the older one? This would be quite reasonable, since intentions are aimed to avoid the continuous balancing of other mental states, and the agent had already committed to the older intentions. But, in some situations, this might not be the case. For example, suppose (also) that the agent in question is a robot performing some task and that he starts to run out of energy. Presumably, he should form a new intention, namely to stop and get recharged. However, according to the above description, such an intention would be ignored, since it would conflict with the existing one. In order to decide whether to continue the ongoing task or to recharge, it would be necessary to establish revision criteria that take into account the origin of intentions. We could define, for example, that intentions that come from obligations are stronger than the ones originating from desires; and that the ones coming from necessities are even stronger. In fact, it is necessary to define the other pro-attitudes that must be part of the rational agent structure, and how they must relate to intentions, introducing concepts like priorities among motives. Also changes in these pro-attitudes may cause the abandonment of intentions. If an intention i is adopted because of an obligation o (suppose, for instance, an order given by someone) and, afterwards, such obligation is retracted (for instance, the order is canceled), the agent should also abandon the respective intention. This notion of motives of intentions is also interesting to correctly define the notion of commitment. An agent must commit to an intention as long as (as in section 4) he does not believe it to be impossible, or to have not yet been achieved, and (differently from the existing theories) as long as the motives to that intention are still valid.

[13] And, in fact, the same happens with the existing theories that adopt a designer's perspective.

[14] An exception would be [**?**], but it is an architecture definition, indeed.

[15] Exceptions would be [**?**] and [13], where intentions are connected to preconceived strategies for achieving them.

Also, the tracking of the success of actions performed to achieve intentions may be used to decide whether to continue trying to achieve them or not.

6 Conclusion

In this paper, we sketched the basic definitions of a theory of intentions that adopt an agents' perspective, instead of a designers' one. We have presented the logical basis, namely ELP with a paraconsistent version of WFSX and a variation of the Event Calculus, on which we based the preliminary definitions of such a theory of intentions. Preliminary definitions, because we have only showed how to treat the static aspects of intentions, namely its semantics and its relation to beliefs. As discussed in section 5, a complete theory must encompass also those dynamic aspects of intentions. We will focus, in further work, on what we have called the *motive-to-intention process*, since it deeply affects the intention revision process. We are convinced that the change of perspective we have suggested here is more adequate for a theory of agency, since it provides tools for the agents, focus on aspects so far left apart and still provides tools with which we may reason about the agent. Also, we believe that this approach reduces the gap between agent specification and agent implementation.

References

1. J.J. Alferes. *Semantics of logic programs with explicit negation.* PhD thesis, Universidade Nova de Lisboa, Lisbon, Portugal, october 1993.
2. M.E. Bratman. What is intention? In P.R. Cohen, J.L. Morgan, and M. Pollack, editors, *Intentions in Communication*, chapter 1. The MIT Press, Cambridge, MA, 1990.
3. C. Castelfranchi. Social power. In Y. Demazeau and J.P. Muller, editors, *Descentralized AI – Proceedings of the First European Workshop on Modelling Autonomous Agents and Multi-Agents Worlds (MAAMAW'89)*, Amsterdam, The Netherlands, 1990. Elsevier Science Publishers.
4. P.R. Cohen and H.J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42:213–261, 1990.
5. N.R. Jennings. On being responsible. In E. Werner and Y. Demazeau, editors, *Descentralized AI 3 – Proceedings of the Third European Workshop on Modelling Autonomous Agents and Multi-Agents Worlds (MAAMAW'91)*, Amsterdam, The Netherlands, 1992. Elsevier Science Publishers.
6. K. Konolige and M. Pollack. A representationalist theory of intentions. In *Proceedings of the XII International Joint Conference on Artificial Intelligence (IJCAI'93)*, Chambéry, France, 1993. IJCAI inc.
7. R.A. Kowalski and M.J. Sergot. A logic-based calculus of events. *New Generation Computing*, 4:67–95, 1986.
8. L. Messiaen. *Localized abductive planning with the event calculus.* PhD thesis, Katholieke Universiteit Leuven, Leuven (Heverlee), 1992.
9. P. Quaresma and G.P. Lopes. Unified logical programming approach to the abduction of plans and intentions in information-seeking dialogues. *Journal of Logic Programming*, 24(1&2), 1995. Special Issue on Computational Linguistics and Logic Programming.

10. A.S. Rao and M.P. Georgeff. Modelling rational agents within a BDI-architecture. In R. Fikes and E. Sandewall, editors, *Proceedings of the Knowledge Representation and Reasoning'91 (KR&R'91)*, San Mateo, CA., 1991. Morgan Kauffman Publishers.
11. M. Shanaham. A circumscriptive calculus of events. *Artificial Intelligence*, 2, 1995.
12. M. Singh. A critical examination of the Cohen-Levesque theory of intentions. In *Proceedings of the Tenth European Conference of Artificial Intelligence (ECAI'92)*, Vienna, Austria, 1992. ECAI inc.
13. M. Singh. *Multiagent systems: a theoretical framework for intentions, know-how, and communications.* Springer-Verlag, Heidelberg, Germany, 1994. Lecture Notes in Artificial Intelligence (LNAI 799).
14. E. Werner. A unified view of information, intentions and ability. In Y. Demazeau and J.P. Muller, editors, *Descentralized AI 2 – Proceedings of the Second European Workshop on Modelling Autonomous Agents and Multi-Agents Worlds (MAAMAW'90)*, Amsterdam, The Netherlands, 1991. Elsevier Science Publishers.

4 Iberamia96(WDAI) –"Motive Processing and its Role in Rational Reasoning"

Este "position paper" submetido ao "workshop" de IA Distribuída no Iberamia de 1996[MÓR 96] defende a idéia que é necessário diminuir a distância existente entre teorias e arquiteturas de agentes, a fim de que seja possível modelar e construir agentes que garantidamente exibam as propriedades esperadas: autonomia, reatividade e pro-atividade. Nos modelos correntes, pode-se distinguir entre teorias de agentes, que apresentam formalmente as características teóricas destes agentes, e arquiteturas de agentes, que focam nos aspectos de implementação dos mesmos. Esta aproximação entre modelos e arquiteturas é o princípio básico que guia a construção do modelo desenvolvido.

POSITION PAPER: Motive Processing and its Role in Rational Reasoning

Michael da Costa Móra[*] Rosa Maria Viccari[†]

Abstract

In this position paper we argue for two ideas: that an adequate theory of agency must reduce the gap that exists between agent theories and architectures and that the notion of *motive* or *motivational attitude* must be part of such a theory of agency, if it is to account for properties like autonomy, reactivety and pro-activeness.

1 Introduction

In the recent years, the term *agent* has been incorporated to the vocabulary of mainstream Computer Science (CS), used to name from simple system process to highly skilled software/hardware ensembles, denoting an entity created to perform some task or set of tasks. Such a wide notion is straightened in Artificial Intelligence (AI), where researchers are interested in a notion that ascribes to the agent the property of being intelligent. Nevertheless, there is not, in the research community, a consensual definition of what an agent is. We can notice that, instead of defining what an agent is, it is more usual to state which properties a system should present in order to be considered an agent. Generally, an agent is a piece of hardware and/or software that enjoys properties like autonomy, social ability, reactivity, pro-activeness and adaptability. Except for adaptability, the notion of agency that is defined by these properties is widely accepted, even in mainstream CS[1] as, for instance, in object-based concurrent programming that adopt independent, concurrently active logical objects. Adaptability, however, and pro-activeness to its full extent, are not so easily found in simple systems, but are of special importance to intelligent agents.

[*]MsC in Computer Science, PhD student at the Curso de Pós-Graduação em Ciência da Computação, Universidade Federal do Rio Grande do Sul. E-mail: michael@inf.ufrgs.br.

[†]PhD in Electrical Engineering, Assistent Professor at the Curso de Pós-Graduação em Ciência da Computação, Universidade Federal do Rio Grande do Sul. E-mail: rosa@inf.ufrgs.br.

[1]Shoham calls these four properties a *weak notion of agency*, in contrast to a *strong notion of agency*, which assumes the use of mental states concepts to model agents.

For some researchers, though, it is not enough that a system present such properties in order to be considered an agent. For those that adopt a *mentalistic approach* the term agent means a computer system that can be viewed as consisting of mental states such as beliefs, intentions, motives, expectations, obligations and so on[8] [**?**]. When a mentalistic approach is adopted, the central problem becomes to choose the mental states that should be used to characterize an agent. Searle[20] divided the mental states or attitudes in two major categories: *information attitudes* and *pro-active attitudes*. Information attitudes relate to the information an agent has about the world where it leaves, precisely *knowledge* and *belief*. Pro-attitudes are those that, in some way, guide the agent's behavior. Exactly what combination of such mental states is adequate to describe an agent is no consensus. In general, it seems reasonable to require, at least, one attitude of each kind. Davidson[7] argues that *desires* and *beliefs* are the two basic mental states, and that all the other ones could be reduced to them. Bratman[2], in his turn, states that intentions, which seem not to be reducible to desires and beliefs , must be considered. In fact, there is a great deal of mental states configurations that can be found in the literature.

Our purpose in this position paper is to argue for two ideas: that an adequate theory of agency must reduce the gap that exists between agent theories and architectures and that the notion of motive or motivational attitude must be part of such a theory of agency, if it is to account for properties like autonomy, reactivety and pro-activeness.

2 Theories X Architectures: the Gap Problem

It is interesting to notice that the existing models of agents have one of two characteristics: or it is a formal theory of agency, defined in some sort of logical language, or it is an agent architecture oriented to implementation. In the case of theories [6][21][17][12][5], they adopt a *designer perspective*, i.e., they are models that provide logical tools with which designers may specify and reason about agents, and that may be used as guides for constructing agents, with no guarantee that the final result will respect the specification. As an immediate consequence, they do not provide a tool, a logical framework with which agents may reason. In fact, they are quite distant from computational system, as they use logics that cannot be easily treated, nor they provide algorithms that can be used for constructing such model. In the case of architectures [9][10][13][**?**][11][4][15], we identify the opposite situation. They are implementation-oriented, and do not provide adequate tools for designers to reason about agents. In general, it is not possible to prove properties of such architectures, since they are not formally defined. A notable exception is the SEM architecture[**?**], that uses a formal framework to define its components. Nevertheless, it does not supply a logic for agent to reason with, and also defines informally some of its aspects, namely the relation among the mental states.

This all mean that there is a great gap between agent specification and agent implementation. It would be convenient to have an agent model formally defined using some formalism computationally tractable. Such a model would allow us to formally define an agent and its properties, verify if the agent respects the properties defined, and then test the agent to see if the results it presents when executed are as expected, i.e., if the property as it is defined is correspondent to the behavior obtained. This is our first motivation. We argue that an adequate theory of agency must reduce the gap that exists between agent theories and architectures.This can be achieved by adopting an agent perspective, i.e., defining a theory that is to be used by the agent when reasoning about the world[2]. This changes of perspective simplify some traditional problems and induces us to deal with some questions that are in general ignored by traditional theories[14]. In [14], a preliminary definition of this sort is presented. In that paper, a theory of intentions is sketched using as underlying formalism *Logic Programming Extended with Explicit Negation* (ELP) with its semantics being given by a paraconsistent version of the *Well-Founded Semantics* augmented with *Explicit Negation* (WFSX)[1]. On top of this logical basis, a variation of the Event Calculus (EC) is built and used as the logical formalism for action and time.

3 The Quest for Motivational Attitudes

There is a consensual idea on how we should define beliefs and intentions. Beliefs have long been studied and constitute consistent body of knowledge in AI. Intentions are not so developed, but the pending questions are concerned only with implementation and representation issues. Its role, its relation to other mental states, namely beliefs, are very well defined and accepted[3]. However, these attitudes are not enough if we are to define a theory of agency. Another pro-active attitude, at least, is needed. This is so because intentions by itself, along with beliefs, do not totally explain rational behavior. By definition, the agent must act on behalf of its intentions, trying to achieve them. In case of failure, it must persist until he believes it is no longer possible to achieve it, or it does not know how to do it; for this reason, the agent cannot adopt contradictory intentions (neither with relation to other intentions, nor to beliefs). These conditions on intentions are part of the definition of rational behavior. But there some other important questions: where do intentions come from? How do they vanish, when an agent repeatedly fails to achieve it? What to do if an agent is acting on behalf of an intention, and an unexpected event causes an urgent situation that requires the agent's attention?

The need to answer these questions is our second motivation. One way to try to answer them is to define the agent's fundamental purposes. Not its goals, which are transient, but its reasons for having goals, its "role in life ", the

[2]The agent builds the world model while reasoning.

rules that govern its existence. The most usual solution to this problem is to adopt desires as another pro-active mental state. This is, although, a partial solution, because desires may be used to justify the origins of intentions (as in [**?**]) but do not provide an answer to the other questions. Also, it may be a wrong philosophical choice. In [19], the author states that maybe actions do not arise from desires (through adopted intentions) as it is usually thought. He argues for the need to make a distinction between two senses of desire: roughly, genuine desires and pro-attitudes. Moreover, he argues that the apparently plausible explanations of actions in terms of the agent's desires can be seen to be mistaken. So, we need a broader notion, the notion of *motives* or *motivational attitudes.*

The motives define the general profile of the agent's behavior. Consequently, they influence directly on the agent's decision process. Roughly speaking, motives are pro-attitudes like needs, interests, preferences, obligations and so[18]. Their role is to originate behaviors through the creation of intentions and to govern the agent, when choosing among contradictory motives or intentions, during planning, when it is necessary to decide how to satisfy intentions.

This idea is quite recent in AI literature. The concepts are still very intuitive and need to be supported by a conceptual framework (like the one from [19], for instance). Also, there is not a formalization of such ideas. Some preliminary work, like [13] and [16], has already been done. Nevertheless, they do not present any formal definition of such concepts.

References

[1] J. Alferes. *Semantics of logic programs with explicit negation.* PhD thesis, Universidade Nova de Lisboa, Lisbon, Portugal, october 1993.

[2] M. Bratman. *Intentions, plans and practical reasoning.* Harvard University Press, Cambridge, MA, 1987.

[3] M. Bratman. What is intention? In P. Cohen, J. Morgan, and M. Pollack, editors, *Intentions in Communication*, chapter 1. The MIT Press, Cambridge, MA, 1990.

[4] M. Bratman, D. Israel, and M. Pollack. Plans and resource bounded practical reasoning. *Computational Intelligence*, 4:349–355, 1988.

[5] P. Cohen and H. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42:213–261, 1990.

[6] P. Cohen and H. Levesque. Persistence, intention and commitment. In P. Cohen, J. Morgan, and M. Pollack, editors, *Intentions in Communication*, chapter 2. The MIT Press, Cambridge, MA, 1990.

[7] D. Davidson. *Essays on actions and events.* Oxford University Press, New York, NY, 1980.

[8] D. Dennet. *The intentional stance.* The MIT Press, Cambridge, MA, 1987.

[9] I. Ferguson. *Touringmachines: an Architecture for Dynamic Rational Mobile Agents.* PhD thesis, Cambridge University, Cambridge, UK, 1992.

[10] M. Georgeff and A. Lansky. Reactive reasoning and planning. In *Proceedings of the National Conference of Artificial Intelligence (AAAI'91).* AAAI inc., 1991.

[11] N. Jennings. Controlling cooperative problem solving in industrial multi-agent systems using joint intentions. *Artificial Intelligence Journal*, 47, 1994.

[12] K. Konolige and M. Pollack. A representationalist theory of intentions. In *Proceedings of the XII International Joint Conference on Artificial Intelligence (IJCAI'93)*, Chambéry, France, 1993. IJCAI inc.

[13] D. Moffat and N. Frijda. Where there's a *will* there's an agent. In *Proceedings of ECAI'94 Workshop on Agent Theories, Architecture and Languages (ATAL'94)*, Amsterdam, NE, 1995. ECAI inc. Lecture Notes in Artificial Intelligence (LNAI) 890.

[14] M. Móra, J. Lopes, and H. Coelho. Modelling intentions with extended logic programming. In J. Wainer and A. Carvalho, editors, *Advances in artificial intelligence: proceedings of the 12th Brasilian Symposium on Artificial Intelligence*, Berlin, Germany, 1995. SBC, Springer-Verlag. Lecture Notes in Artificial Intelligence (LNAI 991).

[15] N. Nilsson. Shakey, the robot. Technical report, Menlo Park, CA, 1984.

[16] T. Normam and D. Long. Goal creation in motivated agents. In *Proceedings of ECAI Workshop on Agent Theories, Architectures and Languages (ATAL'94)*, Amsterdam, NE., 1995. ECAI org.

[17] A. Rao and M. Georgeff. Modelling rational agents within a BDI-architecture. In R. Fikes and E. Sandewall, editors, *Proceedings of the Knowledge Representation and Reasoning'91 (KR&R'91)*, San Mateo, CA., 1991. Morgan Kauffman Publishers.

[18] G. Schueler. Pro-attitudes and direction of fit. *Mind*, 100:277–281, 1991.

[19] G. Schueler. *Desire: Its role in Practical Reason and the Explanation of Action.* The MIT Press, Cambridge, Ma., 1995.

[20] J. Searle. What is an intentional state? In H. Dreyfuss and H. Hall, editors, *Husserl, Intentionality and Cognitive Science*, volume 42, pages 213–261. 1984.

[21] M. Singh. *Multiagent systems: a theoretical framework for intentions, know-how, and communications.* Springer-Verlag, Heidelberg, Germany, 1994. Lecture Notes in Artificial Intelligence (LNAI 799).

5 EPIA97 – “Modelling Dynamic Aspects of Intentions”

Este artigo, apresentado no EPIA97[MÓR 97], mostra como a teoria de intenções desenvolvida anteriormente pode ser aumentada de modo a englobar também os aspectos dinâmicos deste estado mental. Notamente, os precessos de adoção e abandono de intenções, tendo em consideração a relação de causa e efeito entre desejos e intenções. Desta forma, complementa-se o modelo formal proposto para o agente, que agora inclui aspectos estáticos e dinâmicos das intenções.

Affecting the Stability of Intentions

Michael da Costa Móra[2,1] [*], Gabriel Pereira Lopes[1] [**], Helder Coelho[3] [***], and Rosa M. Viccari[2] [†]

[1] CENTRIA/DI – Universidade Nova de Lisboa
Quinta da Torre – 2825 Monte da Caparica – Portugal
[2] CPGCC – II – Universidade Federal do Rio Grande do Sul
Bloco IV – Campus do Vale – Av. Bento Gonçalves 9500 – Porto Alegre – RS – Brasil
[3] DI - FCUL – Universidade de Lisboa
Bloco C5, Piso 1 – Campo Grande – 1700 Lisboa – Portugal

Abstract.

Keywords: intentions; mental states modeling; agent architectures; distributed artificial intelligence; logic programming.

1 Introduction

When we model agents using mental states as beliefs, desires and intentions, one of the hardest tasks is to represent dynamic aspects of the agent's behavior. We illustrate these aspects in the following example.

Example 1. Mike has two distinct programmes for his evening. He may go to the local opera house or to a soccer game, at the local stadium. Both events start at the same time.

Mike cannot go to both the opera and the soccer game. Therefore, he has to *choose* one of them. Mike may make this choice considering many factors, like his preferences (he may be more fond of opera than soccer, for instance), costs (tickets for the opera cost three times as much as those for the soccer game; the stadium may be several kilometers far from his home while the opera house is 15 minutes walk distance) or others. Suppose that, after all these considerations, Mike chooses to go to the opera. As he is a rational person, we expect him to plan actions and to act in order to accomplish his intents to go to the opera. We also expect that, since he has already considered the pros and cons of his

[*] PhD student at the CPGCC/UFRGS, in Brazil. Currently at the CENTRIA/DI/UNL, in Portugal. Supported by project DIXIT/JNICT. Email: mdm@di.fct.unl.pt .

[**] Researcher at the CENTRIA/DI/UNL, in Portugal. Supported by project DIXIT/JNICT. Email: gpl@di.fct.unl.pt .

[***] Professor at the DI/FCUL/UL. Email: hcoelho@di.fc.ul.pt .

[†] Professor at the CPGCC/UFRGS, in Brazil. Email: rosa@inf.ufrgs.br .

options, he does not reconsiders again and again his previous choice. That is, we expect that, after he decides what to do, he commits to that decision.

Meanwhile, he receives a call from a colleague telling him the term paper they believed was due to the end of the month is, in fact, due to the next morning. Although Mike had already decided to go to the opera, we expect him to reconsider his choices, as he is now in presence of new facts that pose him a new important situation that was not previously present.

Although this behavior may be characterized as rational and it is what one would expect of a rational agent, most of the existing formal models of agents are not capable of capturing it. These models follow Bratman's analysis[Bra90], where he describes the relation among the three basic mental states that compose an agent — beliefs-desires-intentions (BDI)[1]. Bratman argues that, since agents are assumed to be resource-bounded, they cannot continuously evaluate their competing beliefs and desires in order to act rationally. After some reasoning, agents have to commit to some set of choices. It is this choice followed by a commitment that characterizes intentions. Thus, intentions are viewed as a compromise the agent assumes with a specific possible future that is abandoned only in certain circumstances, namely when the agent believes it is satisfied or when it believes it is impossible to achieve it. This means that, once an intention is adopted, the agent will pursue that intention, planning actions to accomplish it, re-planning when a failure occurs, and so. Also, it means that intentions will constrain the adoption of future intentions, i.e., they will form a *screen of admissibility*[Bra90] for adopting new intentions, preventing the agent to make new choices that are not consistent with those previously made. Therefore, with intentions as a kernel mental attitude, agents have a general strategy that prevent them from having to reconsider all of their mental states every time, in order to know what to do.

But this characterization of intentions lead to some strange behavior. For instance, in our example, as Mike had already adopted as intention to go to the Opera, this intention would constrain the adoption of new ones. Therefore, when he is notified about the real due date of his term paper, he would not adopt *to finish the paper tonight* as an intention, since it would be contradictory with a previous one, namely *to go to the Opera*.

In this paper, we show how to model these dynamics aspects of the agent's behavior, namely how an agent decides to break commitment to satisfy previous intentions and re-evaluate its competing mental states in order to decide what to do, without having to constantly engage in this expensive reasoning process.

Recall that, in existing models (like [CL90,Sin94]), after committing to some intention, agents will only re-evaluate their choices when they believe that an intention has been satisfied or that an intention is impossible to satisfy. Those models specify these conditions, but they do not model how an agent may detect

[1] In fact, Bratman stresses on the relation between intentions and beliefs. He mentions that desires would constitute a set of options where the agent would pick up his intentions, but he does not focus on this mental state and its relations with the others.

that such conditions hold and that it is relevant, at a certain time, to reconsider its decisions. We do this by modelling them as constraints that are part of the agent's beliefs set and that, when violated, trigger the intention revision process. Having defined how an agent detects relevant situations and how it triggers its reasoning process, we define additional triggering conditions, based on the causal relation between desires and intentions, that relax the notion of commitment and avoids the type of behaviors described above.

Our model focus on the formal definition of mental states and on how the agent behaves, given such mental states. Initially we define the three basic mental states and the static relations between them, namely constraints on consistency among those mental states. Afterwards, we advance with the definition of how agents select their intentions among its competing desires. Desires are schemas of properties agents will eventually want to bring about. Differently from intentions, agents are not committed to their desires, i.e., agents do not necessarily act in order to satisfy all of its desires. Instead, they select among their (possibly contradictory) desires those that are relevant and that they believe may be achieved. We rely on the logical formalism, namely Extended Logic Programming with the Well-Founded Extended Semantics to detect and remove contradiction, as well as to perform several types of non-monotonic reasoning, namely defeasible and abductive reasoning.

As the reasoning mechanisms provided by ELP are at the very core of our model, in the next sections we start by briefly describing the syntax and semantics of extended logic programming, as well as how to use it to perform the necessary non-monotonic reasoning (section 2) and to deal with action and time using an extension of the Event Calculus[Mes92,QL97](section 3). Afterwards, we present the formal definition of our model of intentions (section 4). Finally, we draw some conclusions and point to some future work.

2 Extended Logic Programming

Differently of normal logic programs, where negative information is only stated implicitly (i.e., a proposition is false only if it cannot be proved to be true), extended logic programs have a second kind of negation that allows us to explicitly represent negative information. If, on one hand, this second negation increases the representational power of the language, it may also introduce contradictory information in programs. Therefore, it is necessary to attribute some meaning to contradictory programs and to be able to deal with contradiction. The paraconsistent version of the $WFSX$ semantics, $WFSX_P$, attributes meaning to contradictory programs and is used as a tool by the revision mechanism that restores consistency to contradictory programs. This revision mechanism may be used to perform several kinds of reasoning, as we will see in the following sections.

2.1 Language and Semantics

An extended logic program (ELP) is a set of rules

$$H \leftarrow B_1, \ldots, B_n, not\ C_1, \ldots, not\ C_m \qquad (\text{m,n} \geq 0)$$

where $H, B_1, \ldots, B_n, C_1, \ldots, C_m$ are objective literals. An objective literal is either an atom A or its explicit negation $\neg A$. The symbol *not* stands for negation by default and *not* L is a default literal. Literals are either objective or default literals and $\neg\neg L \equiv L$. The language also allows for integrity constraints of the form

$$A_1 \vee \ldots \vee A_l \Leftarrow B_1, \ldots, B_n, not\ C_1, \ldots, not\ C_m \qquad (\text{m,n} \geq 0;\ \text{l} \geq 1)$$

where $A_1, \ldots, A_l, B_1, \ldots, B_n, C_1, \ldots, C_m$ are objective literals, stating that at least one of the A_i (i ≥ 1), should hold if its body $B_1, \ldots, B_n, not\ C_1, \ldots,$ *not* C_m (m,n ≥ 0) holds. Particularly, when $A =\perp$, where $\perp$ stands for *contradiction*, it means that a contradiction is raised when the constraint body holds. The set of all objective literals of a program P is the extended Herbrand base of P denoted by $\mathcal{H}(P)$.

In order to attribute meaning to ELP programs, we need to define the notions of interpretation and satisfaction of ELP clauses. Roughly, an interpretation of an ELP program is a set of literals. A literal L is true in an interpretation I if $L \in I$, L is false in I if *not* $L \in I$, and L is undefined, otherwise.

Definition 2 (Interpretation). An interpretation I of an ELP program P is denoted by $T \cup not\ F$, where T and F are subsets of $\mathcal{H}(P)$ and that verify the coherence principle: if $\neg L \in T$ then $L \in F$. When $F = \{a_1, \ldots, a_n, \ldots\}$, *not* F means $\{not\ a_1, \ldots, not\ a_2, \ldots\}$. The set T contains all ground objective literals *true* in I, the set F contains all ground objective literals *false* in I. The truth value of the remaining objective literals is *undefined* (the truth value of a default literal *not* L is the 3-valued complement of L).

Differently from the definition of interpretation for the $WFSX$, that enforces that T and F be disjoint subsets of the Herbrand base, $WFSX_P$ allows contradictions by allowing T and F to have elements in common.

Definition 3 (Satisfaction of an Extended Logic Program). Let I be an interpretation with respect to a given logic program. We say that:

– I satisfies a default or objective literal, denoted by $I \models L$, iff $L \in I$;
– I satisfies the conjunction $L_1, \ldots, L_n, not\ G_1, \ldots, not\ G_m$,
denoted by $I \models L_1, \ldots, L_n, not\ G_1, \ldots, not\ G_m$ iff
$I \models L_1$,..., and $I \models L_n$, and $I \models not\ G_1$,, ..., and $I \models not\ G_m$;
– I satisfies the rule $H \leftarrow L_1, \ldots, L_n, not\ G_1, \ldots, not\ G_m$,
denoted by $I \models (H \leftarrow L_1, \ldots, L_n, not\ G_1, \ldots, not\ G_m)$,
iff whenever the body of the rule is satisfied by I then the head is also satisfied by I.

Definition 4 (Model of an Extended Logic Program). An interpretation I is a model of an extended logic program P iff every rule in P is satisfied by I.

The definition of model is not enough, as an ELP program may have several models.

Example 5. Consider the program P_5 bellow:

$$\begin{aligned} &\neg b \\ &b \leftarrow a \\ &a \leftarrow not\ a, not\ c \end{aligned}$$

Both $I_1 = \{\neg b, not\ \neg b\}$ and $I_2 = \{\neg b, not\ \neg b, not\ a\}$ are models for P_5.

The $WFSX_P$ definition enforces some other conditions that attribute a unique model to each program. Through some program transformations, which produce (possibly more than one) non-negative programs, it is possible to get all the consequences of the original program, even those leading to contradictions, as well as those arising from contradictions. It can be shown that the operator that is used to obtain such a set of consequences (named Φ_P) is monotonic under set inclusion of interpretations, for any program P[2]. Hence, it has a least fixpoint, which can be obtained by iterating Φ_P starting from the empty set. This least fixed point is the desired well-founded model.

Definition 6 (Paraconsistent WFSX). The paraconsistent WFSX of an extended logic program P, denoted by $WFSX_P(P)$, is the least fixpoint of Φ applied to P. If some literal L belongs to the paraconsistent WFSX of P we write $P \models_p L$. A program P is said to be *contradictory* iff, for some positive literal L, both $P \models_p L$ and $P \models_p \neg L$ hold, or iff a constraint is not satisfied. P is said to be consistent iff it is not contradictory.

2.2 Program Revision and Non-Monotonic Reasoning

As we stated before, due to the use of explicit negation, programs may be contradictory. In order to restore consistency in programs that are contradictory with respect to $WFSX$, the program is submitted to a revision process that relies on the allowance to change the truth value of some set of literals. This set of literals is the set of *revisable literals*, and can be any subset of $\mathcal{H}$ for which there are no rules or there are only facts in the program. No other restriction is made on which literals should be considered revisable. They are supposed to be provided by the user, along with the program.

The revision process changes the truth value of revisable literals in a minimal way and in all alternative ways of removing contradiction. *Minimally revised programs* can be defined as those programs obtained from the original one after modifying the subsets of revisable literals.

[2] Due to lack of space, the formal definitions of such operators are not presented here. For details, refer to [AP96].

Example 7. Consider program P_7:

$$a \leftarrow \neg b \tag{1}$$
$$\neg a \leftarrow \textit{not } c \tag{2}$$
$$\neg b \tag{3}$$

P_7 is a contradictory program, as we have $P_7 \models_P a$ ($\neg b$ is true, and entails a) and $P_7 \models_P \neg a$ (c is false by default, i.e., *not* c is true, and entails $\neg a$). If we have the set $R = \{c, \neg b\}$ of revisable literals, we have two possible minimal revisions, namely $Rev_1 = \{c \leftarrow \mathbf{t}\}$ and $Rev_2 = \{\neg b \leftarrow \mathbf{f}\}$. That is, we can restore consistency in program P_7 changing the truth value of c to true or changing the truth value of $\neg b$ to false[3].

The ability to represent negative information, along with a well-defined procedure that restores consistency in logic program, makes ELP suitable to be used to perform different forms of non-monotonic reasoning [AP96]. In particular, we are interested in two of these forms of non-monotonic reasoning, namely *defeasible reasoning* and *abductive reasoning.*

Defeasible Reasoning A defeasible rule is a rule of the form *Normally if A then B.* ELP allows us to express defeasible reasoning and to give meaning to a set of rules (defeasible or not) when contradiction arises from the application of the defeasible rules.

Example 8. Consider the following statements (from [AP96]):

1. Normally birds fly.
2. Penguins don't fly.
3. Penguins are birds.
4. a is a penguin.

represented by the program P_8 bellow:

$$f(X) \leftarrow b(X), \textit{not } ab(X) \tag{4}$$
$$\neg f(X) \leftarrow p(X) \tag{5}$$
$$b(X) \leftarrow p(X) \tag{6}$$
$$p(a) \tag{7}$$

Since there are no rules for $ab(a)$, *not* $ab(a)$ holds and entails $f(a)$. Also, $f(a)$ follows from $p(a)$, in the second rule. Therefore, the program is contradictory. In this case, rule 4 is defeasible and gives raise to contradiction due to the assumption that $ab(a)$ is false. Thus, $f(a)$ should not have been concluded and the intended meaning of P requires $\neg f(a)$ and *not* $f(a)$ (by coherence).

[3] In fact, there are two types of program revision, three-valued and two-values. In three-valued program revision, truth values may change from **t** and **f** to **u**, and from **u** to **t** or **f**. In this work, we adopt two-valued program revisions.

If we revise P_8 with revisable set $R = \{ab(X)\}$, we obtain one revision $Rev = \{ab(a) \leftarrow \mathbf{t}\}$. The semantics of the revised program obtained with these revision is $\{p(a), not\ \neg p(a), b(a), not\ \neg b(a), \neg f(a), not\ f(a), not\ \neg ab(a)\}$, as it was intended.

We use defeasible reasoning to select a consistent subset of the agent's desires that will constitute its intentions (in section 4).

Abductive Reasoning Abductive reasoning consists of, given a theory T and a set of observations O, to find a theory Δ such that $T \cup \Delta \models O$ and $T \cup \Delta$ is consistent.

In the ELP context, an abductive framework P' is a tuple $\langle P, Abd, IC\rangle$, where P is an extended logic program, Abd is the set of abducible literals and IC is the set of integrity constraints. An observation O has an abductive explanation Δ iff $P' \cup \Delta \models_P O$ and $P \not\models_P \Delta$. We may abduce a theory Δ that explains such observations making each of these observations O a new integrity constraint $O \Leftarrow$ and revising the program with revisable set Abd (recall that a program is contradictory iff for some literal L in program P, $P \models L$ and $P \models \neg L$, or a constraint is not satisfied).

Example 9. Consider the following statements:

1. Normally birds fly.
2. Penguins don't fly.
3. Penguins are birds.
4. Normally Tigers don't fly.
5. Normally Lions are mammals.

represented by the program P_9 bellow:

$$\begin{aligned} f(X) &\leftarrow b(X), not\ ab1(X) \\ \neg f(X) &\leftarrow p(X) \\ b(X) &\leftarrow p(X) \\ \neg f(X) &\leftarrow t(X), not\ ab2(X) \\ m(X) &\leftarrow l(X), not\ ab3(X) \end{aligned}$$

Is there a possible explanation for the fact $\neg f(a)$?

The abductive framework derived from program P_9 and observation $\neg f(a)$ is $P' = \langle P_9, \{t(X), p(X), l(X)\}, \{\neg f(a) \Leftarrow\}\rangle$. Since there are no rules for $p(a)$ and $t(a)$, P does not entails $\neg f(a)$. Therefore, the constraint is not satisfied and the program is contradictory. For revisable set Abd, there are two revisions that restore consistency in P', namely $Rev_1 = \{t(a)\}$ and $Rev_2 = \{p(a)\}$. These are the two possible explanations for $\neg f(a)$.

We use abductive reasoning to determine if an agent believes there is a possible course of actions that satisfies a desire (in section 4).

Preferred Revisions Sometimes, it may be necessary to state that we do not only want the minimal revisions provided by the formalism, but also that we prefer revisions that have a certain fact only after finding that no other revisions including other facts exist. The ELP formalism allows us to express preferences over the order of revisions using a labeled directed acyclic and/or graph defined by rules of the form[DNP94]:

$$Level_0 \ll Level_1 \wedge Level_2 \wedge \ldots Level_n (\mathrm{n} \geq 1)$$

$Level_i$ nodes in the graph are preference level identifiers. To each of these levels is associated a set of revisables denoted by $\mathcal{R}(Level_i)$. Rules like the one above for $Level_0$ state that we want to consider revisions for $Level_0$ only if, for some rule body, its levels have been considered and there are no revisions at any of those levels. The root of the preference graph is the node denoted by **bottom**. If we associate to example 9 the following preference graph:

$$\begin{array}{ll} 1 \ll bottom & 2 \ll 1 \\ \mathcal{R}(\mathbf{bottom}) = \{t(X)\} & \\ \mathcal{R}(1) = \{p(X)\} & \mathcal{R}(2) = \{l(X)\} \end{array}$$

then, instead of two revisions for that abductive framework, we would have only $Rev_1 = \{t(a)\}$, as revisions for $p(a)$ (at level 1) would only be considered if there were no revisions for lower levels ($t(a)$ at **bottom**, in this case).

We use preference specification when selecting the most appropriate subset of desires that will be adopted as intentions (in section 4).

3 The Event Calculus

When we reason about pro-attitudes like desires and intentions, we need to deal with properties that should hold at an instant of time and with actions that should be executed at a certain time. Therefore, in order to represent them and to reason about them, we need to have a logical formalism that deals with actions and time. In this work, we use a modified version of the Event Calculus (EC) proposed in [Qua97,QL97]. This version of the EC allows events to have a duration and an identification, instead of being instantaneous and identified to the instant of time the event occurs, as in [Mes92]. As a consequence, events may occur simultaneously.

The predicate *holds_at* defining the properties that are true at a specific time is:

$$\begin{aligned} holds\_at(P,T) \leftarrow\ & happens(E,T_i,T_f), \\ & initiates(E,T_P,P), \\ & T_P < T, \\ & T_P >= T_i, \end{aligned} \tag{8}$$

$$\begin{aligned} & \mathit{persists}(T_P, P, T). \\ \mathit{persists}(T_P, P, T) &\leftarrow \mathit{not\ clipped}(T_P, P, T). && (9) \\ \mathit{clipped}(T_P, P, T) &\leftarrow \mathit{happens}(C, T_{ci}, T_{cf}), && (10) \\ & \mathit{terminates}(C, T_C, P), \\ & T_C >= T_{ci}, \\ & \mathit{not\ out}(T_C, T_P, T). \\ \mathit{out}(T_C, T_P, T) &\leftarrow T \leq T_C. && (11) \\ \mathit{out}(T_C, T_P, T) &\leftarrow T_C < T_P. && (12) \end{aligned}$$

The predicate $\mathit{happens}(E, T_i, T_f)$ means that event E occurred between T_i and T_f; $\mathit{initiates}(E, T, P)$ means that event E initiates P at time T; $\mathit{terminates}$ (E, T, P) means that event E terminates P at time T; $\mathit{persists}(T_P,P,T)$ means that P persists since T_P until T (at least). We assume there is a special time variable Now that represents the present time.

Note that a property P is true at a time T ($\mathit{holds\_at}(P, T)$), if there is a previous event that initiates P and if P persists until T. P persists until T if it can not be proved by default the existence of another event that terminates P before the time T.

We need additional rules for the relation between not holding a property and holding its negation and we also need to define the relation between the two kinds of negation:

$$\begin{aligned} \mathit{holds\_at}(\neg P, T) &\leftarrow \neg \mathit{holds\_at}(P, T). && (13) \\ \neg \mathit{holds\_at}(P, T) &\leftarrow \mathit{not\ holds\_at}(P, T). && (14) \end{aligned}$$

The predicates that will be abduced need to be related by some integrity rules:

1. Events cannot be associated to different time intervals:

$$\begin{aligned} \bot \Leftarrow\ & \mathit{happens}(E, T_{1i}, T_{1f}), && (15) \\ & \mathit{happens}(E, T_{2i}, T_{2f}), \\ & \mathit{not}(T_{1i} = T_{2i}, T_{1f} = T_{2f}). \end{aligned}$$

2. Events cannot have a negative duration:

$$\bot \Leftarrow \mathit{happens}(E, T_i, T_f),\ \mathit{not}(T_f < T_i). \qquad (16)$$

3. Events must have an associated action:

$$\begin{aligned} \bot \Leftarrow\ & \mathit{happens}(E, T_i, T_f), && (17) \\ & \mathit{not}(\mathit{act}(E, A)). \end{aligned}$$

The EC allows us to reason about the future, by hypothetically assuming a sequence of actions represented by *happens*/3 and *act*/2 predicates and verifying which properties would hold. It also allows us to reason about the past. In order to know if a given property P holds at time T, the EC checks what properties remain valid after the execution of the actions that happened before T. We are now ready to define the agent's model.

4 The Model of Intentions

In this section, we present our model of agents. We take a mentalistic approach, i.e., we define agents in terms of their mental states and we characterize the agent's behavior in terms of these mental states. Initially, we define what mental states agents posses and how these mental states relate to each other. Afterwards, we characterize the dynamic aspects of agents' behavior, namely how agents decide what to do (how they choose their intentions) and how and when agents reconsider their decisions.

4.1 The Mental States

Mental states are divided in two major categories: *information attitudes* and *pro-attitudes*[Sea84]. Information attitudes relate to the information an agent has about the environment where it lives, those that tend to reflect the state of the world, precisely *knowledge* and *beliefs*. Pro-attitudes are those that are related to actions and somehow lead the agent to act, that tend to make the agent modify the world in order to fit its mental states, i.e., attitudes like desires, intentions, obligations and so on. What exact combination of such mental states is adequate to describe an agent in a certain environment is no consensus. Very often, in the philosophical literature (see [Sea84], for instance), desires and beliefs have been used as the two basic mental states to define an intelligent agent. All the explanations of rational actions, and consequently all the other mental states, could be reduced to them. Nevertheless, according to Bratman[Bra87,Bra90], a third mental state, *intentions*, should be considered. In his detailed analysis, where he describes the relation among those three mental states — beliefs-desires-intentions (BDI), Bratman argues that, since agents are assumed to be resource-bounded, they cannot continuously evaluate their competing beliefs and desires in order to act rationally. After some reasoning, agents have to commit to some set of choices. It is this choice followed by a commitment that characterizes the intentions.

We start by defining desires. Desires are related to the state of affairs the agent eventually wants to bring about. But desires, in the sense usually presented, does not necessarily drive the agent to act. That is, the fact of an agent having a desire does not mean it will act to satisfy it. It means, instead, that before such an agent decides what to do, it will be engaged in a reasoning process, confronting its desires (the state of affairs it wants to bring about) with its beliefs (the current circumstances and constraints the world imposes). It will choose those desires that are possible according to some criteria it will act upon them. In other words, desires constitute the set of states among which the agent chooses what to do.

Notice that, since agents are not committed to their desires, they need not to be consistent, neither with other desires nor with other mental states.

Definition 10 (Desires Set). The *desires* of an agent is a set $\mathcal{D}$ of ELP sentences of the form $desires(D, P, T, A) \leftarrow Body$, where D is the desire identification, P is a property, T is a time point and A is list of attributes. $Body$ is any conjunction of literals. An agent desires that a property P holds at time T iff $desires(D, P, T, A) \in \mathcal{D}$, for some D and some A.

Our definition of desires allows the agent to have a desire that a certain property holds (or does not hold) in a specific instant of time (when T is instantiated). $Desires/4$ clauses may be facts, representing states the agent may want to achieve whenever possible, or rules, representing states to be achieved when a certain condition holds. The attributes associated to each desire define properties, like urgency, importance or priority[Bea94] that are used by to agent to choose the most appropriate desire (see bellow).

Beliefs constitute the agent's information attitude. They represent the information agents have about the environment and about themselves. Such information includes time axioms and action descriptions (see 3, along with agent capabilities.

Definition 11 (Beliefs Set). The *beliefs* of an agent is a consistent extended logic program $\mathcal{B}$, i.e. $\forall b \in \mathcal{B}.\mathcal{B} \not\models_P b$. An agent believes L iff $\mathcal{B} \models_P L$.

We assume that the agent continuously updates its beliefs to reflect changes it detects in the environment. Describing the belief update process is beyond the scope of this paper. We just assume that, whenever a new belief is added to the beliefs set, consistency is maintained.

As we stated before, intentions are characterized by a *choice* of a state of affairs to achieve, and a *commitment* to this choice. Thus, intentions are viewed as a compromise the agent assumes with a specific possible future. This means that, differently from desires, an intention may not be contradictory with other intentions, as it would not be rational for an agent to act in order to achieve incompatible states. Also, intentions should be supported by the agent's beliefs. That is, it would not be rational for an agent to intend something it does not believe is possible.

Once an intention is adopted, the agent will pursue that intention, planning actions to accomplish it, re-planning when a failure occurs, and so. These actions, as means that are used to achieve intentions, must also be adopted as intentions by agents.

Definition 12 (Intentions Set). The *intentions* of an agent is a set $\mathcal{I}$ of ELP sentences of the form $intends\_that(I, P, T, A)$ or of the form $intends\_to(I, Act, T, A)$, where I is the intention identification, P is a property, T is a time points, A is a list of attributes and Act is an action, and such that

1. $\forall intends\_that(I, P, T, A) \in \mathcal{I}.(Now \leq T)$
2. $\forall intends\_to(I, Act, T, A) \in \mathcal{I}.(Now \leq T)$
3. $\forall intends\_to(I, Act, T, A) \in \mathcal{I}.(\mathcal{B} \not\models_P (happens(E, T, T_F), act(E, Act)))$;
4. $\exists \Delta.(P' \cup \Delta \not\models_P \bot)$, where
   - P' is the abductive framework $\langle \mathcal{B}, \{happens/3, act/2\}, IC(\mathcal{I})\rangle$
   - $IC(\mathcal{I})$ is set of constraints generated by intentions, defined as
     - "$holds\_at(P, T) \Leftarrow$", for every $intends\_that(I, P, T, A)$ in $\mathcal{I}$;
     - "$happens(E, T, T_f) \Leftarrow$" and "$act(E, Act) \Leftarrow$" for every $intends\_to(I, Act, T, A)$ in $\mathcal{I}$;

An agent intends that a property P holds at time T iff $intends\_that(I, P, T, A) \in \mathcal{I}$. A agent intends to do an action Act at time T iff $intends\_to(I, Act, T, A) \in \mathcal{I}$.

The definition of intentions enforces its rationality constraints. Conditions 1 and 2 state that an agent should not intend something at a time that has already past. Condition 3,the non-triviality condition[CL90] states that an agent should not intend something it believes is already satisfied or that will be satisfied with no efforts by the agent. Condition 4 states that an agent only intends something it believes is possible to be achieved, i.e., if it believes there is a course of actions that leads to the intended state of affairs.

Notice that we take some measures to avoid the *side-effect problem*[Bra90]. Bratman states that *an agent who intends to do α an believes that doing α would require it to do β does not have to also intend β.* In our definition, an agent does not intend the *side-effects* of its intentions, as the side-effects are not in the intentions set[4]. To fully avoid this problem, it is also important, when we define dynamic aspects of intentions, namely how to originate intentions and how and when to revise them, that we characterize commitment in a way that side-effects of adopted intentions do not prevent agents from adopting intentions, neither that side-effects make agents revise their intentions.

4.2 Originating Intentions

Once we have characterized intentions and related mental states, it is necessary to define its dynamic aspects, namely how agents select intentions and when and how agents revise selected intentions. In order to illustrate the forthcoming definitions, we introduce the following example.

Example 13 (The Office Robot). In a building with several offices, there is a robot that transport mail from one office to another, when it is ordered to. Also, the robot has to be careful not to let his battery go out of charge, what would

[4] This is similar to the belief base approach, but it is simpler because we do not have to include the derivations in the base, as with belief bases.

prevent it from performing its tasks. The robot's desires and beliefs are:

Desires

$$\begin{aligned}
&desires(1, exec\_order(O), T, [0.6]).\\
&desires(2, charged, T, [1.0]).
\end{aligned}$$

Beliefs

$$\begin{aligned}
initiates(E, exec\_order(O), T) \leftarrow\ & holds\_at(is\_order(O), T),\\
& holds\_at(O, T).\\
initiates(E, movemail(Orig, Dest, Mail), Tf) \leftarrow\ & \\
& holds\_at(took\_mail(Mail,\\
& Orig), Ti),\\
& happens(E, Ti, Tf),\\
& act(E, leave(Mail, Dest)).\\
initiates(E, took\_mail(Mail, Origin), Tf) \leftarrow\ & happens(E, Ti, Tf),\\
& act(E, get(Mail, Orig)).\\
terminates(E, took\_mail(Mail), Tf) \leftarrow\ & happens(E, Ti, Tf),\\
& act(E, leave(Mail, Dest)).\\
initiates(E, charged, Tf) \leftarrow\ & holds\_at(low\_charge, T),\\
& happens(E, Ti, Tf),\\
& act(E, full\_charge).\\
terminates(E, charged, Tf) \leftarrow\ & senses(low\_charge, Ti, Tf).\\
initially(charged).\ & \\
initially(is\_order(movemail(A, B, C)).\ & \\
\perp \Leftarrow\ & exec\_order(O), bat\_charged.
\end{aligned}$$

Agents choose their intentions from two different sources: from its desires and as a refinement from other intentions. We start by defining the creation of intentions from desires.

Intentions as Selected Desires By definition, there are no constraints on the agent's desires. Therefore, an agent may have contradictory desires, i.e., desires that are not jointly achievable. Intentions, on the other hand, are restricted by rationality constraints (as shown in the previous section). Thus, agents must select only those desires that conform to those constraints. We start by defining those desires that are eligible to be chosen and the notion of candidate desires set.

Definition 14 (Eligible Desires). Let $\mathcal{D}$ be the agent's desires. We call *eligible desires* the set

$$\begin{aligned}
\mathcal{D}' = \{desires(D, P, T, A)/\ & [(desires(D, P, T, A) \leftarrow Body) \in \mathcal{D}] \wedge\\
& Now \leq T\\
& (\mathcal{B} \models_P Body)\}
\end{aligned}$$

Eligible desires are those desires the agent believes are not satisfied. Recall that, according to rationality constraint 1 – 2 in section 4, its is not rational for an agent to intend something it believes is already achieved or that is impossible. Notice that if a desire is conditional, then the agent should believe this condition is true.

As the initial set of desires, eligible desires may also be contradictory. Therefore, it is necessary to determine those subsets of the eligible desires that are jointly achievable. In general, there may be more than one subset of the eligible desires that are jointly achievable. Therefore, we should indicate which of these subsets are preferred to be adopted as intentions. We do this through the preference relation defined bellow.

Definition 15 (Desires Preference Relation $<_{Pref}$). Let $\mathcal{D}$ be the agent's desires, $\mathcal{D}'$ the set of eligible desires from $\mathcal{D}$, $\mathcal{P}(\mathcal{D}')$ the power set of $\mathcal{D}'$ and $R, S \in \mathcal{P}(\mathcal{D}')$; $imp(R)$ the set of importances of the desires subset R. We say that $R <_{Pref} S$ iff, for $T = max(imp(R))$ such that $\nexists X \in imp(S).(X = T)$ and $T' = max(imp(S))$ such that $\nexists Y \in imp(R).(Y = T')$,

- $T' < T$; or
- $(\nexists T \vee \nexists T') \wedge (\#S < \#R)$.

We say that R is an *antecedent* of S iff $\nexists T \in \mathcal{P}(revisable).[(R <_{Pref} T) \wedge (T <_{Pref} S)]$

According to this definition, the agent should prefer to satisfy first the most important desires. Additionally to preferring the most important ones, the agent adopts as much desires as it can.

Example 16. Given the set

$$D' = \{desires(1, a, U, [0.5]), desires(2, b, U, [0.3]), \\ desires(3, c, U, [0.3]), desires(4, d, U, [0.2])\}$$

of eligible desires:

- for $d_1 = \{desires(2, b, U, [0.3]), desires(4, c, U, [0.2])\}$ and $d_2 = \{desires(1, a, T, [0.5])\}$,we have $(d_2 <_{Pref} d_1)$. That is, d_2 is less preferred than d_1, since $T = 0.5$ for d_2 and $T' = 0.3$ for d_1;
- for $d_3 = \{desires(3, c, U, [0.3])\}$, we have $d_3 <_{Pref} d_1$, as $T = 0.2$ for d_1 and there is no T' for d_3, but $\#d_3 < \#d_1$.

Notice that the preference relation is not an order relation. For instance, if an agent were to choose between $d_4 = \{desires(2, b, U, [0.3])\}$ and $d_5 = \{desires(3, c, U, [0.3])\}$, based only on the importance of desires and maximization of desires satisfied, it would not prefer either of them. And, indeed, according to the preference relation, we have that neither $(d_4 <_{Pref} d_5)$ nor $(d_5 <_{Pref} d_4)$.

Based on this preference order, we define the preference graph that will be used to revise the mental states and the revision process.

Definition 17 (Desires Preference Graph). Let $\mathcal{D}$ be the agent's desires and $\mathcal{D}'$ the set of eligible desires from $\mathcal{D}$. Let *Revisable* be the set $\{unsel(D)$ / $\exists desires(D,P,T,A) \in \mathcal{D}'\}$ and $index : \mathcal{P}(Revisable) \longrightarrow \aleph^+$ a function from the power set of *Revisable* to natural numbers (zero excluded) that attributes a level number to elements of $\mathcal{P}(Rev)$. The *desires preference graph* is the graph defined by

1. $Rev(\mathbf{bottom}) = \{happens(E,T_i,T_f), act(E,A)\}$;
2. $Rev(i) = R \cup \{happens(E,T_i,T_f),\ act(E,A)\}$, where $R \in \mathcal{P}(Revisable)$ and $i = index(R)$;
3. $i \ll \mathbf{bottom}$, where $i = index(R)$, $R \in \mathcal{P}(Revisable)$ and $\nexists S \in \mathcal{P}(Revisable)$. $(S <_{Pref} R)$;
4. $j \ll k_1,\ldots,k_n$, where $j = index(R)$, $k_i = index(S_i)$ $(1 \leq i \leq n)$, $R, S_i \in \mathcal{P}(Rev)$ $(1 \leq i \leq n)$ and R is an antecedent of S.

For the example 16, the preference graph for eligible desires would be:

$$Rev(\mathbf{bottom}) = \{happens/3, act/2\} \qquad Rev(1) = \{unsel(4)\}$$
$$Rev(2) = \{unsel(2)\} \qquad Rev(3) = \{unsel(3)\}$$
$$Rev(4) = \{unsel(2), unsel(4)\} \qquad Rev(5) = \{unsel(3), unsel(4)\}$$
$$Rev(6) = \{unsel(2), unsel(3)\} \qquad Rev(7) = unsel(2), unsel(3), unsel(4)$$
$$Rev(8) = \{unsel(1)\} \qquad Rev(9) = \{unsel(1), unsel(4)\}$$
$$Rev(10) = \{unsel(1), unsel(3)\} \qquad Rev(11) = \{unsel(1), unsel(2)\}$$
$$Rev(12) = \{unsel(1), unsel(3), unsel(4)\} \qquad Rev(13) = \{unsel(1), unsel(2), unsel(4)\}$$
$$Rev(14) = \{unsel(1), unsel(2), unsel(3)\} \quad Rev(15) = \{unsel(1), unsel(2), unsel(3), unsel(4)\}$$

$$\begin{array}{ccc}
1 \ll \mathbf{bottom} & 2 \ll 1 & 3 \ll 1 \\
4 \ll 2,3 & 5 \ll 2,3 & 6 \ll 4 \\
6 \ll 5 & 7 \ll 6 & 8 \ll 6 \\
9 \ll 8 & 10 \ll 9 & 11 \ll 9 \\
12 \ll 11,10 & 13 \ll 11,10 & 14 \ll 13 \\
14 \ll 12 & 15 \ll 14 &
\end{array}$$

and every $R(i)$ united with $Rev(\mathbf{bottom})$.

That is, when revising the eligible desires set, the preferred revisions are those that eliminate first the less important desires, and the least possible amount of desires, as shown in the definition bellow.

Definition 18 (Candidate Desires Set). Let $\mathcal{D}$ be the agent's desires and $\mathcal{D}'$ the set of eligible desires from $\mathcal{D}$ with a preference graph associated to it. We call *candidate desires set* any set

$$\begin{aligned}\mathcal{D}'_C = \{desires(D,P,T,A)/\ & (desires(D,P,T,A) \in \mathcal{D}')\wedge \\ & [\exists \Delta.(\mathcal{B} \cup \Delta \models_P (holds\_at(P,T), not\ unsel(D)))\wedge \\ & (P' \cup \Delta \not\models_P \bot)]\}\end{aligned}$$

where

– P' is the abductive framework $\langle \mathcal{B}, \{happens(E,T_i,T_f), act(E,Act), unsel(D)\}, IC\rangle$;
– IC is a set of constraints of the form
  - $\{holds\_at(P,T) \Leftarrow Body, not\ \ unsel(D)\}$ for every $desires(D,P,T,A)$ in $\mathcal{D}'$;

- the constraints $IC(\mathcal{I})$ generated by intentions (see definition 12).

In the revision process, we mix abductive reasoning with defeasible reasoning. In fact, the constraints are defeasible and the literal $unsel(D)$ plays the same role as $ab(X)$ in example 8. Its intuitive meaning is *"Desire D should not be selected as an intention"*. If the agent believes it is possible to satisfy all of its desires (if it can abduce actions that satisfy all desires and satisfy all constraints), it will find a revision that contains only *happens*/3 and *act*/2. This is enforced by the preference graph with $Rev(\mathbf{bottom}) = \{happens(E, T_i, T_f), act(E, A)\}$.

When constraints may not be all concurrently satisfied, it means that the adoption of all desires as intentions leads to contradictions, i.e., they are not jointly satisfiable.

Notice that contradictions do not arise if the actions necessary to satisfy two different intentions have contradictory effects. Recall that, according to the EC axioms, a property P holds if there is an action that initiates it or, alternatively, if $\neg P$ implicitly does not hold, and vice-versa for $\neg P$. Therefore, actions that make contradictory properties hold in fact just cancel each other. This allows us to avoid the *side-effect problem*[Bra90]. If we allowed for this kind of situations to raise contradictions, we would be making the agent preclude intentions that have contradictory consequences, but that are not directly contradictory with each other, and making the agent intend the logical consequences of its intentions.

On the other hand, if an action necessary to satisfy an intention cancels a property that is also an intention, a constraint is violated and that course of action is rejected. In this case, the revision will try to defeat intentions, changing the truth value of $unsel(D)$ literals. According to the preference graph, it will try to defeat those constraints that represent the less important desires, trying to preserve the maximum of the most important ones.

As we mentioned before, the desires preference relation is not an order relation. Therefore, it is possible to have more than one candidate set after a revision. However, if we consider only achievability and desires atributes as decision criteria, it makes no difference for the agent to adopt any of the candidate desires set[5][JLS93][**?**].

Definition 19 (Primary Intentions). Let $\mathcal{D}$ be the agent's desires, $\mathcal{D}'_C$ a candidate desires set from $\mathcal{D}$. The *primary intentions* of an agent is the set

$$\{intends\_that(D, P, T, A)/desires(D, P, T, A) \in Des'_C)\}$$

Example 20 ((cont. from 13)). Suppose the robot has received two orders, *movemail* (101, 105, *letter*) and *movemail* (102, 108, *magazine*). Initially, the robot (who had no intentions) is going to select, among its desires, its intentions. Its eligible desires are

$$desires(1, exec\_order(movemail(101, 105, letter)), T, [0.6]).$$
$$desires(2, exec\_order(movemail(102, 108, magazine)), T, [0.6]).$$
$$desires(3, bat\_charged, T, [1.0]).$$

[5] The revision process provided by the ELP framework defines a *sceptical revision*[AP96], that is the revision formed by the union of all the minimal program revision. This kind of approach prevents the agent from having to choose one of the minimal revisions. However, for intentions, this is not adequate, since we would like our agents to try to satisfy all the eligible desires it can.

as all of then have their (null) bodies true and are in time. It is now necessary to verify which desires can be jointly satisfied, computing the candidate desires sets. The preference graph derived from the importance attribute is

$$\begin{array}{rr} Rev(\mathbf{bottom}) = \{happens/3, act/2\} & Rev(1) = \{unsel(1)\} \\ Rev(2) = \{unsel(2)\} & Rev(3) = \{unsel(1), unsel(2)\} \\ Rev(4) = \{unsel(3)\} & Rev(5) = \{unsel(3), unsel(1)\} \\ Rev(6) = \{unsel(3), unsel(2)\} & Rev(7) = \{unsel(1), unsel(2), unsel(3)\} \end{array}$$

$$\begin{array}{ccc} 1 \ll \mathbf{bottom} & 2 \ll \mathbf{bottom} & 3 \ll 1, 2 \\ 4 \ll 3 & 5 \ll 4 & 6 \ll 4 \\ 7 \ll 6, 5 & & \end{array}$$

with every $Rev(i)$ united with $Rev(\mathbf{bottom})$. There is one candidate desires set, namely

$$\begin{aligned} D'_C = \{&desires(1, exec\_order(movemail(101, 105, letter)), T, [0.6]), \\ &desires(2, exec\_order(movemail(102, 108, magazine)), T, [0.6])\} \end{aligned}$$

as the robot believes it can perform the following sequence of actions

$$\begin{array}{ll} happens(E, T_1i, T_1f) & act(E, get(101, letter)) \\ happens(E, T_2i, T_2f) & act(E, get(102, magazine)) \\ happens(E, T_3i, T_3f) & act(E, leave(letter, 105)) \\ happens(E, T_4i, T_4f) & act(E, leave(magazine, 108)) \end{array}$$

$$T_1i < T_1f < T_2i < T_2f < T_3i < T_3f < T_4i < T_4f$$

that brings about both desires. Notice that the robot does not select "charged" to be an intention because its initiation depends on the battery being low, which is not the case. Thus, its primary intentions will be that only candidate desires set.

Intentions as Refinements from Intentions Once the agent adopts its intentions, it will start planning to achieve those intentions. During planning, the agent will form intentions that are relative to pre-existing intentions. That is, they "refine" their existing intentions. This can be done in various ways, for instance, a plan that includes an action that is not directly executable can be elaborated by specifying particular way of carrying out that action; a plan that includes a set of actions can be elaborated by imposing a temporal order on that set[KP93]. Since the agent commits to the adopted intentions, these previously adopted intentions constrain the adoption of new ones. That is, during the elaboration of plans, a potential new intention is only adopted if it is not contradictory with the existing intentions and with beliefs.

Definition 21 (Planning Process, Relative Intentions). Let $\mathcal{I}_P$ be the set of primary intentions. A *planning process* is a procedure that, for each $i \in \mathcal{I}_P$, will generate a set of temporal ordered actions $\mathcal{I}_R$ that achieve i, such $\mathcal{B} \cup \mathcal{I}_P \cup \mathcal{I}_R$ is non-contradictory. The set $\mathcal{I}_R$ are the *relative intentions* of the agent.

The non-contradiction condition enforces again the notion of commitment, i.e., once an intention is adopted it constrains the adoption of new intentions.

4.3 Revising Intentions

In the previous section, we defined how the agent chooses its intentions. As we have seen, weighing motivations and beliefs means finding inconsistencies in competing desires, checking valid desires according to beliefs and intentions, resolving constraints imposed by intentions and desires, i.e., very expensive reasoning activities. It is now necessary to define *when* the agent should perform this process.

We argue that it is not enough to state that an agent should revise its intentions when it believes a certain condition holds, like to believe that an intention has been satisfied or that it is no longer possible to satisfy it, as this suggests that the agent needs to verify its beliefs constantly. Instead, we take the stance that it is necessary to define, along with those conditions, a mechanism that triggers the reasoning process without imposing a significant additional burden on the agent. Our approach is to define those conditions that make the agent start reasoning about intentions as constraints over its beliefs. Recall that we assume that an agent constantly has to maintain its beliefs consistent, whenever new facts are incorporated.

Definition 22 (Trigger Constraints from Intentions). Let $\mathcal{B}$ be the agent's beliefs set and $\mathcal{I}$ its intentions. We add to $\mathcal{B}$ the following *trigger constraints*:

- $(\bot \Leftarrow Now > T, not\, rev\_int)$, for each $(intends_that(\, I,P,T,A), intends_to(I,Act,T,A)) \in \mathcal{I}$;
- $(\bot \Leftarrow happens(E,T_i,T_f), act(E,Act), not\ rev\_int$, for each $intends_to(I,Act,T,A)) \in \mathcal{I}$;
- no other constraint in $\mathcal{B}$ are trigger constraints.

The literal rev_int is part of the revisable set of beliefs, and its initial value is $false$. Whenever the agent revises its beliefs and one of the conditions for revising beliefs hold, a contradiction is raised. We identify such contradiction by testing if rev_int is in the selected revision for the beliefs set, i.e., if it has to have its truth value modified in order to restore consistency. The intention revision process is triggered when one of these constraints is violated.

Trigger Constraints from Desires The conditions we have defined so far are the usual ones defined by formal models of agents. As we have seen before, this characterization of intentions may lead to some fanatical behavior. Therefore, we need to adopt additional constraints that will avoid those unwanted behaviors.

We take the stance that the same reasons that originated intentions may be used to break commitment associated to them[Bra92]. If we accept that an intention is originated from desires, it is reasonable to state that it is not rational to persist with an intention whose reasons are superseded by more urgent or important ones. The agent's normal behavior would be to weigh its competing desires and beliefs, selecting its intentions. The agent would commit to these intentions and they would constitute the filter of admissibility for other intentions. Also, the agent would try to satisfy those intentions, until successful accomplishment, impossibility (as usually defined), or *until some of his other desires that were not selected before would become eligible*, or *until the desires that originated them would not be eligible anymore*, re-activating the revision process that would weigh (again) competing desires and beliefs.

Since the notion of commitment is preserved, the agent would not start this process every time there is a change in beliefs, but only if relevant conditions trigger the intention revision process, changing the agent's focus of attention[BC96]. These triggers are determined by the desires pre-conditions. We model this triggers using an approach similar to the normative constraints in [SdAMP97,Wag96].

Definition 23 (Trigger Constraints from Desires). Let $\mathcal{D}$ be the agent's desires and $\mathcal{D}'$ the set of eligible desires from $\mathcal{D}$. We define *trigger constraints from desires* as

1. For every $desires(D,P,T,A) \leftarrow Body \in (\mathcal{D} - \mathcal{D}')$ with importance A bigger that the biggest importance in intentions, we define a trigger constraint
$$\perp \Leftarrow Body, not\ rev\_int$$
2. given the set of actions Δ abduced by the agent (see definition 18), for each $desires(D,P,T,A) \in (\mathcal{D}' - \mathcal{D}'_C)$ with importance A bigger that the biggest importance in intentions, we define a trigger constraint
$$\perp \Leftarrow C_1, \ldots, C_n, not\ rev_int$$
where $Cond_i$ $(1 \leq i \leq n)$ are the conditions the agent could not bring about when selecting the candidate desires set.

The first constraint trigger is formed by the the pre-conditions of those desires that were not eligible and that are more important than those that were evaluated. It means that if the pre-conditions of such desires become true, these desires (that were not considered during reasoning) become eligible. Therefore, it is necessary to re-evaluate desires and beliefs to check if this new desire may be brought about. The second constraint is formed by the pre-conditions of those eligible desires that, although more important, were not relevant when the agent made his choice.

Notice that there are no triggers for those desires that were eligible but that were ruled out during the choice of a revision. This is so because they had already been evaluated and they have been considered less important than the other desires. Therefore, it is of no use to trigger the whole process again (i.e., to shift the agent's attention) to re-evaluate them.

Example 24 ((cont. example 13)). Suppose that, while executing his orders, the robot senses a low battery condition. We would expect him to interrupt his tasks and proceed to recharge. Indeed, as "charged" was an eligible desire but it was not selected as a candidate desire, a trigger constraint of the form

$$\perp \Leftarrow holds\_at(low\_charge, T), not\ rev_int$$

is added to the robot beliefs set. When it updates its beliefs with "*senses* (*low_charge*, T_i, T_f)", the constraint is violated, rev_int is the revision that restores consistency and, therefore, the role reasoning procedure to decide what to do starts again.

5 Conclusion and Further Work

The main contribution of this paper is to provide a formal model of intentions that accounts for both static and dynamic aspects of intentions. By static aspects, we mean its relation with desires and beliefs and the properties that this relation has (namely, consistency and avoidance of side-effects). By dynamic aspects, we mean establishing criteria, other than those derived from commitment, to drop intentions (namely, based on the cause-effect relation between desires and intentions). We also define where intentions come from and how the agent chooses its intentions.

We may compare our work to other models of intentions in the literature. The classical model is due to Cohen and Levesque[CL90]. In this model, the authors do not address the question of where and how the agent chooses its intentions, neither they overcome the fanatical behavior caused by the commitment strategy. The same may be said about other models that followed Cohen and Levesque's and that tried to overcome some of its theoretical problems: Rao and Georgeff[RG91] tried to solve the side-effect problem present in [CL90] introducing the notions of strong and weak realism; Konolidge and Pollack[KP93] also tried to solve the side-effect problem using non-normal modal logics, where the solution were embedded in the semantics; Singh[Sin94] introduces the notion of know-how and defines communication and multi-agent interaction using its model, but he does not treat the question of the origins of intentions nor how to relax commitment. Van Linder et ali[vLvdHM96] model the origin and choice of intentions and provide a formalism that can be extended to model other pro-active attitudes, but they do not provide any improvements on the notion of commitment.

All these works have also in common the fact that their models are based on modal logics and, therefore, may be just used to specify agents. We are, on the other hand, heading to define a computational theory of agency that has an underlying architecture, the one partially described here. This computational theory will allow us to specify and prove properties about the agents, as well as to execute such an specification to test if the agents behave like expected. Our choice of the logical formalism has been deeply influenced by this long term goal, since it is formal and computationally tractable. This concern with the computational aspects of the agent models can also be found in [Rao96].

Our work has many similarities with the one of Gaspar et ali[GC95]. Although they use a different formalism (that is not so computationally tractable as ours), both works have many objectives in common: to address the question of the origin of intentions, to have a computational model, to revise intentions. While we concentrate on the relation between desires and intentions and use this relation to relax commitment, they focus on the evolution of plans and communication, and on how this affects the stability of intentions. In order to have a complete treatment of these issues, both aspects are important and must be integrated. This is our next step in the development of our model.

References

[AP96] J.J. Alferes and L.M. Pereira. *Reasoning with Logic Programming.* Springer-Verlag, Berlin, DE., 1996. Lecture Notes in Artificial Intelligence Series (LNAI 1111).

[BC96] L.M. Botelho and H. Coelho. Emotion-based attention shift in autonomous agents. In *Proceedings of the I IberoAmerican Workshop on Distributed Artificial Intelligence (IWDAI'96)*, Xalapa, Mexico, 1996.

[Bea94] L. Beaudoin. *Goal Processing in Autonomous Agents.* PhD thesis, Birmingham University, Birmingham, UK, August 1994.

[Bra87] M.E. Bratman. *Intentions, plans and practical reasoning.* Harvard University Press, Cambridge, MA, 1987.

[Bra90] M.E. Bratman. What is intention? In P.R. Cohen, J.L. Morgan, and M. Pollack, editors, *Intentions in Communication*, chapter 1. The MIT Press, Cambridge, MA, 1990.

[Bra92] M. Bratman. Planning and the stability of intentions. *Minds and Machines*, 2:1–16, 1992.

[CL90] P.R. Cohen and H.J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42:213–261, 1990.

[DNP94] C. Damásio, W. Nejdl, and L.M. Pereira. Revise: An extended logic programming system for revising knowledge bases. In *Knowledge Representation and Reasoning*. Morgan Kaufmann inc., 1994.

[GC95] G. Gaspar and H. Coelho. Where do intentions come from?:a framework for goals and intentions adoption, derivation and evolution. In C. Pinto-Ferreira and N.J. Mamede, editors, *Proceedings of the Seventh Portuguese Conference on Artificial Intelligence (EPIA'93)*, Berlin, Germany, 1995. APIA, Springer-Verlag. Lecture Notes on Artificial Intelligence (LNAI 990).

[JLS93] P.N. Johnson-Laird and E. Shafir. The interaction between reasoning and decision making: an introduction. *Cognition*, 49:1–9, 1993.

[KP93] K. Konolige and M. Pollack. A representationalist theory of intentions. In *Proceedings of the XII International Joint Conference on Artificial Intelligence (IJCAI'93)*, Chambéry, France, 1993. IJCAI inc.

[Mes92] L. Messiaen. *Localized abductive planning with the event calculus.* PhD thesis, Katholieke Universiteit Leuven, Leuven (Heverlee), 1992.

[QL97] J. Quaresma and J.G. Lopes. A logic programming framework for the abduction of events in a dialogue system. In *Proceedings of the Workshop on Automated Reasoning*, London, England, 1997. AISB Workshop Series.

[Qua97] P. Quaresma. *Inferência de Atitudes em Dialogos.* PhD thesis, Universidade Nova de Lisboa, Lisbon, Portugal, june 1997. In Portuguese.

[Rao96] A.. Rao. Agentspeak(1): BDI agents speak out in a logical computable language. In *Proceedings of the European Workshop on Modelling Autonomous Agents and Multi-Agents Worlds 1996 (MAAMAW'96)*, Berlin, Germany, 1996. Springer-Verlag.

[RG91] A.S. Rao and M.P. Georgeff. Modelling rational agents within a BDI-architecture. In R. Fikes and E. Sandewall, editors, *Proceedings of the Knowledge Representation and Reasoning'91 (KR&R'91)*, San Mateo, CA., 1991. Morgan Kauffman Publishers.

[SdAMP97] M. Schroeder, Iara de Almeida Móra, and Luís Moniz Pereira. A deliberative and reactive diagnosis agent based on logic programming. In J. P. Müller, M. J. Wooldridge, and N. R. Jennings, editors, *Intelligent Agents III — Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages (ATAL-96)*, volume 1193 of *Lecture Notes in Artificial Intelligence*, pages 293–308. Springer-Verlag, Heidelberg, 1997.

[Sea84] J. Searle. What is an intentional state? In H. Dreyfuss and H. Hall, editors, *Husserl, Intentionality and Cognitive Science*, volume 42, pages 213–261. 1984.

[Sin94] M. Singh. *Multiagent systems: a theoretical framework for intentions, know-how, and communications.* Springer-Verlag, Heidelberg, Germany, 1994. Lecture Notes in Artificial Intelligence (LNAI 799).

[vLvdHM96] B. van Linder, W. van der Hoek, and J. J. Ch. Meyer. Formalizing motivational attitudes of agents: On preferences, goals, and commitments. In M. Wooldridge, J. P. Müller, and M. Tambe, editors, *Intelligent Agents II — Proceedings of the Second International Workshop on Agent Theories,* *Architectures, and Languages (ATAL-95)*, volume 1037 of *Lecture Notes in Artificial Intelligence*, pages 17–32. Springer-Verlag, Heidelberg, 1996.

[Wag96] G. Wagner. A logical and operational model of scalable knowledge and perception-based agents. In *Proceedings of the European Workshop on Modelling Autonomous Agents and Multi-Agents Worlds 1996 (MAA-MAW'96)*, Berlin, Germany, 1996. Springer-Verlag.

6 EIS98 - "Modelling Agents with Extended Logic Programming"

Este artigo, apresentado no EIS98[MÓR 98b], enfatiza a viabilidade e as vantagens de se construir modelos formais de agentes utilizando, ao invés das lógicas modais tradicionais, um formalismo que seja tratável computacionalmente. Além de permitir a definição formal dos agentes, tal modelo pode ser usado diretamente como ferramenta de implementação destes agentes. O formalismo utilizado é a Programação em Lógica Estendida. O Modelo de Agentes aqui apresentado está já completo. Inicia-se, aqui, a discutir as vantagens e desvantagens deste modelo, notadamente: a facilidade de descrição de propriedades e agentes, os mecanismos de prova para verificação de propriedades, sua correção e completude, e os passos necessários para aproximá-lo de arquitetura de agentes.

Modelling Agents with Extended Logic Programming

Michael C. Móra*
CPGCC/II/UFRGS
DI/FCT/UNL

José Gabriel Lopes†
DI/FCT/UNL

Rosa M. Viccari‡
CPGCC/II/UFRGS
CBL/Leeds Univ.

Helder Coelho§
DI/FCUL/UL

Abstract

Much effort has been put in the formalization of rational agents. Through the use of modal logics and antropomorphic notions as mental states, these models characterize the behavior of rational agents. Nevertheless, the design of agent-oriented systems has not had much benefit from these specifications. The main reason for this is the gap that exists between methodologies for specification and for design. In this paper, we take some steps trying to reduce the gap that exists between agent specification and agent implementation. Our approach is to provide a framework to define agents that presents the benefits of specification frameworks — namely providing the tools to formally define agents and to reason about these agents — and that also overcomes their main problem — the absence of computational procedures to deal with the formalism that would allow us to build agents using such a formalism. We achieve this using a logical formalism that has a well-defined proof procedure and that provides tools necessary to model mental states, namely extended logic programming (ELP) with the WFSX (Well-Founded Semantics eXtended) semantics and its corresponding SLX derivation procedure.

*Assistent professor at II/PUCRS, in Brazil. Currently a PhD student at CPGCC/UFRGS, in Brazil. Supported by CAPES/Brazil, PUC/RS and project DIXIT/JNICT. Address: CPGCC/UFRGS; Bloco IV – Campus do Vale; Av. Bento Gonçalves 9500 – Porto Alegre – RS – Brasil. E-mail: michael@inf.ufrgs.br .

†Researcher at the CENTRIA/DI/UNL, in Portugal. Supported by project DIXIT/JNICT. Address: DI/FCT/Universidade Nova de Lisboa; Quinta da Torre – 2825 Monte da Caparica – Portugal. E-mail: gpl@di.fct.unl.pt .

‡Professor at the CPGCC/UFRGS, in Brazil. Supported by CNPq/Brazil. Currently at the CBL group, Leeds University. E-mail: rosav@cbl.leeds.ac.uk.

§Professor at the DI/FCUL/UL. Address: DI/FCUL/ Universidade de Lisboa; Bloco C5, Piso 1 – Campo Grande – 1700 Lisboa – Portugal. E-mail: hcoelho@di.fc.ul.pt .

Keywords: intentions; mental states modeling; agent architectures; distributed artificial intelligence; logic programming.

1 Introduction

The formalization of rational agents is a topic of great interest in Artificial Intelligence. The purpose of such formal definitions is to characterize the behavior of agents using antropomorphic notions, as mental states and actions the agent performs to achieve its purpose. These notions and their properties are formally defined using logical frameworks that allow theorists to analyze, to specify and to verify rational agents, like in [8, 17, 20, 22, 16]. Nevertheless, the design of agent-oriented systems has not had much benefit from these specifications. The main reason for this is the gap that exists between methodologies for specification and for design.

Current specification frameworks are based on multi-modal logics, where each modality defines a mental state like *beliefs*, *desires* and *intentions* (BDI-logics). Although very powerful, usually these logics are not computationally tractable, i.e., there is no effective constructive derivation procedure to verify satisfiability of formulas. This prevents the use of such logics by the agents as a reasoning mechanisms. Agent design methodologies, on the other hand, follow a *programming perspective*[21, 14], i.e., they usually rely on non-formal architectures that arguably implement the specified characteristics of the agents. Apart from this argued relation, usually there are no other concerns on formal relation between specifications and implementations.

In this paper, we take some steps trying to reduce the gap that exists between agent specification and agent implementation. Our approach is to provide a framework to define agents that presents the benefits of specification frameworks — namely providing the tools to formally define agents and to reason about these agents — and that also overcomes their main problem — the absence of computational procedures to deal with the formalism. We achieve this using a logical formalism that has a well-defined proof procedure and that provides tools necessary to model mental states, namely *extended logic programming* (ELP) with the $WFSX$ (Well-Founded Semantics eXtended) semantics.

ELP with WFSX (simply ELP, from now on) extends normal logic programs with a second negation named *explicit*[1] *negation*. This extension allows us to explicitly represent negative information (like a belief that a property P does not hold, or an intention that a property Q should not hold) and increases the expressive power of the language.

However, when we introduce negative information we may have contradictory programs[1]. Indeed, when we represent mental states like desires and beliefs, it is necessary to model and to deal with contradiction, as shown in the following example.

[1]In contrast with the usual *negation as failure* or *negation by default* of normal logic programs, which is called *implicit negation* in the ELP context.

Example 1.1 *Mike has two distinct programmes for his evening. He may go to the local opera house or to a soccer game, at the local stadium. Both events start at the same time. Suppose the program $P_{1.1}$ bellow represents this problem*[2]*:*

$$\begin{array}{rcl} go(opera, t1) & & \\ go(soccer, t1) & & \\ \bot & \Leftarrow & go(opera, t1), go(soccer, t1) \end{array}$$

According to the example above, Mike cannot choose to go both to the opera and to the soccer game, as both start at the same time. The fact that he cannot be at two different places at the same time is represented by the integrity constraint that raises a contradiction $\bot$.

Besides providing the language to model this kind of situations,along with the computational proof procedure for theories expressed in this language, the ELP framework also provides a mechanism to determine how to minimally change a logic program in order to remove contradictions. In the example above, we would have two possible minimal revisions, namely $Rev_1 = \{go(opera, t1) \leftarrow \mathbf{f}\}$ and $Rev_2 = \{go(soccer, t1) \leftarrow \mathbf{f}\}$. That is, we can have only one the two predicates $go/2$ true in order to avoid contradictions. The agent would have to choose one of them. Our model benefits from these features provided by the logical formalism.

As it is usually done[8, 21, 11], we focus on the formal definition of mental states and on how the agent behaves, given such mental states. But, contrasting with these former approaches, our model is not only an agent specification, but it may also be executed in order to verify the actual agent behavior, as well as it may be used as reasoning mechanism by actual agents.

We depart from Bratman's analysis[7], where he states that, along with desires and beliefs, intentions is a fundamental mental state. Therefore, initially we define these three mental states and the static relations between them, namely constraints on consistency among those mental states. Afterwards, we advance with the definition of dynamic aspects of mental states, namely how the agent chooses its intentions, and when and how it revises its intentions[3].

The paper is organized as follows: in section 2, we present the ELP formalism and the reasoning mechanisms it provides; in section 3, we present an action and time theory based on a variation of the event calculus[15] that is used to define the mental states; in section 4, we present our agent model; in section 5, we discuss some related and future work and draw some conclusions.

[2]The symbol $\bot$ stands for *contradiction*

[3]As we stated before, our concern is the definition of agents' behavior in terms of their mental states. Therefore, its beyond the scope of this work to treat dynamic aspects of beliefs, as belief revision. Notice, however, that opposed to models like the ones defined in [8, 21, 11] and others, the logical grounds on which we base our definitions are suitable to define belief revision strategies and to be used by agents as a reasoning tool for truth maintenance. See [2, 13] for two different approaches on this matter

2 Extended Logic Programming

Differently of normal logic programs, where negative information is only stated implicitly (i.e., a proposition is false only if it cannot be proved to be true), extended logic programs have a second kind of negation that allows us to explicitly represent negative information. If, on one hand, this second negation increases the representational power of the language, it may also introduce contradictory information in programs. Therefore, it is necessary to assign some meaning to contradictory programs and to be able to deal with contradiction. The paraconsistent version of the $WFSX$ semantics, $WFSX_P$, assigns meaning to contradictory programs and is used as a tool by the revision mechanism that restores consistency to contradictory programs. This revision mechanism may be used to perform several kinds of reasoning, as we will see bellow.

An extended logic program (ELP) is a set of rules $H \leftarrow B_1, \ldots, B_n$, *not* $C_1, \ldots,$ *not* C_m (m,n $\geq$ 0), where $H, B_1, \ldots, B_n, C_1, \ldots, C_m$ are objective literals. An objective literal is either an atom A or its explicit negation $\neg A$. The symbol *not* stands for negation by default and *not* L is a default literal. Literals are either objective or default literals and $\neg\neg L \equiv L$. The language also allows for integrity constraints of the form $A_1 \vee \ldots \vee A_l \Leftarrow B_1, \ldots, B_n,$ *not* $C_1, \ldots,$ *not* C_m (m,n ≥ 0; l ≥ 1) where $A_1, \ldots, A_l, B_1, \ldots, B_n, C_1, \ldots, C_m$ are objective literals, stating that at least one of the A_i (i ≥ 1), should hold if its body $B_1, \ldots, B_n,$ *not* $C_1, \ldots,$ *not* C_m (m,n ≥ 0) holds. Particularly, when $A = \perp$, where $\perp$ stands for *contradiction*, it means that a contradiction is raised when the constraint body holds. The set of all objective literals of a program P is the extended Herbrand base of P denoted by $\mathcal{H}(P)$.

In order to assign meaning to ELP programs, we need to the notions of interpretation of ELP clause[4]. Roughly, an interpretation of an ELP program is a set of literals that verifies the coherence principle: if $\neg L \in T$ then $L \in F$. A literal L is true in an interpretation I if $L \in I$, L is false in I if *not* $L \in I$, and L is undefined, otherwise. An interpretation I is a model for program P iff, for every clause L in P, $I \models_P L$.

2.1 Program Revision and Non-Monotonic Reasoning

As we stated before, due to the use of explicit negation, programs may be contradictory. In order to restore consistency in programs that are contradictory with respect to $WFSX$, the program is submitted to a revision process that relies on the allowance to change the truth value of some set of literals. This set of literals is the set of *revisable literals*, and can be any subset of $\mathcal{H}$ for which there are no rules or there are only facts in the program. No other restriction is made on which literals should be considered revisable. They are supposed to be provided by the user, along with the program.

The revision process changes the truth value of revisable literals in a minimal way and in all alternative ways of removing contradiction. *Minimally revised*

[4]For a formal definition, see [1].

programs can be defined as those programs obtained from the original one after modifying the subsets of revisable literals.

The ability to represent negative information, along with a well-defined procedure that restores consistency in logic program, makes ELP suitable to be used to perform different forms of non-monotonic reasoning [1]. In particular, we are interested in two of these forms of non-monotonic reasoning, namely *defeasible reasoning* and *abductive reasoning*.

2.1.1 Defeasible Reasoning

A defeasible rule is a rule of the form *Normally if A then B*. ELP allows us to express defeasible reasoning and to give meaning to a set of rules (defeasible or not) when contradiction arises from the application of the defeasible rules. To remove contradiction, we revise the truth value of revisable literals in the body of the defeasible rules

2.1.2 Abductive Reasoning

Abductive reasoning consists of, given a theory T and a set of observations O, to find a theory Δ such that $T \cup \Delta \models O$ and $T \cup \Delta$ is consistent. In the ELP context, an abductive framework P' is a tuple $\langle P, Abd, IC \rangle$, where P is an extended logic program, Abd is the set of abducible literals and IC is the set of integrity constraints. An observation O has an abductive explanation Δ iff $P' \cup \Delta \models_P O$ and $P \not\models_P \Delta$. We may abduce a theory Δ that explains such observations making each of these observations O a new integrity constraint $O \Leftarrow$ and revising the program with revisable set Abd (recall that a program is contradictory iff for some literal L in program P, $P \models L$ and $P \models \neg L$, or a constraint is not satisfied).

2.1.3 Preferred Revisions

Sometimes, it may be necessary to state that we do not only want the minimal revisions provided by the formalism, but also that we prefer revisions that have a certain fact only after finding that no other revisions including other facts exist. The ELP formalism allows us to express preferences over the order of revisions using a labeled directed acyclic and/or graph defined by rules of the form[9]:

$$Level_0 \ll Level_1 \wedge Level_2 \wedge \ldots Level_n (\text{n} \geq 1)$$

$Level_i$ nodes in the graph are preference level identifiers. To each of these levels is associated a set of revisables denoted by $\mathcal{R}(Level_i)$. Rules like the one above for $Level_0$ state that we want to consider revisions for $Level_0$ only if, for some rule body, its levels have been considered and there are no revisions at any of those levels. The root of the preference graph is the node denoted by **bottom**.

3 The Event Calculus

When we reason about pro-attitudes like desires and intentions, we need to deal with properties that should hold at an instant of time and with actions that should be executed at a certain time. Therefore, in order to represent them and to reason about them, we need to have a logical formalism that deals with actions and time. In this work, we use a modified version of the Event Calculus (EC) proposed in [15]. This version of the EC allows events to have a duration and an identification, instead of being instantaneous and identified to the instant of time the event occurs. As a consequence, events may occur simultaneously. The predicate *holds_at* defining the properties that are true at a specific time is:

$$\begin{array}{rcl}
holds\_at(P,T) & \leftarrow & happens(E,T_i,T_f), \\
 & & initiates(E,T_P,P), T_P < T, \\
 & & T_P >= T_i, persists(T_P,P,T). \\
persists(T_P,P,T) & \leftarrow & not\ clipped(T_P,P,T). \\
clipped(T_P,P,T) & \leftarrow & happens(C,T_{ci},T_{cf}), \\
 & & terminates(C,T_C,P), \\
 & & T_C >= T_{ci}, not\ out(T_C,T_P,T). \\
out(T_C,T_P,T) & \leftarrow & (T \leq T_C); (T_C < T_P).
\end{array}$$

The predicate $happens(E,T_i,T_f)$ means that event E occurred between T_i and T_f; $initiates(E,T,P)$ means that event E initiates P at time T; $terminates$ (E, T, P) means that event E terminates P at time T; $persists(T_P,P,T)$ means that P persists since T_P until T (at least). We assume there is a special time variable Now that represents the present time. Note that a property P is true at a time T ($holds\_at(P,T)$), if there is a previous event that initiates P and if P persists until T. P persists until T if it can not be proved by default the existence of another event that terminates P before the time T. We need additional rules for the relation between not holding a property and holding its negation and we also need to define the relation between the two kinds of negation:

$$\begin{array}{rcl}
holds\_at(\neg P,T) & \leftarrow & \neg holds\_at(P,T). \\
\neg holds\_at(P,T) & \leftarrow & not\ holds\_at(P,T).
\end{array}$$

The predicates that will be abduced need to be related by some integrity rules, namely:

$$\begin{array}{r}
\perp \Leftarrow happens(E,T_{1i},T_{1f}), happens(E,T_{2i},T_{2f}), \\
not(T_{1i} = T_{2i}, T_{1f} = T_{2f}). \\
\perp \Leftarrow happens(E,T_i,T_f),\ not(T_f < T_i). \\
\perp \Leftarrow happens(E,T_i,T_f), not(act(E,A)).
\end{array}$$

that state, respectively, that events cannot be associated to different time intervals, that events cannot have a negative duration and that events must have an associated action.

The EC allows us to reason about the future, by hypothetically assuming a sequence of actions represented by *happens*/3 and *act*/2 predicates and verifying which properties would hold. It also allows us to reason about the past. In order to know if a given property P holds at time T, the EC checks what properties remain valid after the execution of the actions that happened before T. We are now ready to define the agent's model.

4 The Agent Model

Mental states are divided in two major categories: *information attitudes* and *pro-attitudes*[19]. Information attitudes relate to the information an agent has about the environment where it lives, those that tend to reflect the state of the world, precisely *knowledge* and *beliefs*. Pro-attitudes are those that are related to actions and somehow lead the agent to act, that tend to make the agent modify the world in order to fit its mental states, i.e., attitudes like desires, intentions, obligations and so on. Very often, in the philosophical literature (see [19], for instance), desires and beliefs have been used as the two basic mental states to define an intelligent agent. All the explanations of rational actions, and consequently all the other mental states, could be reduced to them. Nevertheless, according to Bratman[7], a third mental state, *intentions*, should be considered. In his detailed analysis, where he describes the relation among those three mental states — beliefs-desires-intentions (BDI), Bratman argues that, since agents are assumed to be resource-bounded, they cannot continuously evaluate their competing beliefs and desires in order to act rationally. After some reasoning, agents have to commit to some set of choices. It is this choice followed by a commitment that characterizes the intentions.

We start by defining desires. Desires are related to the state of affairs the agent eventually wants to bring about. But desires, in the sense usually presented, does not necessarily drive the agent to act. That is, the fact of an agent having a desire does not mean it will act to satisfy it. It means, instead, that before such an agent decides what to do, it will be engaged in a reasoning process, confronting its desires (the state of affairs it wants to bring about) with its beliefs (the current circumstances and constraints the world imposes). It will choose those desires that are possible according to some criteria it will act upon them. In other words, desires constitute the set of states among which the agent chooses what to do. Notice that, since agents are not committed to their desires, they need not to be consistent, neither with other desires nor with other mental states.

Definition 4.1 (Desires Set) *The* desires *of an agent is a set $\mathcal{D}$ of ELP sentences of the form $desires(D, P, T, A) \leftarrow Body$, where D is the desire identification, P is a property, T is a time point and A is list of attributes. Body is any conjunction of literals. An agent desires that a property P holds at time T iff $desires(D, P, T, A) \in \mathcal{D}$, for some D and some A.*

Our definition of desires allows the agent to have a desire that a certain property holds (or does not hold) in a specific instant of time (when T is instantiated). $Desires/4$ clauses may be facts, representing states the agent may want to achieve whenever possible, or rules, representing states to be achieved when a certain condition holds. The attributes associated to each desire define properties, like urgency, importance or priority[3] that are used by to agent to choose the most appropriate desire (see bellow).

Beliefs constitute the agent's information attitude. They represent the information agents have about the environment and about themselves. Such information includes time axioms and action descriptions (see 3, along with agent capabilities.

Definition 4.2 (Beliefs Set) *The* beliefs *of an agent is a consistent extended logic program $\mathcal{B}$, i.e. $\forall b \in \mathcal{B}.\mathcal{B} \not\models_P b$. An agent believes L iff $\mathcal{B} \models_P L$.*

We assume that the agent continuously updates its beliefs to reflect changes it detects in the environment. Describing the belief update process is beyond the scope of this paper. We just assume that, whenever a new belief is added to the beliefs set, consistency is maintained.

As we stated before, intentions are characterized by a *choice* of a state of affairs to achieve, and a *commitment* to this choice. Thus, intentions are viewed as a compromise the agent assumes with a specific possible future. This means that, differently from desires, an intention may not be contradictory with other intentions, as it would not be rational for an agent to act in order to achieve incompatible states. Also, intentions should be supported by the agent's beliefs. That is, it would not be rational for an agent to intend something it does not believe is possible. Once an intention is adopted, the agent will pursue that intention, planning actions to accomplish it, re-planning when a failure occurs, and so. These actions, as means that are used to achieve intentions, must also be adopted as intentions by agents.

Definition 4.3 (Intentions Set) *The* intentions *of an agent is a set $\mathcal{I}$ of ELP sentences of the form $intends\_that(I, P, T, A)$ or of the form $intends\_to(I, Act, T, A)$, where I is the intention identification, P is a property, T is a time points, A is a list of attributes and Act is an action, and such that (1)$\forall intends\_that(I, P, T, A) \in \mathcal{I}.(Now \leq T)$; (2)$\forall intends\_to(I, Act, T, A) \in \mathcal{I}.(Now \leq T)$; (3)$\forall intends\_to(I, Act, T, A) \in \mathcal{I}.(\mathcal{B} \not\models_P (happens(E, T, T_F), act(E, Act)))$; (4)$\exists \Delta.(P' \cup \Delta \not\models_P \bot)$, where (a)$P'$ is the abductive framework $\langle \mathcal{B}, \{happens/3, act/2\}, IC(\mathcal{I}) \rangle$; (b)$IC(\mathcal{I})$ is set of constraints generated by intentions, defined as "$holds\_at(P, T) \Leftarrow$", for every $intends\_that(I, P, T, A)$ in $\mathcal{I}$ and "$happens(E, T, T_f) \Leftarrow$" and "$act(E, Act) \Leftarrow$" for every $intends\_to(I, Act, T, A)$ in $\mathcal{I}$;*

An agent intends that a property P holds at time T iff $intends\_that(I, P, T, A) \in \mathcal{I}$. A agent intends to do an action Act at time T iff $intends\_to(I, Act, T, A) \in \mathcal{I}$.

The definition of intentions enforces its rationality constraints. Conditions (1) and (2) state that an agent should not intend something at a time that has already past. Condition (3), the non-triviality condition[8] states that an agent should not intend something it believes is already satisfied or that will be satisfied with no efforts by the agent. Condition (4) states that an agent only intends something it believes is possible to be achieved, i.e., if it believes there is a course of actions that leads to the intended state of affairs.

Notice that we take some measures to avoid the *side-effect problem*[7]. Bratman states that *an agent who intends to do α an believes that doing α would require it to do β does not have to also intend β.* In our definition, an agent does not intend the *side-effects* of its intentions, as the side-effects are not in the intentions set[5]. To fully avoid this problem, it is also important, when we define dynamic aspects of intentions, namely how to originate intentions and how and when to revise them, that we characterize commitment in a way that side-effects of adopted intentions do not prevent agents from adopting intentions, neither that side-effects make agents revise their intentions.

4.1 Originating Intentions

Once we have characterized intentions and related mental states, it is necessary to define how these mental states interact to produce the agent's behavior, namely how agents select intentions and when and how agents revise selected intentions. Agents choose their intentions from two different sources: from its desires and as a refinement from other intentions. We start by defining the creation of intentions from desires.

By definition, there are no constraints on the agent's desires. Therefore, an agent may have contradictory desires, i.e., desires that are not jointly achievable. Intentions, on the other hand, are restricted by rationality constraints (as shown in the previous section). Thus, agents must select only those desires that conform to those constraints. We start by defining those desires that are eligible to be chosen and the notion of candidate desires set.

Definition 4.4 (Eligible Desires) *Let $\mathcal{D}$ be the agent's desires. We call* eligible desires *the set* $\mathcal{D}' = \{desires(D,\ P,\ T,\ A)\ /\ [(desires(D,\ P,\ T,\ A) \leftarrow Body) \in \mathcal{D}] \wedge Now \leq T \wedge (\mathcal{B} \models_P Body)\}$

Eligible desires are those desires the agent believes are not satisfied. Recall that, according to rationality constraint in section 4, its is not rational for an agent to intend something it believes is already achieved or that is impossible. Notice that if a desire is conditional, then the agent should believe this condition is true.

As the initial set of desires, eligible desires may also be contradictory. Therefore, it is necessary to determine those subsets of the eligible desires that are jointly achievable. In general, there may be more than one subset of the eligible desires that are jointly achievable. Therefore, we should indicate which of these subsets are preferred to be adopted as intentions. We do this through the preference relation defined bellow.

[5]This is similar to the belief base approach, but it is simpler because we do not have to include the derivations in the base, as with belief bases.

Definition 4.5 (Desires Preference Relation $<_{Pref}$) *Let $\mathcal{D}$ be the agent's desires, $\mathcal{D}'$ the set of eligible desires from $\mathcal{D}$, $\mathcal{P}(\mathcal{D}')$ the power set of $\mathcal{D}'$ and $R, S \in \mathcal{P}(\mathcal{D}')$. We say that $R <_{Pref} S$ (R is less preferred than S) if the biggest importance occurring in S and not occurring in R is bigger than the biggest importance occurring in R and not occurring in S; if there is no such biggest importance in S, than R is less preferred than S if S has more elements than R.*

According to this definition, the agent should prefer to satisfy first the most important desires. Additionally to preferring the most important ones, the agent adopts as much desires as it can.

Example 4.1 *Given the set $D' = \{desires(1,\ a,\ U,\ [0.5]),\ desires(2,\ b,\ U,\ [0.3]),\ desires(3,\ c,\ U,\ [0.3]),\ desires(4,\ d,\ U,\ [0.2])\}$ of eligible desires: (1) for $d_1 = \{desires(2,\ b,\ U,\ [0.3]),\ desires(4,\ c,\ U,\ [0.2])\}$ and $d_2 = \{desires(1,\ a,\ T,\ [0.5])\}$,we have $(d_2 <_{Pref} d_1)$. That is, d_2 is less preferred than d_1, since $T = 0.5$ for d_2 and $T' = 0.3$ for d_1; (2) for $d_3 = \{desires(3, c, U, [0.3])\}$, we have $d_3 <_{Pref} d_1$, as $T = 0.2$ for d_1 and there is no T' for d_3, but $\#d_3 < \#d_1$.*

Notice that the preference relation is a pre-order relation. For instance, if an agent were to choose between $d_4 = \{desires(2, b, U, [0.3])\}$ and $d_5 = \{desires(3, c, U, [0.3])\}$, based only on the importance of desires and maximization of desires satisfied, it would not prefer either of them. And, indeed, according to the preference relation, we have that neither $(d_4 <_{Pref} d_5)$ nor $(d_5 <_{Pref} d_4)$. Based on this preference order, we define the preference graph that will be used to revise the mental states and the revision process.

Definition 4.6 (Desires Preference Graph) *Let $\mathcal{D}$ be the agent's desires and $\mathcal{D}'$ the set of eligible desires from $\mathcal{D}$. Let Revisable be the set $\{unsel(D)\ /\ \exists\ desires(D,\ P,\ T,\ A) \in \mathcal{D}'\}$ and $index : \mathcal{P}(Revisable) \longrightarrow \aleph^+$ a function from the power set of Revisable to natural numbers (zero excluded) that attributes a level number to elements of $\mathcal{P}(Rev)$. The* desires preference graph *is the graph defined by (1)$Rev(\mathbf{bottom}) = \{happens(E, T_i, T_f), act(E, A)\}$; (2)$Rev(i) = R \cup \{happens(E, T_i, T_f),\ act(E, A)\}$, where $R \in \mathcal{P}(Revisable)$ and $i = index(R)$; (3)$i \ll \mathbf{bottom}$, where $i = index(R)$, $R \in \mathcal{P}(Revisable)$ and $\not\exists S \in \mathcal{P}(Revisable)\ .\ (S <_{Pref} R)$; (4)$j \ll k_1, \ldots, k_n$, where $j = index(R)$, $k_i = index(S_i)$ $(1 \leq i \leq n)$, $R, S_i \in \mathcal{P}(Rev)$ $(1 \leq i \leq n)$ and R is an antecedent of S.*

That is, when revising the eligible desires set, the preferred revisions are those that eliminate first the less important desires, and the least possible amount of desires, as shown in the definition bellow.

Definition 4.7 (Candidate Desires Set) *Let $\mathcal{D}$ be the agent's desires and $\mathcal{D}'$ the set of eligible desires from $\mathcal{D}$ with a preference graph associated to it. We call* candidate desires set *any set $\mathcal{D}'_C = \{desires(D, P, T, A)\ /\ (desires(D,\ P,\ T,\ A) \in \mathcal{D}') \wedge [\exists\ \Delta\ .\ (\mathcal{B} \cup \Delta \models_P (holds\_at(P,\ T),\ not\ \ unsel(D))) \wedge (P' \cup \Delta \not\models_P \bot)]\}$ where (1)P' is the abductive framework $\langle\ \mathcal{B},\ \{happens(E,\ T_i,\ T_f),\ act(E,\ Act),\ unsel(D)\}\ , IC\rangle$; (2) IC is a set of constraints of the form*

- $\{holds\_at(P,T) \Leftarrow Body, not\ \ unsel(D)\}$ *for every* $desires(D,P,T,A)$ *in* $\mathcal{D}'$*;*
- *the constraints* $IC(\mathcal{I})$ *generated by intentions (see definition 4.3).*

In the revision process, we mix abductive reasoning with defeasible reasoning, where the literal $unsel(D)$ is defeasible. Its intuitive meaning is *"Desire D should not be selected as an intention"*. If the agent believes it is possible to satisfy all of its desires (if it can abduce actions that satisfy all desires and satisfy all constraints), it will find a revision that contains only $happens/3$ and $act/2$. This is enforced by the preference graph with $Rev(\mathbf{bottom}) = \{happens(E, T_i, T_f), act(E, A)\}$. When constraints may not be all concurrently satisfied, it means that the adoption of all desires as intentions leads to contradictions, i.e., they are not jointly satisfiable.

Notice that contradictions do not arise if the actions necessary to satisfy two different intentions have contradictory effects. Recall that, according to the EC axioms, a property P holds if there is an action that initiates it or, alternatively, if $\neg P$ implicitly does not hold, and vice-versa for $\neg P$. Therefore, actions that make contradictory properties hold in fact just cancel each other. This allows us to avoid the *side-effect problem*[7]. If we allowed for this kind of situations to raise contradictions, we would be making the agent preclude intentions that have contradictory consequences, but that are not directly contradictory with each other, and making the agent intend the logical consequences of its intentions. On the other hand, if an action necessary to satisfy an intention cancels a property that is also an intention, a constraint is violated and that course of action is rejected. In this case, the revision will try to defeat intentions, changing the truth value of $unsel(D)$ literals. According to the preference graph, it will try to defeat those constraints that represent the less important desires, trying to preserve the maximum of the most important ones.

As we mentioned before, the desires preference relation is not an order relation. Therefore, it is possible to have more than one candidate set after a revision. However, if we consider only achievability and desires attributes as decision criteria, it makes no difference for the agent to adopt any of the candidate desires set[6][4].

Definition 4.8 (Primary Intentions) *Let* $\mathcal{D}$ *be the agent's desires,* $\mathcal{D}'_C$ *a candidate desires set from* $\mathcal{D}$*. The* primary intentions *of an agent is the set* $\{intends\_that(D,\ P,\ T,\ A)\ /\ desires(D,\ P,\ T,\ A) \in Des'_C)\}$

4.1.1 Intentions as Refinements from Intentions

Once the agent adopts its intentions, it will start planning to achieve those intentions. During planning, the agent will form intentions that are relative to

[6]The revision process provided by the ELP framework defines a *sceptical revision*[1], that is the revision formed by the union of all the minimal program revision. This kind of approach prevents the agent from having to choose one of the minimal revisions. However, for intentions, this is not adequate, since we would like our agents to try to satisfy all the eligible desires it can.

pre-existing intentions. That is, they "refine" their existing intentions. This can be done in various ways, for instance, a plan that includes an action that is not directly executable can be elaborated by specifying particular way of carrying out that action; a plan that includes a set of actions can be elaborated by imposing a temporal order on that set[12]. Since the agent commits to the adopted intentions, these previously adopted intentions constrain the adoption of new ones. That is, during the elaboration of plans, a potential new intention is only adopted if it is not contradictory with the existing intentions and with beliefs.

Definition 4.9 (Relative Intentions) *Let $\mathcal{I}_P$ be the set of primary intentions. A* planning process *is a procedure that, for each $i \in \mathcal{I}_P$, will generate a set of temporal ordered actions $\mathcal{I}_R$ that achieve i, such $\mathcal{B} \cup \mathcal{I}_P \cup \mathcal{I}_R$ is non-contradictory. The set $\mathcal{I}_R$ are the* relative intentions *of the agent.*

The non-contradiction condition enforces again the notion of commitment, i.e., once an intention is adopted it constrains the adoption of new intentions.

4.2 Revising Intentions

In the previous section, we defined how the agent chooses its intentions. As we have seen, weighing motivations and beliefs means finding inconsistencies in competing desires, checking valid desires according to beliefs and intentions, resolving constraints imposed by intentions and desires, i.e., very expensive reasoning activities. It is now necessary to define *when* the agent should perform this process.

We argue that it is not enough to state that an agent should revise its intentions when it believes a certain condition holds, like to believe that an intention has been satisfied or that it is no longer possible to satisfy it, as this suggests that the agent needs to verify its beliefs constantly. Instead, we take the stance that it is necessary to define, along with those conditions, a mechanism that triggers the reasoning process without imposing a significant additional burden on the agent. Our approach is to define those conditions that make the agent start reasoning about intentions as constraints over its beliefs. Recall that we assume that an agent constantly has to maintain its beliefs consistent, whenever new facts are incorporated.

Definition 4.10 (Trigger from Intentions) *Let $\mathcal{B}$ be the agent's beliefs set and $\mathcal{I}$ its intentions. We add to $\mathcal{B}$ the following* trigger constraints*: (1) $(\bot \Leftarrow Now > T,\ not\ \ rev\_int)$, for each $(intends\_that(\ I,\ P,\ T,\ A),\ intends\_to(\ I,\ Act,\ T,\ A)) \in \mathcal{I}$; (2) $(\bot \Leftarrow happens(E,\ T_i,\ T_f),\ act(E,\ Act),\ not\ rev\_int$, for each $intends\_to(\ I,\ Act,\ T,\ A)) \in \mathcal{I}$.*

The literal *rev_int* is part of the revisable set of beliefs, and its initial value is $false$. Whenever the agent revises its beliefs and one of the conditions for revising beliefs hold, a contradiction is raised. We identify such contradiction by testing if *rev_int* is in the selected revision for the beliefs set, i.e., if it has to have its truth value modified in order to restore consistency. The intention revision process is triggered when one of these constraints is violated.

The conditions we have defined so far are the usual ones defined by formal models of agents. As we have seen before, this characterization of intentions may lead to some fanatical behavior. Therefore, we need to adopt additional constraints that will avoid those unwanted behaviors. We take the stance that the same reasons that originated intentions may be used to break commitment associated to them[6]. If we accept that an intention is originated from desires, it is reasonable to state that it is not rational to persist with an intention whose reasons are superseded by more urgent or important ones. The agent's normal behavior would be to weigh its competing desires and beliefs, selecting its intentions. The agent would commit to these intentions and they would constitute the filter of admissibility for other intentions. Also, the agent would try to satisfy those intentions, until successful accomplishment, impossibility (as usually defined), or *until some of his other desires that were not selected before would become eligible*, or *until the desires that originated them would not be eligible anymore*, re-activating the revision process that would weigh (again) competing desires and beliefs. Since the notion of commitment is preserved, the agent would not start this process every time there is a change in beliefs, but only if relevant conditions trigger the intention revision process, changing the agent's focus of attention[5]. These triggers are determined by the desires pre-conditions. We model this triggers using an approach similar to the normative constraints in [18, 23].

Definition 4.11 (Trigger Constraints from Desires) *Let $\mathcal{D}$ be the agent's desires and $\mathcal{D}'$ the set of eligible desires from $\mathcal{D}$. We define* trigger constraints from desires *as*

1. *For every $desires(D,P,T,A) \leftarrow Body \in (\mathcal{D} - \mathcal{D}')$ with importance A bigger that the biggest importance in intentions, we define a trigger constraint $\perp \Leftarrow Body$, not rev_int;*

2. *given the set of actions Δ abduced by the agent (see definition 4.7), for each $desires(D,P,T,A) \in (\mathcal{D}' - \mathcal{D}'_C)$ with importance A bigger that the biggest importance in intentions, we define a trigger constraint $\perp \Leftarrow C_1, \ldots, C_n$, not rev_int, where $Cond_i$ ($1 \leq i \leq n$) are the conditions the agent could not bring about when selecting the candidate desires set.*

The first constraint trigger is formed by the pre-conditions of those desires that were not eligible and that are more important than those that were evaluated. It means that if the pre-conditions of such desires become true, these desires (that were not considered during reasoning) become eligible. Therefore, it is necessary to re-evaluate desires and beliefs to check if this new desire may be brought about. The second constraint is formed by the pre-conditions of those eligible desires that, although more important, were not relevant when the agent made his choice. Notice that there are no triggers for those desires that were eligible but that were ruled out during the choice of a revision. This is so because they had already been evaluated and they have been considered less important than the other desires. Therefore, it is of no use to trigger the whole process again (i.e., to shift the agent's attention) to re-evaluate them.

5 Conclusion and Further Work

The main contribution of this paper is to provide a formal model of agent that reduces the gap between agent specification and agent implementation. Adopting ELP as the underlying formalism has both preserved the main characteristics of formal models, namely the ability to formally define and verify agents, and has provided machinery the agent may use to reason. Also, as a consequence of this shift in the perspective when defining the formal model, we have been able to account for both static and dynamic aspects of intentions. By static aspects, we mean its relation with desires and beliefs and the properties that this relation has (namely, consistency and avoidance of side-effects). By dynamic aspects, we mean establishing criteria, other than those derived from commitment, to drop intentions (namely, based on the cause-effect relation between desires and intentions). We also define where intentions come from and how the agent chooses its intentions.

We may compare our work to other models of intentions in the literature. The classical model is due to Cohen and Levesque[8]. In this model, the authors do not address the question of where and how the agent chooses its intentions, neither they overcome the fanatical behavior caused by the commitment strategy. The same may be said about other models that followed Cohen and Levesque's and that tried to overcome some of its theoretical problems: Rao and Georgeff[17] tried to solve the side-effect problem present in [8] introducing the notions of strong and weak realism; Konolidge and Pollack[12] also tried to solve the side-effect problem using non-normal modal logics, where the solution were embedded in the semantics; Singh[21] introduces the notion of know-how and defines communication and multi-agent interaction using its model, but he does not treat the question of the origins of intentions nor how to relax commitment. Van Linder et ali[22] model the origin and choice of intentions and provide a formalism that can be extended to model other pro-active attitudes, but they do not provide any improvements on the notion of commitment. All these works have also in common the fact that their models are based on modal logics and, therefore, may be just used to specify agents. We are, on the other hand, heading to define a computational theory of agency that has an underlying architecture, the one partially described here. This computational theory will allow us to specify and prove properties about the agents, as well as to execute such an specification to test if the agents behave like expected. Our choice of the logical formalism has been deeply influenced by this long term goal, since it is formal and computationally tractable. This concern with the computational aspects of the agent models can also be found in [16].

Our work has many similarities with the one of Gaspar et ali[10]. Although they use a different formalism (that is not so computationally tractable as ours), both works have many objectives in common: to address the question of the origin of intentions, to have a computational model, to revise intentions. While we concentrate on the relation between desires and intentions and use this relation to relax commitment, they focus on the evolution of plans and communication, and on how this affects the stability of intentions. In order to have a complete treatment of these issues, both aspects are important and must be integrated.

References

[1] J.J. Alferes and L.M. Pereira. *Reasoning with Logic Programming.* Springer-Verlag, Berlin, DE., 1996. Lecture Notes in Artificial Intelligence Series (LNAI 1111).

[2] J.J. Alferes, L.M. Pereira, and T. Przymusinski. Belief revision in non-monotonic reasoning and logic programming. In C. Pinto-Ferreira and N.J. Mamede, editors, *Proceedings of the Seventh Portuguese Conference on Artificial Intelligence (EPIA'93)*, Berlin, Germany, 1995. APIA, Springer-Verlag. Lecture Notes on Artificial Intelligence (LNAI 990).

[3] L. Beaudoin. *Goal Processing in Autonomous Agents.* PhD thesis, Birmingham University, Birmingham, UK, August 1994.

[4] J. Bell and Z. Huang. Dynamic goal hierarchies. In *Proceedings of the Second Workshop on Practical Reasoning and Rationality*, London, England, 1997. AISB Workshop Series.

[5] L.M. Botelho and H. Coelho. Agents that rationalize their decisions. In *Proceedings of the II International Conference on Multi-Agent Systems*, Kyoto, Japan, 1996. AAAI Org.

[6] M. Bratman. Planning and the stability of intentions. *Minds and Machines*, 2:1–16, 1992.

[7] M.E. Bratman. What is intention? In P.R. Cohen, J.L. Morgan, and M. Pollack, editors, *Intentions in Communication*, chapter 1. The MIT Press, Cambridge, MA, 1990.

[8] P.R. Cohen and H.J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42:213–261, 1990.

[9] C. Damásio, W. Nejdl, and L.M. Pereira. Revise: An extended logic programming system for revising knowledge bases. In *Knowledge Representation and Reasoning.* Morgan Kaufmann inc., 1994.

[10] G. Gaspar and H. Coelho. Where do intentions come from?:a framework for goals and intentions adoption, derivation and evolution. In C. Pinto-Ferreira and N.J. Mamede, editors, *Proceedings of the Seventh Portuguese Conference on Artificial Intelligence (EPIA'93)*, Berlin, Germany, 1995. APIA, Springer-Verlag. Lecture Notes on Artificial Intelligence (LNAI 990).

[11] A. Haddadi. *Communication and Cooperation in Agent Systems: a Pragmatic Theory.* Springer-Verlag, Berlin, DE., 1996. Lecture Notes in Artificial Intelligence Series (LNAI 1056).

[12] K. Konolige and M. Pollack. A representationalist theory of intentions. In *Proceedings of the XII International Joint Conference on Artificial Intelligence (IJCAI'93)*, Chambéry, France, 1993. IJCAI inc.

[13] R. Li and L.M. Pereira. Knowledge assimilation in domains of actions: A possible causes approach. *Journal of Applied Non-Classical Logic*, 1996. Special issue on Inconsistency Handling in Knowledge Systems.

[14] M.C. Móra, J.G. Lopes, and H. Coelho. Modelling intentions with extended logic programming. In J. Wainer and A. Carvalho, editors, *Advances in artificial intelligence: proceedings of the 12th Brasilian Symposium on Artificial Intelligence*, Berlin, Germany, 1995. SBC, Springer-Verlag. Lecture Notes in Artificial Intelligence (LNAI 991).

[15] J. Quaresma and J.G. Lopes. A logic programming framework for the abduction of events in a dialogue system. In *Proceedings of the Workshop on Automated Reasoning*, London, England, 1997. AISB Workshop Series.

[16] A.. Rao. Agentspeak(1): BDI agents speak out in a logical computable language. In *Proceedings of the European Workshop on Modelling Autonomous Agents and Multi-Agents Worlds 1996 (MAAMAW'96)*, Berlin, Germany, 1996. Springer-Verlag.

[17] A.S. Rao and M.P. Georgeff. Modelling rational agents within a BDI-architecture. In R. Fikes and E. Sandewall, editors, *Proceedings of the Knowledge Representation and Reasoning'91 (KR&R'91)*, San Mateo, CA., 1991. Morgan Kauffman Publishers.

[18] M. Schroeder, Iara de Almeida Móra, and Luís Moniz Pereira. A deliberative and reactive diagnosis agent based on logic programming. In J. P. Müller, M. J. Wooldridge, and N. R. Jennings, editors, *Intelligent Agents III — Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages (ATAL-96)*, volume 1193 of *Lecture Notes in Artificial Intelligence*, pages 293–308. Springer-Verlag, Heidelberg, 1997.

[19] J. Searle. What is an intentional state? In H. Dreyfuss and H. Hall, editors, *Husserl, Intentionality and Cognitive Science*, volume 42, pages 213–261. 1984.

[20] Y. Shoham. Agent-oriented programming. *Artificial Intelligence*, 60:51–92, 1993.

[21] M. Singh. *Multiagent systems: a theoretical framework for intentions, know-how, and communications*. Springer-Verlag, Heidelberg, Germany, 1994. Lecture Notes in Artificial Intelligence (LNAI 799).

[22] B. van Linder, W. van der Hoek, and J. J. Ch. Meyer. Formalizing motivational attitudes of agents: On preferences, goals, and commitments. In M. Wooldridge, J. P. Müller, and M. Tambe, editors, *Intelligent Agents II — Proceedings of the Second International Workshop on Agent Theories, Architectures, and Languages (ATAL-95)*, volume 1037 of *Lecture Notes in Artificial Intelligence*, pages 17–32. Springer-Verlag, Heidelberg, 1996.

[23] G. Wagner. A logical and operational model of scalable knowledge and perception-based agents. In *Proceedings of the European Workshop on Modelling Autonomous Agents and Multi-Agents Worlds 1996 (MAAMAW'96)*, Berlin, Germany, 1996. Springer-Verlag.

7 ATAL98 –“BDI Models and Systems: Reducing the Gap”

Este artigo, apresentado no ATAL98[MÓR 98a], mostra o modelo formal e executável de agente desenvolvido. Mostra, ainda, como a estratégia adotada, de escolha de um formalismo apropriado e mudança de perspectiva na construção do modelo, reduz a distância entre a teoria formal de agentes e sua implementação. Nomeadamente, ao se eleminar definições recursivas e definir os axiomas de comportamento do agente, considerou-se não somente a correção das definições como carcaterísticas necessárias para que estes axiomas pudessem ser executados: tratabilidade, parametrização dos axiomas aos estilo de arquiteturas, e correção do ponto de vista operacional.

BDI Models and Systems: Reducing the Gap

Michael C. Móra*‡, José G. Lopes†*, Rosa M. Viccari‡** and Helder Coelho *

* CPGCC/II, Universidade Federal do Rio Grande do Sul
Bloco IV, Campus do Vale, Av. Bento Gonçalves 9500, Porto Alegre, RS, Brasil
{michael,rosa}@inf.ufrgs.br

† CENTRIA/DI – Universidade Nova de Lisboa
Quinta da Torre – 2825 Monte da Caparica – Portugal
gpl@di.fct.unl.pt

‡ II – Pontifícia Universidade Católica do Rio Grande do Sul
Prédio 30 – Av. Ipiranga 6618, Porto Alegre, RS, Brasil
michael@inf.pucrs.br

* DI/FCUL – Universidade de Lisboa
Bloco C5, Piso 1, Campo Grande, 1700 Lisboa, Portugal
hcoelho@di.fc.ul.pt

Abstract. **B**eliefs-**D**esires-**I**ntentions models (or BDI models) of agents have been around for quit a long time. The purpose of these models is to characterize agents using anthropomorphic notions, such as mental states and actions. However, despite the fact that many systems have been developed based on these models, it is a general concern that there is a gap between those powerful BDI logics and practical systems. The purpose of this paper is to present a BDI model that, besides being a formal model of agents, is also suitable to be used to implement agents. Instead of defining a new BDI logic or choosing an existing one, and extending it with an operational model, we define the notions of belief, desires and intentions using a logic formalism that is both well-defined and computational.

Keywords: BDI models; mental states modeling; agent architectures; logic programming.

1 Introduction

Beliefs-**D**esires-**I**ntentions models (or BDI models) of agents have been around for quite a long time. The purpose of these models is to characterize agents using anthropomorphic notions, such as mental states and actions. Usually, these notions and their properties are formally defined using logical frameworks that allow theorists to analyze, to specify and to verify rational agents,like in [10] [26] [29] [31] [25] [17](among others).

* Supported by project DIXIT/JNICT.
** Supported by CNPq/Brazil.

However, despite the fact that many systems have been developed based on these models ([8,16,11,17], to mention some), it is a general concern that there is a gap between those powerful BDI logics and practical systems. We believe that the main reason for the existence of this gap is that the logical formalisms used to define the models do not have an operational model that support them. By an operational model we mean proof procedures that are correct and complete with respect to the logical semantics, as well as mechanisms to perform different types of reasoning needed to model agents [20,21].

There are, at least, two major approaches that may be used to overcome this limitation of BDI models. One is to extend existing BDI logics with appropriate operational models so that agent theories become computational. This is the approach taken by Rao, in [25], where he defines a proof procedure for the propositional version of his BDI logic[26]. Also, Bell in [4] develops a computational theory that can be applied to his formalizations of goal hierarchies[5]. Still, Schild's[27] representation of Rao and Georgeff's[26] BDI theory with standard logic of concurrency follows this approach.

The other approach is to define BDI models using a suitable logical formalism that is both powerful enough to represent mental states and that has operational procedures that allow us to use the logic as a knowledge representation formalism, when building the agent. This is the path followed by Corrêa and Coelho[11], where they define their SEM architecture using situation theory[15] as their underlying formalism and use Nakashima's operational model[23,13]. This is also the approach we follow in this paper.

The purpose of this paper is to present a BDI model that, besides being a formal model of agents, is also suitable to be used to implement agents. Instead of defining a new BDI logic or choosing an existing one, and extending it with an operational model, we define the notions of belief, desires and intentions using a logical formalism that is both well-defined and computational. The formalism we are using is *logic programming extended with explicit negation* (ELP) with the *Well-Founded Semantics eXtended for explicit negation* (WFSX). ELP with WFSX (simply ELP, from now on) extends normal logic programs with a second negation named *explicit*[1] *negation*. This extension allows us to explicitly represent negative information (like a belief that a property P does not hold, or an intention that a property Q should not hold) and increases the expressive power of the language. When we introduce negative information, we may have to deal with contradictory programs[1]. The ELP framework, besides providing the computational proof procedure for theories expressed in its language, also provides a mechanism to determine how to minimally change a logic program in order to remove contradictions. Our model benefits from these features provided by the logical formalism.

As it is usually done[10,30,17], we focus on the formal definition of mental states and on how the agent behaves, given such mental states. But, contrasting with these former approaches, our model is not only an agent specification, but it may also be executed in order to verify the actual agent behavior, as well as it may be used as reasoning mechanism by actual agents. We depart from Bratman's analysis[9], where he states

[1] In contrast with the usual *negation as failure* or *negation by default* of normal logic programs, which is called *implicit negation* in the ELP context.

that, along with desires and beliefs, intentions is a fundamental mental state. Therefore, initially we define these three mental states and the static relations between them, namely constraints on consistency among those mental states. Afterwards, we advance with the definition of dynamic aspects of mental states, namely how the agent chooses its intentions, and when and how it revises its intentions[2].

The paper is organized as follows: in section 2, we present the ELP formalism and the reasoning mechanisms it provides; in section 3, we present an action and time theory based on a variation of the event calculus[24] that is used to define the mental states; in section 4, we present our agent model; in section 5, we discuss some related and future work and draw some conclusions.

2 Extended Logic Programming

Unlike normal logic programs, where negative information is only stated implicitly (i.e., a proposition is false only if it cannot be proved to be true), extended logic programs have a second kind of negation that allows us to explicitly represent negative information. If, on one hand, this second negation increases the representational power of the language, it may also introduce contradictory information in programs. Therefore, it is necessary to assign some meaning to contradictory programs and to be able to deal with contradiction. The paraconsistent version of the $WFSX$ semantics, $WFSX_P$, assigns meaning to contradictory programs and is used as a tool by the revision mechanism that restores consistency to contradictory programs. This revision mechanism may be used to perform several kinds of reasoning, as we will see bellow.

An extended logic program (ELP) is a set of rules $H \leftarrow B_1, \ldots, B_n, not\ \ C_1, \ldots, not\ \ C_m$ (m,n $\geq$ 0), where $H, B_1, \ldots, B_n, C_1, \ldots, C_m$ are objective literals. An objective literal is either an atom A or its explicit negation $\neg A$. The symbol not stands for negation by default and $not\ L$ is a default literal. Literals are either objective or default literals and $\neg\neg L \equiv L$. The language also allows for integrity constraints of the form $A_1 \vee \ldots \vee A_l \Leftarrow B_1, \ldots, B_n, not\ \ C_1, \ldots, not\ \ C_m$ (m,n $\geq$ 0; l $\geq$ 1) where $A_1, \ldots, A_l, B_1, \ldots, B_n, C_1, \ldots, C_m$ are objective literals, stating that at least one of the A_i (i $\geq$ 1), should hold if its body $B_1, \ldots, B_n, not\ C_1, \ldots, not\ \ C_m$ (m,n $\geq$ 0) holds. Particularly, when $A = \perp$, where $\perp$ stands for *contradiction*, it means that a contradiction is raised when the constraint body holds. The set of all objective literals of a program P is the extended Herbrand base of P denoted by $\mathcal{H}(P)$.

In order to assign meaning to ELP programs, we need the notions of interpretation of ELP clause[3]. Roughly, an interpretation of an ELP program is a set of literals that verifies the coherence principle: if $\neg L \in T$ then $L \in F$. A literal L is true in an

[2] As we stated before, our concern is the definition of agents' behavior in terms of their mental states. Therefore, its beyond the scope of this work to treat dynamic aspects of beliefs, as belief revision. Notice, however, that opposed to models like the ones defined in [10,30,17] and others, the logical grounds on which we base our definitions are suitable to define belief revision strategies and to be used by agents as a reasoning tool for truth maintenance. See [2,19] for two different approaches on this matter.

[3] For a formal definition, see [1].

interpretation I if $L \in I$, L is false in I if $not\ L \in I$, and L is undefined, otherwise. An interpretation I is a model for program P iff, for every clause L in P, $I \models_P L$.

2.1 Program Revision and Non-Monotonic Reasoning

As we stated before, due to the use of explicit negation, programs may be contradictory. In order to restore consistency in programs that are contradictory with respect to $WFSX$, the program is submitted to a revision process that relies on the allowance to change the truth value of some set of literals. This set of literals is the set of *revisable literals*, and can be any subset of $\mathcal{H}$ for which there are no rules or there are only facts in the program. No other restriction is made on which literals should be considered revisable. They are supposed to be provided by the user, along with the program.

The revision process changes the truth value of revisable literals in a minimal way and in all alternative ways of removing contradiction. *Minimally revised programs* can be defined as those programs obtained from the original one after modifying the subsets of revisable literals.

The ability to represent negative information, along with a well-defined procedure that restores consistency in logic program, makes ELP suitable to be used to perform different forms of non-monotonic reasoning [1]. In particular, we are interested in two of these forms of non-monotonic reasoning, namely *defeasible reasoning* and *abductive reasoning*.

Defeasible Reasoning A defeasible rule is a rule of the form *Normally if A then B*. ELP allows us to express defeasible reasoning and to give meaning to a set of rules (defeasible or not) when contradiction arises from the application of the defeasible rules. To remove contradiction, we revise the truth value of revisable literals in the body of the defeasible rules.

Abductive Reasoning Abductive reasoning consists of, given a theory T and a set of observations O, to find a theory Δ such that $T \cup \Delta \models O$ and $T \cup \Delta$ is consistent. In the ELP context, an abductive framework P' is a tuple $\langle P, Abd, IC \rangle$, where P is an extended logic program, Abd is the set of abducible literals and IC is the set of integrity constraints. An observation O has an abductive explanation Δ iff $P' \cup \Delta \models_P O$ and $P \not\models_P \Delta$. We may abduce a theory Δ that explains such observations making each of these observations O a new integrity constraint $O \Leftarrow$ and revising the program with revisable set Abd (recall that a program is contradictory iff for some literal L in program P, $P \models L$ and $P \models \neg L$, or a constraint is not satisfied).

Preferred Revisions Sometimes, it may be necessary to state that we do not only want the minimal revisions provided by the formalism, but also that we prefer revisions that have a certain fact only after finding that no other revisions including other facts exist. The ELP formalism allows us to express preferences over the order of revisions using a labeled directed acyclic and/or graph defined by rules of the form[14]:

$$Level_0 \ll Level_1 \wedge Level_2 \wedge \ldots Level_n (\text{n} \geq 1)$$

$Level_i$ nodes in the graph are preference level identifiers. To each of these levels is associated a set of revisables denoted by $\mathcal{R}(Level_i)$. Rules like the one above for $Level_0$ state that we want to consider revisions for $Level_0$ only if, for some rule body, its levels have been considered and there are no revisions at any of those levels. The root of the preference graph is the node denoted by $\mathbf{bottom}$.

3 The Event Calculus

When we reason about pro-attitudes like desires and intentions, we need to deal with properties that should hold at an instant of time and with actions that should be executed at a certain time. Therefore, in order to represent them and to reason about them, we need to have a logical formalism that deals with actions and time. In this work, we use a modified version of the Event Calculus (EC) proposed in [24]. This version of the EC allows events to have a duration and an identification, instead of being instantaneous and identified to the instant of time the event occurs. As a consequence, events may occur simultaneously. The predicate $holds\_at$ defining the properties that are true at a specific time is:

$$\begin{aligned}
holds\_at(P,T) &\leftarrow happens(E,T_i,T_f),\\
&\qquad initiates(E,T_P,P), T_P < T,\\
&\qquad T_P >= T_i, persists(T_P,P,T).\\
persists(T_P,P,T) &\leftarrow not\ clipped(T_P,P,T).\\
clipped(T_P,P,T) &\leftarrow happens(C,T_{ci},T_{cf}),\\
&\qquad terminates(C,T_C,P),\\
&\qquad T_C >= T_{ci}, not\ out(T_C,T_P,T).\\
out(T_C,T_P,T) &\leftarrow (T \leq T_C); (T_C < T_P).
\end{aligned}$$

The predicate $happens(E,T_i,T_f)$ means that event E occurred between T_i and T_f; $initiates(E,T,P)$ means that event E initiates P at time T; $terminates\ (E,\ T,\ P)$ means that event E terminates P at time T; $persists(T_P,P,T)$ means that P persists since T_P until T (at least). We assume there is a special time variable Now that represents the present time. Note that a property P is true at a time T ($holds\_at(P,T)$), if there is a previous event that initiates P and if P persists until T. P persists until T if it can not be proved by default the existence of another event that terminates P before the time T. We need additional rules for the relation between not holding a property and holding its negation and we also need to define the relation between the two kinds of negation:

$$\begin{aligned}
holds\_at(\neg P,T) &\leftarrow \neg holds\_at(P,T).\\
\neg holds\_at(P,T) &\leftarrow not\ holds\_at(P,T).
\end{aligned}$$

The predicates that will be abduced need to be related by some integrity rules, namely:

$$\perp \Leftarrow happens(E,T_{1i},T_{1f}), happens(E,T_{2i},T_{2f}),$$

$$not(T_{1i} = T_{2i}, T_{1f} = T_{2f}).$$
$$\perp \Leftarrow happens(E, T_i, T_f),\ not(T_f < T_i).$$
$$\perp \Leftarrow happens(E, T_i, T_f), not(act(E, A)).$$

that state, respectively, that events cannot be associated to different time intervals, that events cannot have a negative duration and that events must have an associated action.

The EC allows us to reason about the future, by hypothetically assuming a sequence of actions represented by $happens/3$ and $act/2$ predicates and verifying which properties would hold. It also allows us to reason about the past. In order to know if a given property P holds at time T, the EC checks what properties remain valid after the execution of the actions that happened before T. We are now ready to define the agent's model.

4 The Agent Model

Now that we have the logical formalism set, we start defining the BDI model. We depart from Bratman's analysis about intentions, its role in rational reasoning and how it relates to beliefs and desires. According to Bratman[9], since agents are assumed to be resource-bounded, they cannot continuously evaluate their competing beliefs and desires in order to act rationally. After some reasoning, agents have to commit to some set of choices. It is this choice followed by a commitment that characterizes the intentions. Our model does not define a complete agent, but only the cognitive structure that is part of the agent model.

Definition 1 (Agent Cognitive Structure). An *agent cognitive structure* is a tuple $\mathcal{A}g = \langle \mathcal{B}, \mathcal{D}, \mathcal{I}, \mathcal{T}Ax \rangle$ where

– $\mathcal{B}$ is the set of agent's beliefs;
– $\mathcal{D}$ is the set of agent's desires;
– $\mathcal{I}$ is the set of agent's intentions;
– $\mathcal{T}Ax$ is the set of time axioms, as defined in section 3.

This cognitive structure contains both the mental states that compose the agent and the rules that govern the interaction of these mental states (and, consequently, the agent's behavior).

We should now define every component of this structure. We start by defining desires. Desires are related to the state of affairs the agent eventually wants to bring about. But desires, in the sense usually presented, does not necessarily drive the agent to act. That is, the fact of an agent having a desire does not mean it will act to satisfy it. It means, instead, that before such an agent decides what to do, it will be engaged in a reasoning process, confronting its desires (the state of affairs it wants to bring about) with its beliefs (the current circumstances and constraints the world imposes). The agent will choose those desires that are possible according to some criteria. In other words, desires constitute the set of states among which the agent chooses what to do. Notice that, since agents are not committed to their desires, they need not to be consistent, neither with other desires nor with other mental states.

Definition 2 (Desires Set). The *desires* of an agent is a set of sentences of the form

$$\mathcal{D} = \{holds\_at(des(D, Ag, P, Atr), P) \leftarrow Body\}$$

where D is the desire identification, P is a property, T is a time point, Ag is an agent identification and A is list of attributes. $Body$ is any conjunction of literals. An agent desires that a property P holds at time T iff

$$(holds\_at(des(D, Ag, P, Atr), T) \leftarrow Body) \in \mathcal{D},$$

for some D, Ag, Atr.

Our definition of desires allows the agent to have a desire that a certain property holds (or does not hold) in a specific instant of time (when T is instantiated). Desire clauses may be facts, representing states the agent may want to achieve whenever possible, or rules, representing states to be achieved when a certain condition holds. The attributes associated to each desire define properties, like urgency, importance or priority[3,12,22] that are used by the agent to choose the most appropriate desire (see bellow).

Beliefs constitute the agent's information attitude. They represent the information agents have about the environment and about themselves.

Definition 3 (Beliefs Set). The *beliefs* of an agent is a consistent extended logic program $\mathcal{B}$, i.e. $\mathcal{B} \not\models_P \bot$. An agent believes agent A believes a property P holds at time T iff $\{\mathcal{B} \cup \mathcal{T}Ax\} \models_P holds\_at(bel(A, P), T)$.

We assume that the agent continuously updates its beliefs to reflect changes it detects in the environment. Describing the belief update process is beyond the scope of this paper. We just assume that, whenever a new belief is added to the beliefs set, consistency is maintained.

As we stated before, intentions are characterized by a *choice* of a state of affairs to achieve, and a *commitment* to this choice. Thus, intentions are viewed as a compromise the agent assumes with a specific possible future. This means that, differently from desires, an intention may not be contradictory with other intentions, as it would not be rational for an agent to act in order to achieve incompatible states. Also, intentions should be supported by the agent's beliefs. That is, it would not be rational for an agent to intend something it does not believe is possible. Once an intention is adopted, the agent will pursue that intention, planning actions to accomplish it, re-planning when a failure occurs, and so on. These actions, as means that are used to achieve intentions, must also be adopted as intentions by agents.

Definition 4 (Intentions Set). The *intentions* of an agent at a time T is the set

$$\mathcal{I} = \{X/X = int\_that(I, Ag, P, A) \vee X = int\_to(I, Ag, Act, A)\}$$

where I is the intention identification, P is a property, Ag is an agent, A is a list of attributes, Act is an action, and such that

1. $holds\_at(int\_that(I, Ag, P, A), T)$ or $holds\_at(int\_to(I, Ag, Act, A), T)$;
2. $\forall int\_that(I, Ag, P, A) \in \mathcal{I}.(Now \leq T)$;

3. $\forall int\_to(I, Ag, Act, A) \in \mathcal{I}.(Now \leq T)$;
4. $\forall int\_to(I, Ag, Act, A) \in \mathcal{I}.(\{\mathcal{B} \cup \mathcal{T}Ax\} \not\models_P (happens(E, T, T_F), act(E, Act)))$;
5. $\exists \Delta.(P' \cup \Delta \not\models_P \bot)$, where
   (a) P' is the abductive framework $\langle\{\mathcal{B} \cup \mathcal{T}Ax\}, \{happens/3, act/2\}, IC(\mathcal{I})\rangle$;
   (b) $IC(\mathcal{I})$ is set of constraints generated by intentions, defined as

$$holds\_at(bel(Ag, P), T) \Leftarrow$$

for every $int\_that(I, Ag, P, A)$ in $\mathcal{I}$ and

$$happens(E, T, T_f) \Leftarrow$$
$$act(E, Act) \Leftarrow$$

for every $intends\_to(I, Act, T, A)$ in $\mathcal{I}$;

An agent intends that a property P holds at time T iff $\mathcal{B} \cup \mathcal{T}Ax \cup \mathcal{R} \models_P holds\_at(int\_that\ (I, Ag, P, A), T)$, and it intends to execute an action Act at time T iff $\mathcal{B} \cup \mathcal{T}Ax \cup \mathcal{R} \models_P holds\_at(int\_to\ (I, Act, P, A), T)$.

The definition of intentions enforces its rationality constraints. Conditions 2 and 3 state that an agent should not intend something at a time that has already past. Condition 4, the non-triviality condition[10] states that an agent should not intend something it believes is already satisfied or that will be satisfied with no efforts by the agent. Condition 5 states that an agent only intends something it believes is possible to be achieved, i.e., if it believes there is a course of actions that leads to the intended state of affairs. When designing an agent, we specify only the agent's beliefs and desires. It is up to the agent to choose its intentions appropriatly from its desires. Those rationality constraints must also be guaranteed during this selection process. Notice also that we are able to avoid the *side-effect problem*[9]. Bratman states that *an agent who intends to do α an believes that doing α would imply β does not have to also intend β*. In our definition, an agent does not intend the *side-effects* of its intentions, as the side-effects are not in the intentions set[4]. To fully avoid this problem, it is also important, when we define how the agent selects its intentions and how and and how and when to revise them, that we characterize commitment in a way that side-effects of adopted intentions do not prevent agents from adopting intentions, neither that side-effects make agents revise their intentions.

Example 5 (Warehouse Robot). This is a toy example to illustrate the use of the formalism. Suppose there is a robot rbt that should take all the objects placed in an input counter and store those objects in a warehouse. It also must recharge its batteries when-

[4] This is similar to the belief base approach, but it is simpler because we do not have to include the derivations in the base, as with belief bases.

ever they run out of charge.

Desires

$$
\begin{aligned}
\perp \Leftarrow\ & holds\_at(des({}_{,}rbt, bel(rbt, bat\_chged),{}_{)}\ , T),\\
& holds\_at(des({}_{,}rbt, bel(rbt, stored(O)),{}_{)}\ , T).\\
& holds\_at(des(1, rbt, bel(rbt, bat\_chged), [0.5]), T).\\
& holds\_at(des(2, rbt, bel(rbt, stored(O)), [0.3]), T) \leftarrow holds\_at(bel(rbt,\\
& \qquad input(O)), T).
\end{aligned}
$$

Beliefs

$$
\begin{aligned}
initiates(E, T_f, bel(rbt, bat\_chged)) &\leftarrow happens(E, T_i, T_f),\\
& \quad act(E, charge).\\
terminates(E, T, bel(rbt, bat\_chged)) &\leftarrow happens(E, T, T),\\
& \quad act(E, sense\_low\_bat).\\
initiates(E, T_f, bel(rbt, stored(O))) &\leftarrow happens(E, T_i, T_f),\\
& \quad act(E, store(O)).\\
initiates(E, T_i, bel(rbt, input(O))) &\leftarrow happens(E, T_i, T_i),\\
& \quad act(E, put\_input(O)).\\
terminates(E, T_i, bel(rbt, input(O))) &\leftarrow happens(R, T_i, T_f),\\
& \quad act(E, store(O)).\\
happens(e1, t1, t2).\ act(e1, input(o1)).
\end{aligned}
$$

The agent definition describes only its desires and beliefs. Intentions will be adopted as the agent tries to satisfy its desires.

4.1 Originating Intentions

Once we have characterized intentions and related mental states, it is necessary to define how these mental states interact to produce the agent's behavior, namely how agents select intentions and when and how agents revise selected intentions. Agents choose their intentions from two different sources: from its desires and as a refinement from other intentions. We start by defining the creation of intentions from desires.

By definition, there are no constraints on the agent's desires. Therefore, an agent may have conflicting desires, i.e., desires that are not jointly achievable. Intentions, on the other hand, are restricted by rationality constraints (as shown before). Thus, agents must select only those desires that respect those constraints. We start by defining those desires that are eligible to be chosen and the notion of candidate desires set.

Definition 6 (Eligible Desires). Let $\mathcal{D}$ be the agent's desires. We call *eligible desires* at a time T the set

$$
\begin{aligned}
\mathcal{D}' = \{des(D, Ag, P, A)\ /\ & [holds\_at(des(D, Ag, P, A), T) \leftarrow Body) \in \mathcal{D}] \wedge\\
& Now \leq T \wedge (\mathcal{B} \models_P Body) \wedge\\
& \{\mathcal{B} \cup \mathcal{T}Ax\} \models_P \neg holds\_at(bel(Ag, P), T)\}
\end{aligned}
$$

Eligible desires are those desires the agent believes are not satisfied. Recall that, according to rationality constraints in section 4, it is not rational for an agent to intend something it believes is already achieved or that is impossible. Notice that if a desire is conditional, then the agent should believe this condition is true.

As the initial set of desires, eligible desires may also be contradictory. Therefore, it is necessary to determine those subsets of the eligible desires that are jointly achievable. In general, there may be more than one subset of the eligible desires that are jointly achievable. Therefore, we should indicate which of these subsets are preferred to be adopted as intentions. We do this through the preference relation defined bellow.

Definition 7 (Desires Preference Relation $<_{Pref}$). Let $\mathcal{D}$ be the agent's desires, $\mathcal{D}'$ the set of eligible desires from $\mathcal{D}$, $\mathcal{P}(\mathcal{D}')$ the power set of $\mathcal{D}'$ and $R, S \in \mathcal{P}(\mathcal{D}')$. We say that $R <_{Pref} S$ (R is less preferred than S) if the biggest value for importance occurring in S and not occurring in R is bigger than the biggest value for importance occurring in R and no

t occurring in S; if there is no such biggest value in S, than R is less preferred than S if S has more elements than R.

According to this definition, the agent should prefer to satisfy first the most important desires. Additionally to preferring the most important ones, the agent adopts as much desires as it can.

Example 8. Given the set

$$D' = \{des(1, Ag, a, [0.5]), des(2, Ag, b, [0.3]),\\ des(3, Ag, c, [0.3]), des(4, Ag, d, [0.2])\}$$

of eligible desires:

1. for $d_1 = \{des(2,\ Ag,\ b,\ [0.3]),\ des(4,\ Ag,\ c,\ [0.2])\}$ and $d_2 = \{des(1,\ Ag,\ a,\ [0.5])\}$,we have $(d_1 <_{Pref} d_2)$. That is, d_1 is less preferred than d_2, since $A = 0.5$ for d_2 and $A' = 0.3$ for d_1;
2. for $d_3 = \{des(3, Ag, c, [0.3])\}$, we have $d_3 <_{Pref} d_1$, as $A = 0.2$ for both d_1 and d_3, but $\#d_3 < \#d_1$ ($\#d$ stands for cardinality of set d).

Notice that the preference relation is a pre-order relation. For instance, if an agent were to choose between $d_4 = \{des(2, Ag, b, [0.3])\}$ and $d_5 = \{des(3,\ Ag,\ c,\ [0.3])\}$, based only on the importance of desires and maximization of desires satisfied, it would not prefer either of them. And, indeed, according to the preference relation, we have that neither $(d_4 <_{Pref} d_5)$ nor $(d_5 <_{Pref} d_4)$. Based on this preference order, we define the preference graph that will be used to revise the mental states and the revision process.

Definition 9 (Desires Preference Graph). Let $\mathcal{D}$ be the agent's desires and $\mathcal{D}'$ the set of eligible desires from $\mathcal{D}$. Let $Revisable$ be the set

$$Revisable = \{unsel(D)/\exists des(D, Ag, P, A) \in \mathcal{D}'\}$$

and $index : \mathcal{P}(Revisable) \longrightarrow \aleph^+$ a function from the power set of $Revisable$ to natural numbers (zero excluded) that attributes a level number to elements of $\mathcal{P}(Rev)$. The *desires preference graph* is the graph defined by

1. $Rev(\mathbf{bottom}) = \{happens(E, T_i, T_f), act(E, A)\}$;
2. $Rev(i) = R \cup \{happens(E, T_i, T_f),\ act(E, A)\}$, where $R \in \mathcal{P}(Revisable)$ and $i = index(R)$;

3. $i \ll \mathbf{bottom}$, where $i = index(R)$, $R \in \mathcal{P}(Revisable)$ and $\nexists S \in \mathcal{P}\,(Revisable)$. $(S <_{Pref} R)$;
4. $j \ll k_1, \ldots, k_n$, where $j = index(R)$, $k_i = index(S_i)$ $(1 \leq i \leq n)$, $R, S_i \in \mathcal{P}(Rev)$ $(1 \leq i \leq n)$ and R is an antecedent of S.

The desires graph definition starts by defining the set of revisable literals, i.e., the set of literals that will have their truth value changed when performing revision in order to select a subset of desires. According to the definition, to each desire D there is a revisable literal $unsel(D)$ associated to it. Next, it defines a numbering function (the $index$ function) that assigns to each set composed by elements of $Revisable$ a number. This number is used as the level numbers (see section 2 about preferred revisions) in the preference graph. At the root of the preference graph ($\mathbf{bottom}$ level) the revisable literals are $happens/3$ and $act/3$, i.e., we try to see if there is a course of actions that jointly satisfies all the eligible desires. If this is not the case, then we have to select a subset of the eligible desires. This is the role of the $unsel/1$ literals. Their initial value is false and, as we may see in definition 10, they are attached to each eligible desire as a default literal. Therefore, initially they have no influence in the evaluation of desires. If we cannot find a course of actions to satisfy all eligible desires, we have to start checking subsets of $\mathcal{D}'$. The preference graph states that we prefer to revise first the $unsel/1$ literals associated with less important desires, and second preserving as many eligible desires as we can. That is, when revising the eligible desires set, the preferred revisions are those that eliminate first the less important desires, and the least possible amount of desires, as shown in the definition bellow.

Definition 10 (Candidate Desires Set). Let $\mathcal{D}$ be the agent's desires and $\mathcal{D}'$ the set of eligible desires from $\mathcal{D}$ with a preference graph associated with it. We call *candidate desires set* any set

$$\begin{aligned}\mathcal{D}'_C = \{&des(D, Ag, P, A)/(des(D, Ag, P, A) \in \mathcal{D}')\wedge \\ &(P' \cup \Delta \not\models_P \bot)]\}\wedge \\ &[\exists \Delta.(\mathcal{B} \cup \mathcal{T}Ax \cup \mathcal{R} \cup \Delta \models_P (holds\_at(bel(Ag, P), T), not\ unsel(D)))\end{aligned}$$

where

1. P' is the abductive framework

$$\langle \mathcal{B} \cup \mathcal{T}Ax \cup \mathcal{R}, \{happens(E, T_i, T_f), act(E, Act), unsel(D)\}, IC\rangle;$$

2. IC is a set of constraints of the form
   – $\{holds\_at(bel(Ag, P), T) \Leftarrow not\ \ unsel(D)\}$ for every $des(D, Ag, P, A)$ in $\mathcal{D}'$;
   – the constraints $IC(\mathcal{I})$ generated by intentions (see definition 4).

In the revision process, we mix abductive reasoning with defeasible reasoning, where the literal $unsel(D)$ is defeasible. Its intuitive meaning is *"Desire D should not be selected as an intention"*. If the agent believes it is possible to satisfy all of its desires (if it can abduce actions that satisfy all desires and satisfy all constraints), it will find a revision that contains only $happens/3$ and $act/2$. When constraints may not be all concurrently satisfied, it means that the adoption of all desires as intentions leads to contradictions, i.e., they are not jointly satisfiable.

Notice that contradictions do not arise if the actions necessary to satisfy two different intentions have contradictory effects. Recall that, according to the EC axioms, a property P holds if there is an action that initiates it or, alternatively, if $\neg P$ implicitly does not hold, and vice-versa for $\neg P$. Therefore, actions that make contradictory properties hold in fact just cancel each other. This allows us to avoid the *side-effect problem*[9]. If we allowed for this kind of situations to raise contradictions, we would be making the agent preclude intentions that have contradictory consequences, but that are not directly contradictory with each other, and making the agent intend the logical consequences of its intentions. On the other hand, if an action necessary to satisfy an intention cancels a property that is also an intention, a constraint is violated and that course of action is rejected. In this case, the revision will try to defeat intentions, changing the truth value of $unsel(D)$ literals. According to the preference graph, it will try to defeat those constraints that represent the less important desires, trying to preserve the maximum of the most important ones.

As we mentioned before, the desires preference relation is not an order relation. Therefore, it is possible to have more than one candidate set after a revision. However, if we consider only achievability and desires attributes as decision criteria, it makes no difference for the agent to adopt any of the candidate desires set[5][5].

Definition 11 (Primary Intentions). Let $\mathcal{D}$ be the agent's desires, $\mathcal{D}'_C$ a candidate desires set from $\mathcal{D}$. The *primary intentions* of an agent is the set $\{int\_that(D, Ag, P, A)$ / $des(D, Ag, P, A) \in Des'_C)\}$

Intentions as Refinements from Intentions Once the agent adopts its intentions, it will start planning to achieve those intentions. During planning, the agent will form intentions that are relative to pre-existing intentions. That is, they "refine" their existing intentions. This can be done in various ways, for instance, a plan that includes an action that is not directly executable can be elaborated by specifying particular way of carrying out that action; a plan that includes a set of actions can be elaborated by imposing a temporal order on that set[18]. Since the agent commits to the adopted intentions, these previously adopted intentions constrain the adoption of new ones. That is, during the elaboration of plans, a potential new intention is only adopted if it is not contradictory with the existing intentions and with beliefs.

Definition 12 (Relative Intentions). Let $\mathcal{I}_P$ be the set of primary intentions. A *planning process* is a procedure that, for each $i \in \mathcal{I}_P$, will generate a set of temporal ordered actions $\mathcal{I}_R$ that achieve i, such $\mathcal{B} \cup \mathcal{T}Ax \cup \mathcal{I}_P \cup \mathcal{I}_R$ is non-contradictory. The set $\mathcal{I}_R$ are the *relative intentions* of the agent.

[5] The revision process provided by the ELP framework defines a *sceptical revision*[1], that is the revision formed by the union of all the minimal program revision. This kind of approach prevents the agent from having to choose one of the minimal revisions. However, for intentions, this is not adequate, since we would like our agents to try to satisfy all the eligible desires it can.

The non-contradiction condition enforces again the notion of commitment, i.e., once an intention is adopted it constrains the adoption of new intentions.

Example 13 (cont. from example 5). As the robot starts working, it verifies what it is going to do (it adopts intentions). According to our definitions, it initially selects its eligible desires.

$$\mathcal{D}' = \{des(2, rbt, bel(rbt, stored(a1)), [0.3])\}$$

Since there is only one desire to be satisfied, there are no conflicts and it is adopted as a primary intention, i.e.,

$$\mathcal{I}_P = \{int\_that(1, rbt, bel(rbt, stored(a1)), [0.3])\}$$

Notice that none of the rationality constraints have been violated. Only one action is enough to satisfy its intention, namely

$$\mathcal{I}_R = \{int\_to(2, rbt, bel(rbt, store(a1)), [0.3])\}$$

4.2 Revising Intentions

In the previous section, we defined how the agent chooses its intentions. As we have seen, weighing motivations and beliefs means finding inconsistencies in competing desires, checking valid desires according to beliefs and intentions, resolving constraints imposed by intentions and desires, i.e., very expensive reasoning activities. It is now necessary to define *when* the agent should perform this process[**?**].

We argue that it is not enough to state that an agent should revise its intentions when it believes a certain condition holds, like to believe that an intention has been satisfied or that it is no longer possible to satisfy it, as this suggests that the agent needs to verify its beliefs constantly. Instead, we take the stance that it is necessary to define, along with those conditions, a mechanism that triggers the reasoning process without imposing a significant additional burden on the agent. Our approach is to define those conditions that make the agent start reasoning about intentions as constraints over its beliefs. Recall that we assume that an agent constantly has to maintain its beliefs consistent, whenever new facts are incorporated.

Definition 14 (Trigger from Intentions). Let $\mathcal{B}$ be the agent's beliefs set and $\mathcal{I}$ its intentions. We add to $\mathcal{B}$ the following *trigger constraints*:

– $(\bot \Leftarrow Now > T, not\ \ rev\_int)$, for each $(int\_that(I, Ag, P, A), int\_to(I, Ag, Act, A)) \in \mathcal{I}$;
– $(\bot \Leftarrow happens(E, T_i, T_f), act(E, Act), not\ rev\_int$, for each $int\_to(\ I, Ag, Act, A)) \in \mathcal{I}$.

The literal rev_int is part of the revisable set of beliefs, and its initial value is $false$. Whenever the agent revises its beliefs and one of the conditions for revising beliefs hold, a contradiction is raised. We identify such contradiction by testing if rev_int is in the selected revision for the beliefs set, i.e., if it has to have its truth value modified in order to restore consistency. The intention revision process is triggered when one of these constraints is violated.

The conditions we have defined so far are the usual ones defined by formal models of agents. As we have seen before, this characterization of intentions may lead to some fanatical behavior. Therefore, we need to adopt additional constraints that will avoid those unwanted behaviors. We take the stance that the same reasons that originated intentions may be used to break commitment associated to them[7]. If we accept that an intention originated from desires, it is reasonable to state that it is not rational to persist with an intention whose reasons are superseded by more urgent or important ones. The agent's normal behavior would be to weigh its competing desires and beliefs, selecting its intentions. The agent would commit to these intentions and they would constitute the filter of admissibility for other intentions. Also, the agent would try to satisfy those intentions, until successful accomplishment, impossibility (as usually defined), or *until some of his other desires that were not selected before would become eligible*, or *until the desires that originated them would not be eligible anymore*, re-activating the revision process that would weigh (again) competing desires and beliefs. Since the notion of commitment is preserved, the agent would not start this process every time there is a change in beliefs, but only if relevant conditions trigger the intention revision process, changing the agent's focus of attention[6]. These triggers are determined by the desires pre-conditions. We model this triggers using an approach similar to the normative constraints in [28,32].

Definition 15 (Trigger Constraints from Desires). Let $\mathcal{D}$ be the agent's desires and $\mathcal{D}'$ the set of eligible desires from $\mathcal{D}$. We define *trigger constraints from desires* as

1. For every $des(D,Ag,P,A) \leftarrow Body \in \mathcal{D}$ and not in $\mathcal{D}'$ with importance A bigger that the biggest importance in intentions, we define a trigger constraint $\perp \Leftarrow Body, not\ \ rev\_int$;
2. given the set of actions Δ abduced by the agent (see definition 10), for each $des(D, Ag, P, A) \in (\mathcal{D}' - \mathcal{D}'_C)$ with importance A bigger that the biggest importance in intentions, we define a trigger constraint $\perp \Leftarrow C_1, \ldots, C_n, not\ \ rev\_int$, where C_i ($1 \leq i \leq n$) are the conditions the agent could not bring about when selecting the candidate desires set.

The first constraint trigger is formed by the pre-conditions of those desires that were not eligible and that are more important than those that were evaluated. It means that if the pre-conditions of such desires become true, these desires (that were not considered during reasoning) become eligible. Therefore, it is necessary to re-evaluate desires and beliefs to check if this new desire may be brought about. The second constraint is formed by the pre-conditions of those eligible desires that, although more important, were not relevant when the agent made his choice. Notice that there are no triggers for those desires that were eligible but that were ruled out during the choice of a revision. This is so because they had already been evaluated and they have been considered less important than the other desires. Therefore, it is of no use to trigger the whole process again (i.e., to shift the agent's attention) to re-evaluate them.

Example 16 (cont. from example 5). The robot would have as triggers in its beliefs

$$\bot \Leftarrow happens(E, T_i, T_f), act(E, store(a1)), not\ rev\_int.$$
$$\bot \Leftarrow holds\_at(bel(rbt, \neg bat\_chged), T), not\ rev_int.$$

Suppose that, while executing the $store(a1)$ action, an event with $sense\_low\_bat$ happens. This would raise a contradiction in the belief revision, as property $bel(rbt,\ bat\_chged\ )$ would hold and would make the robot revise its desires and intentions. Now, the eligible desires would be

$$\mathcal{D}' = \{des(1, rbt, bat\_chged, [0.5]), des(2, rbt, stored(a1), [0.3])\}$$

But these are conflicting desires, therefore we must choose an appropriate subset of desires. There desires preference graph would be

$$Rev(1) = \{unsel(2)\} \cup Rev(\mathbf{bottom}) \qquad Rev(2) = \{unsel(1)\} \cup Rev(\mathbf{bottom})$$
$$Rev(3) = \{unsel(1), unsel(2)\} \cup Rev(\mathbf{bottom})$$
$$1 \ll \mathbf{bottom} \qquad 2 \ll 1 \qquad 3 \ll 2$$

and would produce as candidate desire set $\mathcal{D}'_C = \{des(1, rbt, bel(rbt, bat\_chged), [0.5])\}$. The robot's intentions then would be $\mathcal{I}_P = \{int\_that(\ 1, rbt, bat\_chged), [0.5])\ \}$ and $\mathcal{I}_R = \{int\_to(2,\ rbt,\ charge,\ [0.5])\ \}$.

5 Conclusion

The main contribution of this paper is to provide a formal model of agents that reduces the gap between agent specification and agent implementation. Adopting ELP as the underlying formalism has both preserved the main characteristics of formal models, namely the ability to formally define and verify agents, and has provided machinery the agent may use to reason. Besides that, our model presents other advantages, such as modelling both static and dynamic aspects of pro-active mental states[20] (see also [21] for a comparison of our work with many other formal BDI models). Our next step is to focus on the evolution of plans and communication, on how this affects the stability of intentions and integrate it to our framework.

References

1. J.J. Alferes and L.M. Pereira. *Reasoning with Logic Programming*. Springer-Verlag, Berlin, DE., 1996. Lecture Notes in Artificial Intelligence Series (LNAI 1111).
2. J.J. Alferes, L.M. Pereira, and T. Przymusinski. Belief revision in non-monotonic reasoning and logic programming. In C. Pinto-Ferreira and N.J. Mamede, editors, *Proceedings of the Seventh Portuguese Conference on Artificial Intelligence (EPIA'93)*, Berlin, Germany, 1995. APIA, Springer-Verlag. Lecture Notes on Artificial Intelligence (LNAI 990).
3. L. Beaudoin. *Goal Processing in Autonomous Agents*. PhD thesis, Birmingham University, Birmingham, UK, August 1994.

4. J. Bell. A planning theory of practical reasoning. In *Proceedings of the AAAI'95 Fall Symposium on Rational Agents Concepts*. AAAI, 1995.
5. J. Bell and Z. Huang. Dynamic goal hierarchies. In *Proceedings of the Second Workshop on Practical Reasoning and Rationality*, London, England, 1997. AISB Workshop Series.
6. L.M. Botelho and H. Coelho. Agents that rationalize their decisions. In *Proceedings of the II International Conference on Multi-Agent Systems*, Kyoto, Japan, 1996. AAAI Org.
7. M. Bratman. Planning and the stability of intentions. *Minds and Machines*, 2:1–16, 1992.
8. M. Bratman, D.J. Israel, and M.E Pollack. Plans and resource bounded practical reasoning. *Computational Intelligence*, 4:349–355, 1988.
9. M.E. Bratman. What is intention? In P.R. Cohen, J.L. Morgan, and M. Pollack, editors, *Intentions in Communication*, chapter 1. The MIT Press, Cambridge, MA, 1990.
10. P.R. Cohen and H.J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42:213–261, 1990.
11. M. Corrêa and H. Coelho. Around the architectural approach to model conversations. In *Proceedings of the Fifth European Workshop on Modelling Autonomous Agents and Multi-Agents Worlds (MAAMAW'93)*, 1993.
12. M. Corrêa and H. Coelho. A framework for mental states and agent architectures. In *Proceedings of the MASTAS Workshop at EPIA'97*, 1997.
13. M. Corrêa and S. Mendes. A computational approach to situation theory based on logic programming to design cognitive agents. In *Advances in Artificial Intelligence: Proceedings of the 12th Brazilian Symposium on Artificial Intelligence*. Springer-Verlag, 1995. (Lecture Notes on Artificial Intelligence 991).
14. C. Damásio, W. Nejdl, and L.M. Pereira. Revise: An extended logic programming system for revising knowledge bases. In *Knowledge Representation and Reasoning*. Morgan Kaufmann inc., 1994.
15. K. Devlin. *Logic and Information*. Cambridge University Press, 1991.
16. M. Georgeff and A. Lansky. Reactive reasoning and planning. In *Proceedings of the National Conference of Artificial Intelligence (AAAI'91)*. AAAI inc., 1991.
17. A. Haddadi. *Communication and Cooperation in Agent Systems: a Pragmatic Theory*. Springer-Verlag, Berlin, DE., 1996. Lecture Notes in Artificial Intelligence Series (LNAI 1056).
18. K. Konolige and M. Pollack. A representationalist theory of intentions. In *Proceedings of the XII International Joint Conference on Artificial Intelligence (IJCAI'93)*, Chambéry, France, 1993. IJCAI inc.
19. R. Li and L.M. Pereira. Knowledge assimilation in domains of actions: A possible causes approach. *Journal of Applied Non-Classical Logic*, 1996. Special issue on Inconsistency Handling in Knowledge Systems.
20. M.C. Móra, J.G. Lopes, H. Coelho, and R. Viccari. Affecting the stability of intentions. In E. Costa, editor, *8th Portuguese Conference on Artificial Intelligence*. Springer-Verlag, 1997.
21. M.C. Móra, J.G. Lopes, H. Coelho, and R. Viccari. Modelling agents with extended logic programa. In *International Workshop on Engineering of Intelligent Systems*. ICSC co., 1998.
22. N. Moussale, R.M. Viccari, and M. Corrêa. Tutor-student interaction modelling in an agent architecture based on mental states. In D. Borges and C. Kaestner, editors, *Advances in Artificial Intelligence: Proceedings of the Thirteenth Brasilian Symposium on Artificial Intelligence*, Berlin, Germany, 1996. SBC, Springer-Verlag. Lecture Notes in Artificial Intelligence (LNAI 1159).
23. H. Nakashima, I. Ohsawa, and Y. Kinoshita. Inference with mental situation. Technical report, Umezono, Tsukuba, Ibaraki, Japan, 1981. (TR-91-7).
24. J. Quaresma and J.G. Lopes. A logic programming framework for the abduction of events in a dialogue system. In *Proceedings of the Workshop on Automated Reasoning*, London, England, 1997. AISB Workshop Series.

25. A.. Rao. Agentspeak(1): BDI agents speak out in a logical computable language. In *Proceedings of the European Workshop on Modelling Autonomous Agents and Multi-Agents Worlds 1996 (MAAMAW'96)*, Berlin, Germany, 1996. Springer-Verlag.
26. A.S. Rao and M.P. Georgeff. Modelling rational agents within a BDI-architecture. In R. Fikes and E. Sandewall, editors, *Proceedings of the Knowledge Representation and Reasoning'91 (KR&R'91)*, San Mateo, CA., 1991. Morgan Kauffman Publishers.
27. K. Schild. On the relationship between BDI logics and standard logics of concurrency. In J.P. Müller, M.P. Singh, and A.S. Rao, editors, *Intelligent Agents V — Proceedings of the Fifth International Workshop on Agent Theories, Architectures, and Languages (ATAL-98)*, Lecture Notes in Artificial Intelligence. Springer-Verlag, Heidelberg, 1999.
28. M. Schroeder, Iara de Almeida Móra, and Luís Moniz Pereira. A deliberative and reactive diagnosis agent based on logic programming. In J. P. Müller, M. J. Wooldridge, and N. R. Jennings, editors, *Intelligent Agents III — Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages (ATAL-96)*, volume 1193 of *Lecture Notes in Artificial Intelligence*, pages 293–308. Springer-Verlag, Heidelberg, 1997.
29. Y. Shoham. Agent-oriented programming. *Artificial Intelligence*, 60:51–92, 1993.
30. M. Singh. *Multiagent systems: a theoretical framework for intentions, know-how, and communications*. Springer-Verlag, Heidelberg, Germany, 1994. Lecture Notes in Artificial Intelligence (LNAI 799).
31. B. van Linder, W. van der Hoek, and J. J. Ch. Meyer. Formalizing motivational attitudes of agents: On preferences, goals, and commitments. In M. Wooldridge, J. P. Müller, and M. Tambe, editors, *Intelligent Agents II — Proceedings of the Second International Workshop on Agent Theories, Architectures, and Languages (ATAL-95)*, volume 1037 of *Lecture Notes in Artificial Intelligence*, pages 17–32. Springer-Verlag, Heidelberg, 1996.
32. G. Wagner. A logical and operational model of scalable knowledge and perception-based agents. In *Proceedings of the European Workshop on Modelling Autonomous Agents and Multi-Agents Worlds 1996 (MAAMAW'96)*, Berlin, Germany, 1996. Springer-Verlag.

8 Iberamia98(WDAI) – ”Pedagogical Games using an ITS Architecture”

Este artigo, apresentado no ”workshop” de IA Distribuída do Iberamia98, mostra a utilização do modelo de agentes aqui apresentado na definição do módulo cognitivo de um tutor inteligente para o ensino conceitos ecológicos para alunos do ensino básico, o MCOE. A definição das estratégias do tutor é feita utilizando um agente BDI, e aproveitando-se o fato do modelo ser tanto formal como executável. Trata-se do primeiro experimento com o modelo formal executável, validando as hipóteses apresentadas até então: a mudança de paradigma, o modelo simultaneamente formal e executável, os axiomas parametrizáveis.

Pedagogical game using ITS architecture

Lucia M. M. Giraffa
giraffa@inf.pucrs.br
II/PUCRS – CPGCC/UFRGS

Michael da C. Móra
michael@inf.ufrgs.br
CPGCC/UFRGS – II/PUCRS

Rosa M. Viccari
rosa@inf.ufrgs.br
CPGCC/UFRGS

Instituto de Informática – PUCRS
Av. Ipiranga 6681 – prédio 16 – Porto Alegre – RS – Brazil - 90610-900
Curso de Pós-Graduação em Ciência da Computação – UFRGS
Av. Bento Gonçalves 9500 – Bloco IV – Porto Alegre – RS – Brazil

Abstract

This paper presents some design principles for pedagogical games, illustrated by MCOE (Multi-agent Co-operative Environment). MCOE is a game modelled through agent's techniques using a *multi-agent system* architecture composed of a society of agents which work to achieve a common goal: to assist a student to fight against pollution resulting from foreign elements (polluters) and to maintain the equilibrium of the environment. The system has two kinds of agents: reactive (designed and implemented using the object-oriented approach) and cognitive (designed with a mental state approach). The cognitive agent of the MCOE tutoring system uses the agent model proposed by Móra et al. in [MOR 98b]. The agent model used allows us both to formally define the cognitive agent and to execute it to verify the accuracy of the model.

The interactions between students and the system are performed in a game-like fashion, using multimedia resources. One of the characters of the game, the Ecologist, that represents the Tutor, uses multiple teaching strategies to give advice for the students.

Keywords: Intelligent Tutoring Systems, Multi-agent Systems and Educational Game.

1. Introduction

In recent years, many systems for educational purposes have adopted a multiagents approach. This approach is useful to overcome the traditional restrictions to build a strong student model, and to better explore the interaction and dynamic changes in teaching-learning environments. Intelligent Tutoring Systems (ITS) have been implemented using different approaches [MOU96; AND97; COL97; LES97; GIR97; GIR98; SIL97; SIL98]. In Intelligent Tutoring Systems (ITS) and Intelligent Learning Environments (ILE) we can consider agents as a Pedagogical Agents.

Pedagogical agents have a set of normative teaching goals and plans for achieving these goals (e.g., teaching strategies), and associative resources in the learning environment [SIL97]. Tutoring agents are entities whose ultimate purpose is to communicate with the student in order to efficiently fulfill their respective Tutoring function, as part of the pedagogical mission of the system [MIT97]. They have some fundamental properties: autonomy, social ability, proactiveness, and persistence. Pedagogical agents can act as virtual Tutors, virtual students, or virtual learning companions that can help students in the learning process.

We adopt a *mentalistic approach* to implement the system, where the term agent means a computer system that can be viewed as consisting of mental states such as beliefs, intentions, motives, expectations, obligations and so on.

MCOE is a pedagogical game that simulates a lake with plants, and different types of fish. These elements are typical of real world environments, such as the river in our city and its surroundings.

Modelling the environment represents a new phase of our work that began with the design and implementation of an educational game named Ecological [RAA96]. That system aimed to test the knowledge domain and made it possible to observe the dialogues between human agents (students and Tutor) in a real situation. Prior to achieve the current state of our research we built a first prototype using a multi-agent architecture named Multi-Ecological [GIR97]. This version has many of the early characteristics: only one student, the game interface and simple agent architecture. The Multi-Ecological version is an agent-based system but not a real distributed multi-agent system like MCOE is. We had to change our approach to better achieve our research goals. Therefore we built a testbed to observe the interactions between students and the Tutor, by introducing one more student and creating a more complex situation.

The paper is divided into seven sections. Section 2 describes the environment modelling. Section 3 describes the multi-agent architecture designed for the system. Section 4 describes details of the two kinds of agents. Section 5 describes the system's pedagogical aspects. Section 6 presents our first results and some considerations about this approach, and the references used to support our work are presented on section 7.

2. The MCOE system

The conception of the 'MCOE' was based on an ITS architecture [GIR98; GIR98b]. This architecture is composed of a hybrid society of agents that work to achieve a common goal: to fight against the pollution resulting from foreign elements (pollutants) and to maintain the system equilibrium. We have reactive agents (bottom of the lake, micro-organisms - plankton, water, plants and three types of fish) and cognitive agents (the Tutoring agent, the ecologist, and students represented by characters).

We designed the system to be played by two students using different machines. They can be in the same physical place or not. The first student chooses a character to play using four options: Mother Nature, Mayor, Citizen, and Tourist. After that, the second student can choose one of the three remaining characters. The system defines the game configuration (foreign elements that will cause pollution) by a sorting process using a random function. The students first see a lake in equilibrium, which soon begins to show the action of the polluters. Their challenge is to maintain the equilibrium and fight the action of these foreign elements for a period of ten minutes[1].

The number of plants, micro-organisms and fish swimming in the lake, the water transparency and its pH, and the appearance of the bottom of the lake indicate the situation. There is a visual gauge (the Ecometer) that helps the user to observe the conditions of the lake. A balanced situation is shown when the gauge is full and green. As the equilibrium is lost, the colour changes gradually to yellow, finally changing to red when a dangerous situation is reached.

The system equilibrium is based on the energy level, the occasional predatory actions that can happen in the environment and the number of elements that remain in the system after the pollutants impact. The fish, plankton, and plants also reproduce using a function based on the energy level. However, when they reproduce they naturally loose energy, because they give energy to their descendants. If an agent dies (zero energy), it is removed from the system.

The system was built with Visual C++. The game's main window is implemented with DirectX, and the scenery and its elements are implemented with Direct3D Retained Mode (a tool for building interfaces with resources for creating high performance 3D scenes). The controls that the player can select during the game are implemented as bitmaps controlled by DirectDraw (a programming interface with resources for bitmaps manipulation).

The help is being implemented as an ordinary Windows application using the GDI (without DirectX).

Each agent in the scenery possessing a physical representation is related to a frame. A mesh of polygons and vectors that composes the frame: position, orientation and displacement direction relative to the frame of the universe. The objects were models in 3D Studio MAX 1.0. The figure 1 presents the interface.

The reactive agents have interchanging capability and were designed following previous

[1] The choice of ten minutes is because the game was designed to be played during classroom time, in a context of a real class situation, and a pre-defined period of time is necessary for the teacher to make her/his class plan.

specification [GIR97].

The relationship between the agent and the environment is affected by the occurrence of external incentives (polluters) that cause some partial reaction of the agent. Each example of an agent can have reaction that might differ even if the same incentives is present and will always change during the lifespan of each one.

Initially, we made a set of actions rules for each agent's reaction. These rules control the behaviour pattern (the personality) of each agent. These rules are connected with the incentives used to change the agent's behaviour. Some possible incentives that the agents can receive are *birth*, *death*, and the *time to pass*, and *to collide against the limits of the scenery*. For each one of those incentives they can select reactions such as *to be born*, *to die*, *to move*, *to change swimming direction*. For example, it is possible to choose the action of changing swimming direction, or dying for the incentive of colliding against the limits of the scenery.

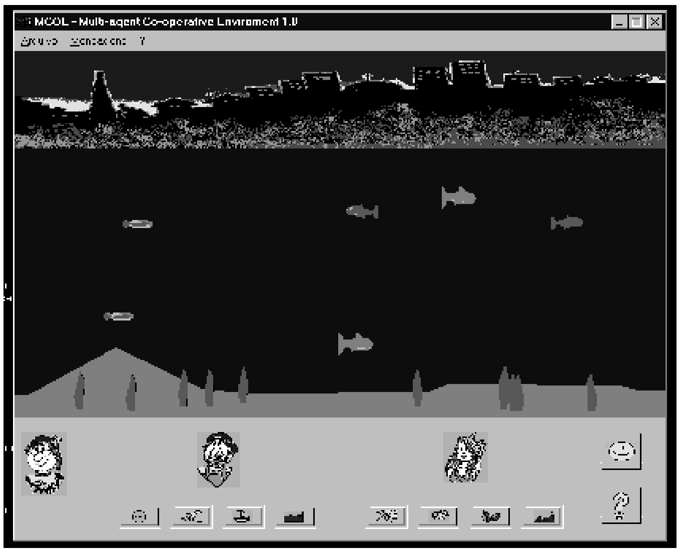

Figure 1: MCOE system interface

The reactions can be altered during the agent's life. It is enough substitute some action for another and another substitutes that action through a new incentive. That is the case of the action of *to move* used in the swimming of the fish. In order to guarantee that the fish will not just swim in straight line the action of changing the swimming direction is activated. After that, the action of changing the swimming direction is replaced by the action of moving.

The incentives of the environment are implemented as events that the agents receive. The actions are objects with a function, which implements the corresponding change in the agent's state. The agents are repositories of actions and data. The system control is always tracing each event for the respective object-action that was instanced to treat that event.

The relationship between the agent and the environment is affected by the occurrence of external incentives (polluters) that cause some partial reaction of the agent. Each example of an agent can have a reaction different to the same incentive that the others have and, it can also alter its reactions along the life.

3. The architecture of the 'MCOE' system

The environment is a closed heterogeneous society, based on distributed control (knowledge, data, and skill) among its different agents. The cognitive agents are goal-based agents, which pursue the maintenance of the ecological equilibrium as their goal. Figure 2 shows the distribution of agents in the system.

The Ecologist (tutor) is inserted in the environment and uses its knowledge and skill to control the pollution through the tactics used by selected behaviour. It receives information from the environment through sensors. This forms a set of inputs that allow the ecologist to evaluate the situation of the environment.

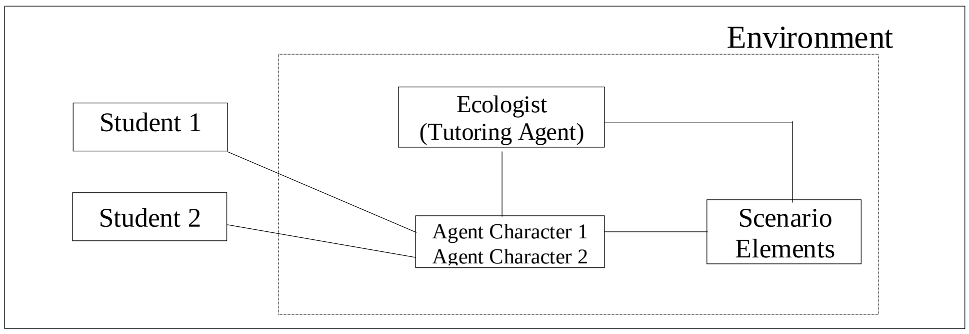

Figure 2: Distribution of agents in the environment

The student model has mental states represented by:

- A set of mental states (*desires, beliefs, and expectations*) concerning the environment (we call this the knowledge of the environment);
- A set of mental states about her/his knowledge;
- A set of mental states about the Ecologist (the Tutor) behaviour and its knowledge;
- Beliefs about what tool to use with specific foreign elements (we call this the knowledge about foreign elements);
- Sensors for receiving information about the environment;
- Dialog window to exchange messages with another student;
- Dialog window to receive messages from Tutor.

The student receives messages from the other student, the energy level of the scenario elements, and the Tutor advice. S/he uses it jointly with her/his own beliefs to build a strategy in order to fight against the pollution. The strategy will be built with the other student in a dynamic way. They have a short time strategy because they are testing hypothesis about the scenario reaction and they can see the impact of their actions on screen. If they have some doubts about what is going on with these elements they can consult the Help button and find information about elements of scenery.

The tutor also has architecture with:

- a set of mental states concerning the environment and about the students based on formalism of Mora et al. [MOR97;MOR98]. It is a BDI model that represents the desires and beliefs of the tutor. The knowledge about the students (beliefs about the student's beliefs and intentions) is in the tutor's beliefs;
- Knowledge about what tool to use with specific foreign elements. This is also part of the tutor's beliefs, in the formal model;
- Knowledge about energy levels of scenario elements and about some parts of the tutor's beliefs;
- Rules to help it to decide what action should be taken according to the changes of the students' mental states. This is also part of the tutor's beliefs;
- Sensors for receiving data about the environment.

The kernel of the Tutor architecture is the Behavioural Decision centre (figure 3).

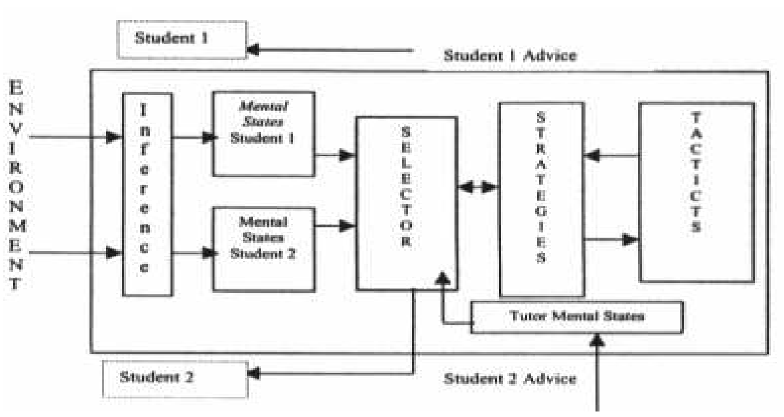

Figure 3: Behavioural Decision Centre

The tutor receives information about each student model. The information is composed by the mental states of each student, the students' action (tool), and the environment energy levels checked by the control. The tutor uses the information to select a specific strategy with the associated tactics, according to the selector module. The Tutor observes the interactions without acting on the environment, but advising the students. The tutor does not know what will happen during the game, i.e., it does not know what and how many foreign elements will appear in the scenery. The students' mental states and their actions (tool activation) are the perceptions that the tutor receives from the environment. The system works as if the tutor hears and sees what happens in the environment. These would be its sensorial capabilities.

The set of the students' mental states is obtained through a module of inference of attitudes, as presented in Quaresma [QUA97]. The dialogues are seen as composed by a sequence of speech acts. According to the [SEA84], to speak is to act.

4. The agents

The *reactive agents* are modelled using object techniques. The object structure is presented in figure 4.

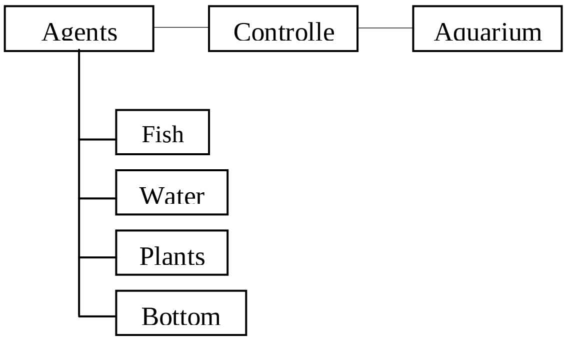

Figure 4: Objects distribution

The *Agents* (header) is responsible for agents' basic reactions (presentation, energy level, position, velocity, acceleration, and other physical states). This object is also responsible for the agent's final behaviour. This header contains the basic structure used to build the other agents.

The *Controller* co-ordinates the different system elements. It controls the agent's behaviour during the simulation. It also works as a mediator in the agent society (between agents and their relations with the environment).

The *Aquarium* is responsible for representing the environment conditions (water appearance) related to the pollution level, pH, oxygen level, and so on. All of these effects are pre-determined by a set of rules.

The *cognitive agents* are modelled through a set of mental states. The model of each student and the tutor contains a set of basic beliefs, desires and expectations. From this set emerges the dynamic selection done by Tutor to select a personal teaching strategy.

We start from Bratman's analysis about intentions, its role in rational reasoning and how it relates beliefs and desires. According to Bratman [BRA 90], since agents are assumed to be resource-bounded, they cannot continuously evaluate their competing beliefs and desires in order to act rationally. After some reasoning, agents have to commit to some set of choices. It is this choice, followed by a commitment that characterises the intentions. Our model does not define a complete agent, but only the cognitive structure that is part of the agent model.

An *agent cognitive structure* is a tuple $\langle B, D, I, T \rangle$ where *B* is the set of agent's beliefs, *D* is the set of agent's desire, *I* is the set of agent's intentions and *T* is the set of time axioms, as defined above. The *desires* of the agent is a set of sentences *DES(Ag,P,Atr) if Body*, where *Ag* is an agent identification, *P* is a property and *Atr* is a list of attributes. Desires are related to the state of affairs the agent eventually wants to bring about.

Beliefs constitute the agent's information attitude. They represent the information that agents have about the environment and about themselves. The set *B* contains sentences describing the problem domain using ELP[2]. An agent *A* believes that a property *P* holds at a time *T* if, from *B* and *T*, the agent can deduce *BEL(Ag,P)* for the time *T*. We assume that the agent continuously updates its beliefs to reflect changes that it detects in the environment. We assume that, whenever a new belief is added to the beliefs set, consistency is maintained.

Intentions are characterised by a *choice* of a state of affairs to be achieved, and a *commitment* to this choice. Thus, intentions are seen as a compromise, which the agent assumes with a specific possible future. This means that, differently from desires, an intention may not be contradictory with other intentions, since it would not be rational for an agent to intend something that ii believes is impossible. Once an intention is adopted, the agent will pursue that intention, planning actions to accomplish it, re-planning when a failure occurs, and so on. Agents must also adopt these actions to achieve their intentions.

Tutor and students have the same intention: to control the pollution and maintain the system equilibrium. The students have the desire to win the game. The Tutor desires the students learn to control the system equilibrium. During the interaction more desires, beliefs, and expectations arise.

Due to the limitation of space, we selected a small piece of a dialogue to show how the logical framework is used.

The system places plenty of pollution in the screen and it creates a complex situation. The foreign elements are 2 dredges, 2 Jet Ski, 2 industrial sewer, chemical pollution and 2 gas station. The foreign elements are placed in the scenery at different moments and in different places, in order to increase the difficulties during the game and to promote cumulative effects through interference in the scenery.

Student1 chooses to be the mayor and student2 chooses to be Mother Nature. Both mayor and Mother Nature are characters with plenty of resources to combat the pollution. In the beginning the system activates the dredge and the gas station. The energy level decreases: Plants –40%, Micro-organism –50%, Bottom –50%, Fish –20%, Water –20%. At this time, the Ecometer shows the energy level and the colour of some scenery elements changes.

The tutor has only one purpose: to aid the student to maintain an adequate energy level in the environment. It believes that it may aid by sending the students messages. The contents of these messages will depend on the strategy adopted by the tutor at the moment. These desires and beliefs are modelled as follows

DES(tutor,aid_students).
BEL(tutor,aid_students) if BEL(tutor, send_message)

[2] The formalism we are using is *logic programming extended with explicit negation* (ELP) with the ***W**ell-**F**ounded **S**emantics e**X**tended for explicit negation* (WFSX). ELP with WFSX (simply ELP) extends normal logic programs with a second negation named *explicit*, in addition to the usual *negation as failure* of normal logic programs, which is called *implicit negation* in the ELP context. More details in [MOR97; MOR98].

When there is only one desire, this desire is the only candidate to intention. It will be adopted as intentions if the tutor believes that there is a sequence of actions that student can execute. But, at this moment, the tutor depends on knowledge about the students' mental states in order to decide what to do. Since it does not have (yet) this knowledge, it waits. But, the tutor inserts in its belief triggers[3] that will make itself reconsider its options when interaction occurs. For instance, the tutor has the following belief

BEL(tutor, send_message) if BEL (tutor,next(BEL (Student, receive_aid))),
BEL(energy_level_ecometer >= 70),
message("Continue paying attention to the energy level").

meaning that if it believes the student expects some help, and it believes the energy level is above 70, than the only help it can offer is to advise the student to pay attention to the energy level. This belief could be placed as triggers

$$\begin{aligned}\perp \leftarrow & BEL(tutor, next(BEL(Student, receive\_aid))),\\ & BEL(energy\_level\_ecometer(EE)),\\ & EE >= 70.\end{aligned}$$

i.e., when the pre-conditions for tutor's action of sending a message are satisfied, it is in condition to aid the student. The agent models then transform the desire in an intention to satisfy the tutor to send the message.

Once the tutor has this intention, it uses its beliefs about the environment (tool, energy level, foreign elements, scenery elements), about how to fight against pollution in the environment (different strategies on the use of the tools to eliminate pollution), about how to advise the students in order to help them to control the environment and so on.

5. Pedagogical aspects

The actions of the characters have two basic pedagogical goals:

- show the foreign elements most frequently found in our reality and the possible actions which can be taken against them;
- allow the student to identify positive and negative actions for the equilibrium of the environment.

According to [ARK96], learning is an interactive process and learners construct their own knowledge actively interacting in a world, interpreting their own experiences. So, we developed an ITS that enables learners to develop processes of interaction and think about the different implications about pollution.

Akhras and Self [ARK97] point out that the learning experience promoted by the system must be adapted to the learner's individual needs at each time. As a result, the idea of an instructional planner has been conceived as the computational mechanism that performs these adaptations. Our system incorporates these ideas and presents an alternative way to investigate the possible personal teaching strategy connected with each student style based on her/his mental states set.

Each student plays alternately and the Tutor considers her/his own set of mental states to decide what kind of advice fits into the situation. The set of student's mental states reflects her/his beliefs and perceptions about the environment. The exchanged messages between two students will also interfere in their behaviour (i.e., their own set of mental states). However, this kind of relationship occurs out of the Tutor control.

If there is a conflict between the students' behaviour, they will solve such conflict out of

[3] A trigger is a mechanism that starts the reasoning process whever it is necessary.

the Tutor control through the chat window. Therefore, the Tutor will “know” both (conflict and solution) because the set of mental states of each student will show it.

We provide a set of tactics to implement the Tutor’s action related with the selected classified strategies. The teaching strategies have implicit knowledge about whom to teach and the actions to do.

The Tutor behaviour does not repeat the same tactic, if possible. It always tries to use a different tactic (under the same strategy) to help the student. Nevertheless, we must remember that a selected strategy is not static, i.e., we are building a system that can change the strategy during the same section. This change will be supported by each student mental state analysis and by the Environment State at each moment. The Tutor beliefs, desires and expectations about the student mental states will guide its behaviour.

This is an important point in our system. We are developing an environment that reacts with the student behaviour expressed by her/his mental state activity.

To better understand the behaviour (tutor behaviour) associated with a specific strategy we have modelled three different situations based on real dialogues between students based on the first version of our system [RAA96].

The student chooses the character tourist and the following foreign elements: garbage (toxic products), aquatic sports (Jet Ski), and predatory fishing. This situation has a medium level complexity with three foreign elements, and the character chosen by the student does not have the better tools to fight against the pollution. The tutor allows the student to combine the tools and create strategies to fight against the pollution. In this case, the student and Tutor must pay more attention to their actions and observe carefully the environment reactions.

The Tutor behaviour may be as follows:

- Like a *Reactive Tutor*:

The tutor uses tactics “Give a message explaining the corresponding rule”, and “Give a message explaining the consequence of this action” with associated messages previously selected and stored on a message file. It also uses tactics as “Give a message explaining the consequence of this action” with the corresponding messages. The tactic “Give an example using a similar situation” can be used with the connected examples. However, the Tutor will prefer to use the tactic “Give a message explaining the consequence of this action” because in this situation the student has not a very powerful character. With this configuration the tools are not powerful enough to fight against the foreign elements, and the tactic “Give a message explaining the consequence of this action” can be used in order to reinforce the student attention.

- Like a *Coaching Tutor*:

The Tutor has no better tool to show the student, so it cannot use the tactic “Give a message with the best option (tool)”. It uses the tactics “Show the scenario elements that are important to observe”, “Show the corresponding rule and do not explain anything”, “Show the characteristic of the foreign element”, and “Sound the alarm when student selects a dangerous option” to reinforce student orientation. These orientations are based on the decrease of the energy level of the scenario elements.

- Like an *Assistant Tutor*:

The tutor uses the following tactics: “Give a message to explain the corresponding rule”, “Give a message to explain the consequence of this action”, “Give an example using similar situation”, “Show the scenario elements that are important to observe”, “Show the corresponding rule and do not explain anything”, and “Show the characteristic of the foreign element”. The Tutor must be cautious in this configuration because there is no previous answer (specific rule to fight against this combination).

6. Final considerations

The use of agent’s technique is a good option from the system-engineering viewpoint because it permits data control and knowledge skill distribution. It allows us to investigate, in a new approach, some of the classical problems faced by ITS designers. The results showed us that we could have the following benefits:

- We can build a more powerful and realistic interface for simulations environments like games (designed as an ITS). The actions on screen are more dynamic and express better what is going on in the system;
- Using agents we can design more sophisticated and flexible student and Tutor models, as well as investigate aspects like different behaviour and multiple strategies in a more dynamic way.

This is possible because we can reduce the processing cost of hardware and software by distributing the computational weight through the different elements of the system. By using Multi-Agent Systems to model an ITS, we can think about alternative options to combine some other educational environments modalities and to build more powerful systems from a pedagogical viewpoint.

The Mental States approach allows us to trace more precisely the choreography of the interaction between Tutor and Students. These results can be used to improve future modelling, and help us to build better students and Tutor models. The Mental States approach has much more to provide us and it will demand more research to improve properties as well as to develop ways to represent teaching/learning situations.

The system will be tested in elementary schools. We are developing tools to evaluate some aspect of the system according to the point of view of the teacher and that of the student.

A written report about student's behaviour and observations about the interface and game performance. It is a form with a list of blanks to fill for all aspects studied. We have all aspects in a list and just put a stick to choose one of them.

We are building these instruments under expert's supervision from Education Institute of our University.

Acknowledgements
We say thank you to Leandro Bernsmüller (implementation and Computer Graphics support), and André Raabe (evaluation process at schools).

7. References

[AND97] Andre, E.; Muller, J.; Rist, T. Life-Like Presentations Agents: A New perspective for Computer Based Technical Documentation. AI-ED97: Eighth World Conference on Artificial Intelligence.

[ALF 96] Alferes, J.J.; Pereira, L.M. Reasoning with Logic Programming. Springer-Verlag, Berlin, DE., 1996. **Lecture Notes on Artificial Intelligence Series** (LNAI 1111). Cambridge, MA, 1990.

[ARK96] Akhras, F; Self, J. From the Process of Instruction to the Process of Learning: Constructivist Implications for the Design of Intelligent Learning Environment. Euro AIED: European Conference on Artificial Intelligence in Education, 1, 1996. **Proceedings…** Lisbon: Calouste Gulbenkian, 1996.

[ARK97] Akhras, F; Self, J. Modelling Learning as a Process. AI-ED97: Eighth World Conference on Artificial Intelligence in Education, 8,1997. **Proceedings...** Kobe: Japan, 1997.

[BRA 90] Bratman, M.E.. What is intention? In P.R. Cohen, J.L. Morgan, and M. Pollack (Ed.): **Intentions in Communication**. The MIT Press, 1990.

[COH 87] Cohen, P. R., Levesque, H. J. Intention = Choice + Commitment. Sixteen National Conference on AI, 6.,1987, **Proceedings**.[S.l.],Springer Verlag, 1987.

[COL97] Colazzo, L.; Silvestri, L. The pragmatics of the Tutor: A proposal of modelling. AI-ED97: Eighth World Conference on Artificial Intelligence in Education - Workshop V: Pedagogical Agents, 8., 1997. **Proceedings...** Kobe: Japan, 1997.

[GIR97] Giraffa, L.M.M.; Nunes, M.A.; Viccari, R.M. Multi-Ecological: an Intelligent Learning Environment using Multi-Agent architecture. MASTA'97: Multi-Agent System: Theory and Applications. **Proceedings...** Coimbra: DE-Universidade de Coimbra, 1997.

[GIR98] Giraffa, L.M.M; Viccari, R.M.; Self,J. Improving tutoring activities using a Multi-Agents system Architecture. Twenty-ninth SIGCSE Technical Symposium on Computer Science Education, 29,1998. **Proceedings...** Atlanta: Georgia, 1998.

[GIR 98a] Giraffa, L.M.M; Viccari, R.M.; Self.J. Multi-Agent based on pedagogical games. ITS'98- Fourth International Conference on Intelligent Tutoring Systems. **Proceedings…** San Antonio, Texas,1998.

[LES97] Lester, J. et al.. Mixed Initiative Problem Solving with Animated Pedagogical Agents. AI-ED9: Eighth World Conference on Artificial Intelligence in Education - Workshop V : Pedagogical Agents, 8.,1997. **Proceedings...** Kobe: Japan, 1997.

[MIT97] Mitsuru, I. Et al.. Opportunistic Group Formation. AI-ED97: Eighth World Conference on Artificial Intelligence in Education - Workshop V: Pedagogical Agents, 8., 1997. **Proceedings...** Kobe: Japan, 1997.

[MOR 97] Móra, M.C.; Lopes, J.G.; Coelho, J.G.; Viccari, R. Modelling dynamic aspects of intentions. In E.Costa (Ed.): 8th Portuguese Conference on Artificial Intelligence. Springer-Verlag, **Proceedings...**, 1997.

[MOR 98a] Móra, M.C.; Lopes, J.G.; Coelho, J.G.; Viccari, R. Modelling agents with extended logic programming. In International Workshop on Engineering of Intelligent Systems. ICSC co., 1998.

[MOR 98b] Móra, M.C.; Lopes, J.G.; Coelho, J.G.; Viccari, R. BDI models and systems: Reducing the gap. In: Agents Theory, Architecture and Languages Workshop. **Lecture Notes on Artificial Intelligence**, Springer-Verlag, 1998.

[MOU 96] Moussale, N.M.; Viccari, R.M.; Correa, M. Intelligent Tutoring Systems Modelled Through the Mental States. In: Lectures Notes on Artificial Intelligence - SBIA'96. Borges, D. and Kaestner, C.(Eds.). **Proceedings…**Berlin: Springer Verlag, 1996.

[QUA97] Quaresma, P. **Inferência de atitudes em diálogos.** Lisboa: FCT/UNL, 1997. (Ph.D. Thesis - Portuguese)

[RAA96] Raabe, A.L.A; Javimczik, A .M.; Giraffa, L.M.M. Eco-Lógico: Ambiente interativo para suporte ao ensino de educação ambiental. Simpósio Brasileiro de Informática na Educação, 7., 1996, **Anais...** Belo Horizonte: DCC/UFMG, 1996. (Portuguese)

[SEA 84] Searle, J.. What is an intentional state? In H.Dreyfuss and H.Hall (Ed.): **Intentionality and Cognitive Science**, (42), 1984.

[SIL97] Silveira, R.A.; Viccari, R.M. Projeto Eletrotutor: Desenvolvimento e Avaliação de Ambientes Inteligentes de Ensino-Aprendizagem. CLEI-PANEL'97: XXIII Conferencia Latino americana de Informática, 23, 1997. **Proceedings…** Valparaiso: Chile, 1997.

[SIL98] Silveira,R.A.; Viccari,R.M.. Distance Learning: From the Intelligent Tutor Paradigm To A Multiagents Architecture Lectures Notes on Artificial Intelligence - SBIA'98. Oliveira,F.(Ed.). Berlin: Springer Verlag, 1998. (Submitted paper)

9 SBIA98 –"Modelling the MCOE Tutor using a Computational Model"

Este artigo, apresentado no SBIA98, mostra o desenvolvimento do tutor MCOE utilizando o modelo formal e executável aqui apresentado. Neste artigo, é utilizada uma primeira versão da ferramenta de simulação do comportamento do agente derivada da teoria desenvolvida. A coreografia de estados mentais apresentada mostra como o modelo pode dar origem à implementação do agente. Aqui, não somente se experimenta o modelo formal e executável, mas também a ferramenta construída a partir deste modelo. Esta ferramenta permite que construa o "kernel cognitivo" de um agente, e que este seja embutido em aplicações que implementam os demais aspectos do agente, notadamente sensores e atuadores dos mesmos.

Modelling the MCOE Tutor using a Computational Model

Lúcia M. M. Giraffa[1,2] Michael da C. Móra[2,1] Rosa M. Viccari[2]

[1]Instituto de Informática – PUCRS
Av. Ipiranga 6681 – prédio 16 – Porto Alegre – RS – Brasil 90610-900
{giraffa,michael}@inf.pucrs.br
[2]Curso de Pós-Graduação em Ciência da Computação – UFRGS
Av. Bento Gonçalves 9500 – Bloco IV – Porto Alegre – RS – Brasil
{giraffa,michael,rosa}@inf.ufrgs.br

Abstract: In this paper, we present the definition of the cognitive agent of the MCOE tutoring system using the agent model proposed by Móra et all in [23]. MCOE (Multi-agent Co-operative Environment) is a game modelled using a *multi-agent system* architecture composed of a society of agents who work to achieve a common goal: to assist a student to fight against pollution resulting from foreign elements (polluters) and to maintain the equilibrium of the environment. The system has two kinds of agents: reactive (designed and implemented using the object-oriented approach) and cognitive (designed with a mental state approach). The agent model used allows us both to formally define the cognitive agent and to execute it to verify the accuracy of the model.

Key words: Intelligent Tutoring Systems, Multi-agents Systems. Agent Modelling, Mental States.

1 Introduction

In recent years, a new paradigm arose for educational environments: to make more than one student interact with the same environment under artificial tutor supervision. From the educational point of view, this is not a new idea. However, it became feasible through the development of hardware and software that allow us to connect people using computer networks and related technologies. The impact of these technologies on educational software research was immediate. The researches have begun to introduce these new possibilities to improve educational environments.

Nevertheless, the student model remains the weak part of such systems. There are many factors that contribute to this weakness (hardware and software limitation, and techniques to model the student, knowledge representation, and others). However, the strongest restriction is our imprecise knowledge about the mental activities of the students during teaching/learning process. In order to build an ITS with a good student model, we must understand what is happening in the student's mind during the interaction. We need to understand the process and reproduce it in the machine. Much has been done to understand this process, according to different viewpoints: psychological, educational and computer science. The work of Winograd and Flores [30]; Devlin [11], Gagne [13]; Corrêa [6, 7, 8] and Moussalle [24] are examples of the such improvements.

The current technology and our limited knowledge about the human learning process still do not allow us to build the ideal ITS. At this moment, the researchers' efforts to find a computational theory that explains the human information process has not produced the appropriate answers. In fact, we do have paradigms that try to explain how the information is processed in the human mind. Nevertheless, those paradigms do not allow us to detail the process at the level that we needed. We have now different tools and new possibilities. The multimedia techniques and agents programming paradigm are some of these new technologies that may transform the way to design ITS.

Many systems for educational purposes have adopted the agents' paradigm to better explore the interaction and dynamic changes in teaching-learning environments. As Khuwaja [18] said, even though Intelligent Tutoring Systems have been implemented with relative success they are not practical enough to be used in the real world. The restrictions of these systems can be overcome when we attempt to introduce the notion of co-operation in the teaching-learning process, using a multi-agents focus. Using the agent's paradigm and mental state choreography to improve the student model and the interactions between tutor and students is a new possibility that arose with Corrêa's [6] work. After this work another experiment built by Moussalle [24] showed that is possible to trace some student's mental activities using this approach.

There is not, in the research community, a consensual definition of what an agent is. We adopt a *mentalistic approach,* where the term agent means a computer system that can be viewed as consisting of mental states such as beliefs, intentions, motives, expectations, obligations and so on. In this case the question of what an agent is replaced by the question of what entities can be viewed as possessing mental states. They find their justification in the already classical argument by Dennet [10] and McCarthy [19]. Dennet proposes what he calls the *intentional stance*, where systems are ascribed mental qualities such as intentions and beliefs. According to him, the important aspect is not whether those systems are really intentional, but if they can be coherently described as if such. McCarthy, in his turn, makes a distinction between the legitimacy and the usefulness of ascribing mental qualities to systems. For him, it is legitimate to ascribe mental states such as beliefs, intentions, abilities and so when there is a correspondence to their common-sense counterparts. It is useful when it helps to understand and control the structure of the system, along with its past and future behaviour.

When a mentalistic approach is adopted, the central problem becomes to choose the mental states that should be used to characterise an agent. Searle [28] divided the mental states or attitudes in two major categories: *information attitudes* and *pro-active attitudes.* Information attitudes relate to the information an agent has about the world where it leaves, precisely *knowledge* and *belief.* Pro-attitudes are those that, in some way, guide the agent's behaviour. Exactly what combination of such mental states is adequate to describe an agent is no consensus. In general, it seems reasonable to require, at least, one attitude of each kind. Davidson [9] argues that desires and beliefs are the two basic mental states, and that all the other ones could be reduced to them. Bratman [3], in his turn, states that intentions, which seem not to be reducible to desires and beliefs , must be considered. These three mental states are the ones used in **B**eliefs-**D**esires-**I**ntentions (BDI) models of agents. Usually, these notions and their properties are formally defined using logical frameworks that allow us to analyse, to specify and to verify rational agents, like in [5, 29, 22](among others).

However, despite the fact that many systems have been developed based on these models ([26, 6], to mention some), it is a general concern that there is a gap between those powerful BDI logics and practical systems. We believe that the main reason for the existence of this gap is that the logical formalisms used to define the models do not have an operational model that supports them. [21, 22]. In [23], the authors propose a BDI model that, besides being a formal model of agents, is also suitable to be used to implement agents. In that model, the notions of belief, desires and intentions are defined using a logic formalism that is both well defined and computational. This allows us both to formally define agents in terms of their mental states and to use this formal model to verify how these agents execute. Since the model is executable, it allows us to model the tutor and students, and to verify them executing a mental states choreography.

Previous work using this approach just considered two agents and a simple testbed with a specific teaching/learning situation using *learning by example* strategy. Just one Tutor and one student were modelled and observed. The results were very interesting because the dynamics of the interaction could be traced and observed. Notice that the tendency of educational environments is to consider more than one student working in a co-operative way based on the change of the traditional educational paradigm, where the teacher is the focus, to another one, where the student is the focus (learn to learn). We strongly believe in the usefulness of this paradigm shift and of mental states metaphor to model agents (and, in particular, ITS). Therefore, our purpose in this paper is to join our previous works about the mental state choreography [17] and apply it to a more complex teaching/learning situation where more than two agents appear. We take advantage of the formal and executable aspects of our agent model to build those choreographies [21, 22, 23]. This is an intermediate step towards our long-term goal of achieving a real society formed by the tutor and the students. At the present stage, we can verify how co-operation between the two students evolves. But this is only simulated, as it happens out of system control.

The approach we take is to create an environment using agent's techniques, with a MAS architecture and mental states modelling integrated with a game-like interface, using simulation with the characteristics of problem solving. The purpose of the problem solving is providing a tool for students and teachers to discover the way s/he thinks and how to integrate conceptual and strategy knowledge to a specific situation (problem). Because we have computational limits we are able to model only problems, and not concepts. We designed a simulation environment not an executable environment. We intend to simulate an agent rational behaviour not a human behaviour. Although this imposes us a pedagogical restriction, it is acceptable at the present stage of our work.

The choice of Environment Education for the domain area is based on the importance to educate children to preserve the nature and learn to control pollution and preserve the planet. This is a very complex open problem with many pedagogical, psychological and modelling implications. As we said earlier we do not have a general educational and psychological theory to support strong student models based on mental states. Therefore, we can use the results obtained until now and try to reproduce them in a computer to trace the dynamic of the process in order to improve the interaction between artificial tutor and students. We would like to trace the process implicit in selecting a specific strategy.

This paper is organised as follows: in section 2, we describe the logical formalism and the agent model we use to define our tutor system. In section 3, we describe the elements that compose the MCOE system. In section 4, we present the tutor architecture we propose and that is used to build the MCOE system. In section 5, we describe a dialogue that shows the interaction of the students and the tutor, and how the mental states evolve. In section 6, we show how to formalise these mental states and how the underlying agent model may be used to execute the ITS model. Finally in section 7, we discuss some of the results we obtained, draw some conclusions and point to some future work.

2 BDI Model using Extended Logic Programming

There are, at least, two major approaches that may be used to overcome the limitations of BDI models. One is to extend existing BDI logics with appropriate operational models so that agent theories become computational [27. 2]. The other approach is to define BDI models using a suitable logical formalism that is both powerful enough to represent mental states and that has operational procedures that allow us to use the logic as a knowledge representation formalism, when building the agent [7; 22]. This is the path followed in this work.

2.1 The Logical Formalism

The formalism we are using is *logic programming extended with explicit negation* (ELP) with the ***W**ell-**F**ounded **S**emantics e**X**tended for explicit negation* (WFSX). ELP with WFSX (simply ELP, from now on) extends normal logic programs with a second negation named *explicit*, in addition to the usual *negation as failure* of normal logic programs, which is called *implicit negation* in the ELP context.. This extension allows us to explicitly represent negative information (like a belief that a property *P* does not hold, or an intention that a property *P* should not hold) and increases the expressive power of the language. When we introduce negative information, we may have to deal with contradictory programs [1]. The ELP framework, besides providing the computational proof procedure for theories expressed in its language, also provides a mechanism to determine how to minimally change a logic program in order to remove contradictions. Our model benefits from these features provided by the logical formalism. As it is usually done, we focus on the formal definition of mental states and on how the agent behaves, given such mental states. But, contrasting with former approaches, our model is not only an agent specification, but it may also be executed in order to verify the actual agent behaviour, as well as it may be used as reasoning mechanism by actual agents. We depart from Bratman's analysis [3], where he states that, along with desires and beliefs, intentions is a fundamental mental state. Therefore, initially we define these three mental states and the static relations between them, namely constraints on consistency among those mental states. Afterwards, we advance with the definition of dynamic aspects of mental states, namely how the agent chooses its intentions, and when and how it revises its intentions.

An extended logic program (ELP) is a set of rules $H \leftarrow B_1, \mathrm{K}, B_n, not\ C_1, \mathrm{K}, not\ C_m \quad (\mathrm{m,n} \geq 0)$ where $H, B_1, \mathrm{K}, B_n, C_1, \mathrm{K}, C_m$ are objective literals. An objective literal is either an atom *A* or its explicit negation $\neg A$. The symbol *not* stands for negation by default and *not L* is a default literal. Literals are either objective or default literals. Also, $\neg\neg L \equiv L$. The language also allows for integrity constraints of the form $A \Leftarrow B_1, \mathrm{K}\ B_n, not\ C_1, \mathrm{K}, not\ C_m \ (\mathrm{m,n} \geq 0)$ where $A, B_1, \mathrm{K}, B_n, C_1, \mathrm{K}, C_n$ are objective literals, stating that *A* should hold if its body $B_1, \ldots, B_n, C_1, \ldots, C_n$ holds. Particularly, when $A = \perp$, where $\perp$ stands for *contradiction* , it means that a contradiction is raised when the constraint body holds[1].

When we reason about pro-attitudes like desires and intentions, we need to deal with properties that should hold at an instant of time and with actions that should be executed at a certain time. Therefore, in order to represent them and to reason about them, we need to have a logical formalism that deals with actions and time. In this work, we use a modified version of the Event Calculus (EC) proposed in [20]. The predicate *holds_at(P,T),* defining that property *P* is true at a time *T* is:

$$
\begin{array}{lcl}
\neg holds\_AT(P,T) & \leftarrow & not\ holds\_at(P,T). \\
holds\_at(P,T) & \leftarrow & initially(P),\ persists(0,P,T). \\
holds\_at(P,T) & \leftarrow & happens(E,Te), initiates(E,P), Te < T,\ persists(Te,P,T). \\
holds\_at(P,T) & \leftarrow & senses(P,Ts), Ts < T,\ persists(Ts,P,T). \\
persists(Te,P,T) & \leftarrow & not\ clipped(Te,P,T). \\
clipped(Te,P,T) & \leftarrow & happens(Ie,TIe), terminates(Ie,P),\ not\ out(TIe,Te,T). \\
out(TIe,Te,T) & \leftarrow & T \leq TIe. \\
out(TIe,Te,T) & \leftarrow & TIe < Te.
\end{array}
$$

The predicate *happens(E,T)* means that event *E* occurred at time *T*; *initiates(E,P)* means that event *E* initiates property *P* at the time event *E* occurs; *terminates(E,P)* means that event *E* terminates *P*; *persists(Te,P,T)* means that *P* persists since *Te* until *T* (at least). We assume there is a special time variable *Now* that represents the present time. Note that a property *P* is true at a time *T* (*holds_at(P,T)*) if there is a previous event that initiates *P* and if *P* persists until *T*. *P* persists until *T* if it can not be proved by default the existence of another event that terminates *P* before the time *T*.

These are the original provisions of EC for the *holds_at(P,T)* predicate. It allows us to reason about the future, by hypothetically assuming a sequence of actions represented by *happens(E,T)* and *act(E,A)* predicates and verifying which properties would hold. It also allows us to reason about the past. In order to know if a given property *P* holds at time *T*, the EC checks what properties remain valid after the execution of the actions that happened before *T*. But it assumes that properties change only as a consequence of the actions performed by the agent. This is not a reasonable assumption, as we would like to allow other agents to act, as well as to allow for actions that are not noticed by the agent. This is the role of *sense(P,T).* It means that property *P* is perceived by the agent at time *T*.

ELP with the EC provides the elements that are needed to model mental states. Nevertheless, the use of a higher level language would allow us to have a simpler and clearer description of actions and its effects. Therefore, we cast the ELP language and the EC primitives in a simpler syntax that is an extension to the *A* language proposed by Quaresma et all. [25]. The sentences are:

- *A causes F if* $P_1,\ldots,P_n$ – action *A* causes property *F* if propositions P_i hold;
- *F after A if* $P_1,\ldots,P_n$ – property *F* holds after action *A* if propositions P_i hold;
- *A occurs_in E* – action *A* occurs when event *E* occurs;
- *E preceeds E'* – event *E* occurs before event *E'*.

The language still provides means to reference beliefs that should hold in the future or that should have held in the past. In this paper, we make use only of the operator *next(P)* stating that property *P* should hold at the next instant of time. Now that we have the logical formalism set, we are able to define the BDI model.

2.2. The Agent Model

We depart from Bratman's analysis about intentions, its role in rational reasoning and how it relates to beliefs and desires. According to Bratman [3], since agents are assumed to be resource-bounded, they cannot continuously evaluate their competing beliefs and desires in order to act rationally. After some reasoning, agents have to commit to some set of choices. It is this choice followed by a commitment that characterises the intentions. Our model does not define a complete agent, but only the cognitive structure that is part of the agent model.

[1] For a complete formal definition of the language and its semantics, see [1].

An *agent cognitive structure* is a tuple $\langle B, D, I, T \rangle$ where *B* is the set of agent's beliefs, *D* is the set of agent's desire, *I* is the set of agent's intentions and *T* is the set of time axioms, as defined above. The *desires* of the agent is a set of sentences *DES(Ag,P,Atr) if Body*, where *Ag* is an agent identification, *P* is a property and *Atr* is a list of attributes. Desires are related to the state of affairs the agent eventually wants to bring about. But desires, in the sense usually presented, does not necessarily drive the agent to act. That is, the fact of an agent having a desire does not mean it will act to satisfy it. It means, instead, that before such an agent decides what to do, it will be engaged in a reasoning process, confronting its desires (the state of affairs it wants to bring about) with its beliefs (the current circumstances and constraints the world imposes). The agent will choose those desires that are possible according to some criteria.

Beliefs constitute the agent's information attitude. They represent the information agents have about the environment and about themselves. The set *B* contains sentences describing the problem domain using ELP. An agent *A* believes a property *P* holds at a time *T* if, from *B* and *T*, the agent can deduce *BEL (Ag,P)* for the time *T*. We assume that the agent continuously updates its beliefs to reflect changes it detects in the environment. We assume that, whenever a new belief is added to the beliefs set, consistency is maintained.

Intentions are characterised by a *choice* of a state of affairs to achieve, and a *commitment* to this choice. Thus, intentions are viewed as a compromise the agent assumes with a specific possible future. This means that, differently from desires, an intention may not be contradictory with other intentions, as it would not be rational for an agent to act in order to achieve incompatible states. Also, intentions should be supported by the agent's beliefs. That is, it would not be rational for an agent to intend something it does not believe is possible. Once an intention is adopted, the agent will pursue that intention, planning actions to accomplish it, re-planning when a failure occurs, and so. Agents must also adopt these actions, as means that are used to achieve intentions, as intentions.

The definition of intentions enforces its rationality constraints: an agent should not intend something at a time that has already past; an agent should not intend something it believes is already satisfied or that will be satisfied with no efforts by the agent; an agent only intends something it believes is possible to be achieved, i.e., if it believes there is a course of actions that leads to the intended state of affairs. When designing an agent, we specify only the agent's beliefs and desires. It is up to the agent to choose its intentions appropriately from its desires. Those rationality constraints must also be guaranteed during this selection process [22].

Agents choose their intentions from two different sources: from its desires and as a refinement from other intentions. By definition, there are no constraints on the agent's desires. Therefore, an agent may have conflicting desires, i.e., desires that are not jointly achievable. Intentions, on the other hand, are restricted by rationality constraints (as shown above). Thus, agents must select only those desires that respect to those constraints. First, it is necessary to determine those subsets of the desires that are relevant according to the current beliefs of the agent. Afterwards, it is necessary to determine desires that are jointly achievable. In general, there may be more than one subset of the relevant desires that are jointly achievable. Therefore, we should somehow indicate which of these subsets are preferred to be adopted as intentions. This is done through a preference relation defined on the attributes of desires. According to the theory defined in [21, 22], the agent should prefer to satisfy first the most important desires. Additionally to preferring the most important ones, the agent adopts as much desires as it can. The selection is made combining the different forms of non-monotonic reasoning provided by the logical formalism.

Once the agent adopts its intentions, it will start planning to achieve those intentions. During planning, the agent will form intentions that are relative to pre-existing ones. That is, they "refine" their existing intentions. This can be done in various ways, for instance, a plan that includes an action that is not directly executable can be elaborated by specifying particular way of carrying out that action; a plan that includes a set of actions can be elaborated by imposing a temporal order on that set. Since the agent commits to the adopted intentions, these previously adopted intentions constrain the adoption of new ones. That is, during the elaboration of plans, a potential new intention is only adopted if it is not contradictory with the existing intentions and with beliefs.

The next step is to define *when* the agent should perform all this reasoning about intentions. We argue that it is not enough to state that an agent should revise its intentions when it believes a certain condition holds, like to believe that an intention has been satisfied or that it is no longer possible to satisfy it, as this suggests that the agent needs to verify its beliefs constantly. Instead, we take the stance that it is necessary to define, along with those conditions, a mechanism that triggers the reasoning process without imposing a significant additional burden on the agent. Our approach is to define those conditions that make the agent start reasoning about intentions as constraints over its beliefs. Recall that we assume that an agent constantly has to maintain its beliefs consistent, whenever new facts are incorporated. Whenever the agent revises its beliefs and one of the conditions for revising intentions hold, a contradiction is raised. The intention revision process is triggered when one of these constraints is violated.

3 The MCOE system

The ‘MCOE’ is a pedagogical game that simulates a lake with plants, and different types of fish. These elements are typical of our real world (in this case, the river in our home city and its surroundings).

The conception of the ‘MCOE’ was based on an ITS architecture [15, 16; 17]. This architecture is composed of a hybrid society of agents who work to achieve a common goal: to fight against the pollution resulting from foreign elements (pollutants) and to maintain the equilibrium. We have reactive agents (bottom of the lake, micro-organisms - plankton, water, plants and three types of fish) and cognitive agents (the Tutoring agent, the ecologist, and students represented by characters).

We designed the system to be played by two students using different machines. They can be in the same physical place or not. The first student chooses a character to play using four options: Mother Nature, Mayor, Citizen, and Tourist. After that, the second student can choose one of the three remaining characters. The system defines the game configuration (foreign elements that will cause pollution) by a sorting process using a random function. The students first see a lake in equilibrium, which soon begins to show the action of the polluters. Their challenge is to maintain the equilibrium and fight the action of these foreign elements for a period of ten minutes[2].

The number of fish swimming in the lake, plants and micro-organisms, the water transparency, and the appearance of the bottom of the lake indicate the situation. There is a visual gauge (the Ecometer) that helps the user to observe the conditions of the lake. A balanced situation is shown when the gauge is full and green. As the equilibrium is lost, the colour changes gradually to yellow, finally changing to red when a dangerous situation is reached.

The system equilibrium is based on the energy level, the occasional predatory actions that can happen in the environment and the number of elements that remains in the system after the pollutants impact. The fish, plankton, and plants reproduce using a function based also on the energy level. If an agent dies (zero energy) it is removed from the system.

4 The Tutor architecture

The elements of the Tutoring agent architecture are:

- The Tutor mental states. It is a BDI model that represents the desires and beliefs of the tutor. The knowledge about the students (beliefs about the student’s beliefs and intentions) is in the tutor’s beliefs;
- Knowledge about what tool to use with specific foreign elements. This is also part of the tutor’s beliefs, in the formal model;
- Knowledge about energy levels of scenario elements, a part of the tutor’s beliefs, as well;
- Rules to help it to decide what action should be taken according to changes of the students’ mental states. This is also part of the tutor’s beliefs;
- Sensors for receiving data about the environment.

The kernel of the Tutor architecture is the Behavioural Decision centre (figure 1).

[2] The choice of ten minutes is because the game was designed to be played during classroom time, in a context of a real class situation, and a pre-defined period of time is necessary for the teacher to make her/his class plan.

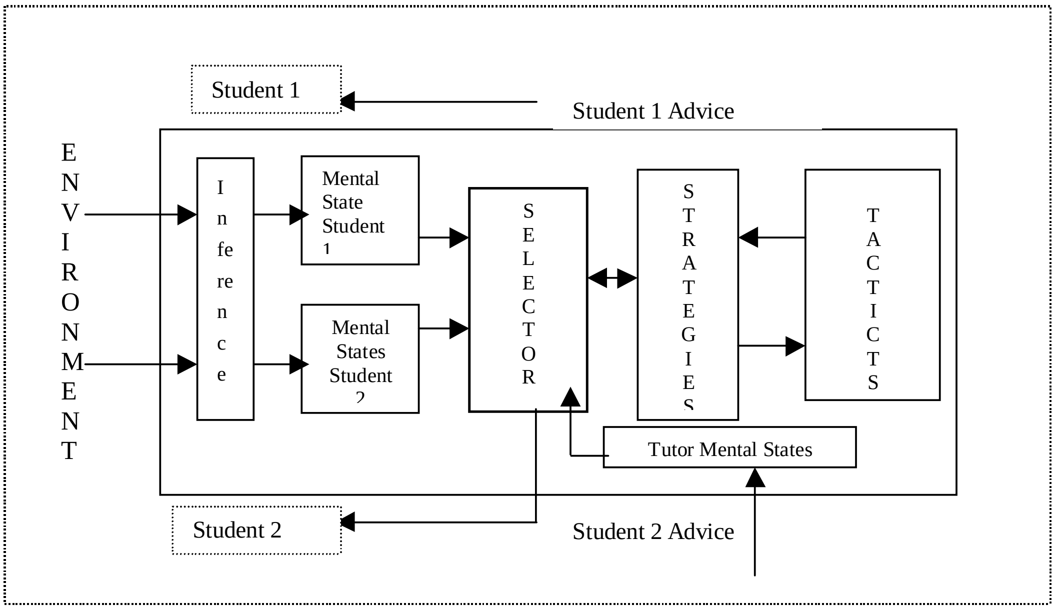

Figure 1: Behavioural Decision Centre

The tutor receives information about each student model. The information is composed by the mental states of each student, after they perform an action, and the environment energy levels. The tutor uses it to select a specific strategy with the associated tactics, according to the selector module. The Tutor itself just observes the interactions, it does not act in the environment. Its only intervention occurs when it gives advices to the student. The tutor doesn't know what will happen along the game, i.e., he doesn't know what foreign elements will appear in the scenery, neither the amount of foreign elements. The students' mental states and their actions (tool activation) are perception information the tutor receives from the environment. The system works as if the tutor hears and sees what happens in the environment. Those would be its sensorial capabilities.

The set of the students' mental states is obtained through a module of inference of attitudes, as presented in Quaresma[25]. The dialogues are seen as composed by a sequence of speech acts. According to [28], to speak is to act. When agents speak they accomplish speech acts. The speech acts, according to Cohen and Perrault [4], are actions whose effects affect both the originator and the receiver. Quaresma's work offers a formal and computational model of how to infer the mental states of the participants in a dialogue. Although this is not part of the implementation of the MCOE system, it is considered in the BDC project as an explanation of how to obtain the students' mental states.

5 A Dialogue Choreography

We analyse, now, how the dialogue preceeds in a situation the simulator can create that. The interface system not allows dialogue in natural language. Due to limitation of space, we selected a small dialogue that is complex enough to show how the logical framework is used.

The system places a lot of pollution in the screen and it creates a complex situation. The foreign elements are 2 dredges, 2 Jet-Ski, 2 industrial sewer, chemical pollution and, 2 gas station. After 30 seconds the game begins. The foreign elements are placed in the scenery in a varied way, and not at the same time. This runs in order to put difficulties along the game and to promote cumulative effects through interference in the scenery.

Student1 chooses to be the mayor and student2 chooses to be the Mother Nature. Both mayor and Mother Nature are characters with plenty of resources to combat the pollution. In the beginning the system activate the dredge and the gas station. The energy level decreases: Plants –40%, Micro-organism –50%, Bottom –

50%, Fish –20%, Water –20%. At this time, the Ecometer shows the energy level and the colour of some scenery elements changes.

The tutor has one and only purpose: to aid the student to maintain an adequate energy level in the environment. It believes that it may aid by sending the students messages. The contents of these messages will depend on the strategy adopted at the moment by the tutor. These desire and belief are modelled as follows

DES(tutor,verify_knowledge(Student)) if
BEL(tutor,energy(ecometer,Ee)),
Ee<100.

Since this is his only desire, it is the only candidate to intention. It will be adopted as intention if the tutor believes there is a sequence of actions that it can executed. But, at this moment, it depends on knowledge about the students' mental states in order to decide what to do. Since it does not have (yet) this knowledge, it waits. But, it inserts in its beliefs triggers that will make it reconsider its options when interaction occurs. For instance, the tutor has the following belief

BEL(tutor, send_message(1)) if BEL(tutor,INT_THAT(Student, receive_aid),
BEL(tutor, energy(ecometer, Ee)), Ee > 70,
BEL(tutor, time(T)),T >8. *(1)*

meaning that if tutor believes the student expects some help, it believes the energy level is bellow 70 and the game time is above eight minutes. So, the help that it can offer is to advise the student to pay attention to the remaining game time, and select the strong tool to fight against the pollution. This belief would place as triggers

$\perp\leftarrow$ *BEL(tutor,INT_THAT(Student, receive_aid),*
BEL(tutor, energy(ecometer, Ee)),Ee > 70, BEL(tutor, time(T)),T >8 *(2)*

i.e., when the pre-conditions for his action of sending a message are satisfied, his is in condition of aiding the student. The agent models then transform the desire in an intention a satisfies it sending the message.

Once the tutor has this intention, it uses its beliefs about the environment (tool, energy level, foreign elements, scenery elements), about how to fight against pollution in the environment (different strategies on the use of the tools to eliminate pollution), about how to advise the students in order to help the students control the environment and so on.

****

(Time Point 1)

Student1: Pollution began and I will apply tool *fine*.

Mental states show the expectation "the energy level will grow up" and "student2 will collaborate".

Student2: I will *accelerate* the degradation.

Mental states show the expectation "the energy level will grow up".

Tutor: It verifies that the students didn't change their believes and the level of energy increased. It stays observing and it doesn't send any message.

None of the triggers set by the tutor are activated by this interaction. Therefore, it does not adopt any intention and remains waiting.

****

(Time Point 2)

The energy level increased but it is not 100%.

Student1: Student1 emits a message for Student2 asking what h/she intends to do.

S/he expects "to receive help from Student2" and shows the belief "Student2 will cooperate".

Student2: Student2 replies with "you should use a tool that increases as much as possible the energy level".

It adopts the intention "cooperate with Student1" and the expectation "the energy level will grow up".

Tutor: The students are co-operating and building a common strategy. As the level of energy is rising and the students maintain its basic believes and desires, the Tutor awaits. All elements have with 100% energy.

****

(Time Point 3)

Student1: Look this! Everything is going well!

Student2: Yes it is working!

Tutor: Awaits.

****

(Time Point 4)

Then appear the polluters: 1 dredge, 1 Jet-Ski and chemical polluters. Energy level at this time: Plants –40%, Micro-organism –40%, Bottom –50%, Fish –30%, Water –20%. The students notices the energy dropped in several places. S/he looks at the colour of the things in the lake.

Student1: I will *intensify* the sunshine because it will place more energy and very fast.

Mental states show the expectation "the energy level will grow up" and the belief "powerful tools increase the energy level soon".

Student2: I will use *removes the license* for everything to increase the energy level fast.

Mental states show the expectation "the energy level will grow up" and the belief "powerful tools increase the energy level soon".

Tutor: Verifies that the energy level grew and the students are not using the rule to control the number of times that the tool can be used.

The students' mental states do not shows the belief "number of tools should be controlled". The tutor emits a message of alert to the two students in this sense.

The Tutor will perform this action when the amount of some of the tools is one. Its beliefs have "number of tools should be controlled" and activates the desire "control number of tools".

In this step, the tutor just analysed the interaction. In fact, according to the agent model, the tutor just expanded its beliefs. But, these new beliefs do not activate any of the triggers. This means that none of the tutor's desires are relevant. In fact, it is not considered rational to act in order to achieve an intentions if one beliefs the intention will be satisfied without any intervention. In our case, if the tutor thinks the students is performing well, it would not be rational to give him any advice.

This is the usual behaviour when our agent model is used. The agent just maintains its set of beliefs consistent. When the change in the belief set activates some trigger, the agent will try to adopt intentions, build plans and so on. In this particular agent, the tutor, the possibility to build plans is not used. In fact, we argue that the tutor should not have a long term plan. Instead, it should have very short term plan that takes into account the history of interactions and, particularly, the last one. It is a cognitive tutor that presents a somewhat reactive behaviour. It is not a simple application of reaction rules because we take into account that history of the interaction and the expectation about the next interaction. Besides being interesting from the computational point of view [22], it also respects the pedagogical approach being used, namely the construtivist approach [14].

****

(Time Point 5)

At this time, more foreign elements appear: 1 gas station and 1 industrial sewer. Total of energy after the new polluters: Plants –65%, Personal computer –35%, Bottom –65%, Fish –25%, Water –80%.

Student1: The lake is dying. I don't have more the tool *intensify* and the strongest I have is *accelerates*. I will play with this one.

Beliefs have "strong tools should be used to grow up the energy fast", "number of tool equal zero then tool unavailable", "should control the number of times that can use a tool" and "should control the time of the game".

The desire "cooperate with student" becomes an intention. There are the expectations: "Student2 will cooperate" and "the energy level will grow up".

Total of energy after this intervention: Plants –50%, Personal computer –20%, Bottom –50%, Fish –10%, Water –65%.

Student2: I will use *prohibition*. The energy level will increase.

Mental states show the beliefs "strong tools should be used to grow up the energy fast", "number of tool equal zero then tool unavailable", "should control the number of times that can use a tool" and, "should control the time of the game". The desire "cooperate with student" is selected as intention. There is the expectation "the energy level will grow up".

Total of energy in the time after this intervention: Plants –35%, Personal computer –10%, Bottom –40%, Fish –5%, Water –55%.

Tutor: Verifies that the two students are co-operating. When co-operation[3] exists the Tutor tends to reinforce their behaviour. In fact, the tutor should have a belief describing what indicates there is co-operation between students. For instance

BEL(tutor, student_co-operation) if BEL(tutor, student_exchange_messages).
BEL(tutor, student_exchange_messages) if BEL(tutor, number_message (NM)),NM > 0.

At this point, the triggers **(2)** connected with belief **(1)** set at by the at the beginning are activated. Therefore, the tutor adopts an intention (to aid the students) that is satisfied by sending a message, in this case a reinforcement message.

The dialogue proceeds in this way until the game is over. The students controlled the pollution and the Tutor aided them.

At this point, the tutor produces a description of the student's interactions. This description, with the evolution of the mental states, allows the teacher to verify how well the students understand the problem domain. The choice of inadequate tools or of low combat power in a critical situation indicates that the student still misunderstands the problem and needs a more direct orientation.

6 Conclusions and Future Work

The use of the mental states for modelling the students and tutor behaviour in ITS is based on the notion that this is the usual way to describe interaction among people. It is a not only a metaphor used to build a model. We try to understand what is happening in the student's mind when s/he is thinking. According to McCarthy [19], the use of mental sates is *useful* because it helps to organise the dynamics of the process and *necessary* to get the insights for this mechanism.

Initially the system MCOE was designed to use the set of mental states according to the SEM agent architecture presented by Corrêa [6]. However, this model does not have an operational environment available at this moment. This made us reconsider the initial proposal and we are now using the Móra, Viccari, Pereira and Coelho [21, 23] approach. They have a formal and computational model to treat the mental states that is based on the same set of mental states as Corrêa. Therefore, we have the same theoretical foundations, we can treat the past, present and next interaction and we profit from them proximity between the description languages used by the two systems.

Although we show here a very simple and small dialogue, in order to identify better all the necessary mental states that should compose the choreography, other dialogues have been created. These dialogues represent two students with great co-operation and content domain, two students with medium co-operation and medium domain content and, two students with low co-operation level and they don't dominate the content. These three stereotypes seems to be enough to our propose. The full formalisation of these more complete dialogues is yet to be done. Nonetheless, the results we have until now are by themselves stimulating and they allow us to trace the agents' behaviour at the desired abstraction level. It also allows us to refine the algorithms and to test the set of rules that regulate the system.

7 References

[1] Alferes, J.J.; Pereira, L.M. Reasoning with Logic Programming. **Lecture Notes in Artificial Intelligence Series.** Springer-Verlag, Berlin, DE., 1996.

[2] Bell, J.; Huang, Z. **Dynamic Goal Hierarchies**. In: Proceedings of the Second Workshop on Practical Reasoning and rationality. AISB Workshop Serires, 1997.

[3] Bratman, M.E.. What is intention? In P.R. Cohen, J.L. Morgan, and M. Pollack, editors, **Intentions in Communication.** The MIT Press, Cambridge, MA, 1990.

[4] Cohen, P.R.; Perrault, R. **Elements of a Plan-Based Theory of Speech Acts.** Cognitive Science, 3:177-212, 1979.

[5] Cohen, P.R; Levesque, H.J.. **Intention is choice with commitment**. Artificial Intelligence, 42,213--261, 1990.

[3] Co-operation is different from collaboration. *Collaboration* is to share information without modify the received information. *Co-operation* is to share information and interfere in the received information. To act together to build something together. All the co-operation involves the collaboration.

[6] CORRÊA,M. **A Arquitetura de Diálogos entre Agentes Cognitivos Distribuídos**. Rio de Janeiro: Universidade Federal do Rio de Janeiro, Pós-Graduação em Engenharia, Rio de Janeiro, 1994. (Ph.D. Thesis -Portuguese)

[7] CORRÊA, M.; MENDES, S. A Computational Approach to Situation Theory Based on Logic Programming to design Cognitive Agents. In: Brazilian Symposium on Artificial Intelligence, 12., 1995, Campinas. **Proceedings...** Campinas: Springer Verlag, 1995.

[8] CORRÊA, M; Frankel, E. A cognitive Approach to Body Psychotherapy. **The Journal of Biosynthesis**. [S.l.]: Abbotisbury Publications, 26(1), 1995.

[9] Davidson, D. **Essays on actions and events**. Oxford University Press, New York, NY, 1980.

[10] Dennet, D.C. **The intentional stance**. The MIT Press, Cambridge, MA, 1987.

[11] Devlin, K. **Logic and Information**. Cambridge: Cambridge Press, 1991.

[12] Frasson, C.; Mengelle,T.; Aimeur,E. Using Pedagogical Agents in a Multi-Strategic Intelligent Tutoring System. In: AI-ED97: Eighth World Conference on Artificial Intelligence in Education - Workshop V: Pedagogical Agents, 8., 1997. **Proceedings...** Kobe: Japan, 1997.

[13] Gagné, R.M. **The conditions of Learning**. Montreal: HRW, 1984.

[14] Giraffa, L.M.M. **Seleção e adoção de estratégias de ensino em sistemas tutores inteligentes.** Porto Alegre: CPGCC/UFRGS, 1997. (Exame de Qualificação – EQ10). (Portuguese).

[15] Giraffa, L.M.M; Viccari, R.M.; Self,J. Improving tutoring activities using a Multi-Agents system Architecture. Twenty-ninth SIGCSE Technical Symposium on Computer Science Education, 29,1998. **Proceedings...** Atlanta: Georgia,1998. (Student Research Contest -http://www1.acm.org/spost/gabstract.txt)

[15a] Giraffa, L.M.M; Viccari, R.M.; Self.J. Multi-Agent based pedagogical games. ITS'98- Fourth International Conference on Intelligent Tutoring Systems. **Proceedings…** San Antonio, Texas, 1998. (Poster Section)

[15b] Giraffa, L.M.M; Viccari, R.M. Tutor behaviour in a multi-agent ITS guided through mental states activities. ITS'98- Fourth International Conference on Intelligent Tutoring Systems. Workshop Pedagogical Agents. **Proceedings…** San Antonio, Texas, 1998.

[18] Khuwaja.R; Desmarais, M.; Cheng, R. Intelligent Guide: Combining User Knowledge Assessment with pedagogical Guidance. International Conference on Intelligent Tutoring Systems - ITS'96, 3., 1996. **Proceedings ...** Berlin: Springer-Verlag Verlag: 1996.

[19] McCarthy, J. Ascribing Mental Qualities to Machines. Menlo Park: Stanford Research Center, 1978. (Technical Report)

[20] Móra, M.C.; Lopes, J.G.; Coelho, H.. Modelling intentions with extended logic programming. In J.Wainer and A.Carvalho, editors, Advances in artificial intelligence: **Proceedings** of the 12th Brazilian Symposium on Artificial Intelligence, Berlin, Germany, 1995. SBC, Springer-Verlag. Lecture Notes in Artificial Intelligence (LNAI 991).

[21] Móra, M.C.; Lopes, J.G.; Coelho, J.G.; Viccari, R. Modelling dynamic aspects of intentions. In E.Costa, editor, 8th Portuguese Conference on Artificial Intelligence. Springer-Verlag, **Proceedings...**, 1997.

[22] Móra, M.C.; Lopes, J.G.; Coelho, J.G.; Viccari, R. Modelling agents with extended logic programme. In International Workshop on Engineering of Intelligent Systems. ICSC co., 1998.

[23] Móra, M.C.; Lopes, J.G.; Coelho, J.G.; Viccari, R. BDI models and systems: Reducing the gap. In: Agents Theory, Architecture and Languages Workshop. **Lecture Notes on Artificial Intelligence**, Springer-Verlag, 1998.

[24] Moussale, N.M.; Viccari, R.M.; Correa, M. Intelligent Tutoring Systems Modelled through the Mental States. In: **Lectures Notes on Artificial Intelligence** - SBIA'96. Borges,D. and Kaestner,C.(Eds.). Berlin: Springer Verlag, 1996.

[25] Quaresma, P. **Inferência de atitudes em diálogos.** Lisboa: FCT/UNL, 1997. (Ph.D. Thesis -Portuguese)

[26] Rao,A.; Georgeff, M. **Modelling Rational Agents with a BDI Architecture**. In: Proceedings of the Knowledge Representation and Reasoning'91 (KR\&R'91). Morgan Kauffman Publishers, 1991.

[27] Rao, A. **Agentspeak(1): BDI agents speak out in a logical computable language**. In: Proceedings of the European Workshop on Modelling Autonomous Agents and Multi-Agents Worlds 1996 (MAAMAW'96). Springer-Verlag, 1996.

[28] Searle, J.. What is an intentional state? In H.Dreyfuss and H.Hall, editors, Husserl, **Intentionality and Cognitive Science**, (42), 1984.

[29] Shoham, Y. **Agent-Oriented Programming**. Artificial Intelligence, 60:51-92, 1993.

[30] Winograd, T.; Flores, F. **Understanding Computers and Cognition**: a new foundation for design. New Jersey: Ablex Publishing, 1984.

10 AIED99 – ”Towards a New Computational Model to Build a Tutor”

Este artigo, apresentado no AIED99 e publicado como ”poster”, mostra como a arquitetura multiagente utilizada para construir o tutor, com o agente cognitivo definido através do modelo formal e executável, pode servir de modelo padrão para construção de qualquer tutor que siga a abordagem adotada pelo MCOE. Uma vez formalizado o modelo, realizados os experimentos com a arquitetura subjacente e com a ferramenta construída a partir do modelo, demosntra-se agora a generalidade desta arquitetura para uma classe de aplicações, nomeadamente os Tutores Inteligentes.

Towards a New a Computational Model to build a Tutor

Lucia M. M. Giraffa
giraffa@inf.pucrs.br
FACIN/PUCRS
CPGCC/UFRGS

Michael da C. Móra
michael@inf.pucrs.br
CPGCC/UFRGS
FACIN/PUCRS

Rosa M. Viccari
rosa@inf.ufrgs.br
CPGCC/UFRGS

Faculdade de Informática – PUCRS
Av. Ipiranga 6681 – prédio 16 – Porto Alegre – RS – Brazil 90610-900

Curso de Pós-Graduação em Ciência da Computação – UFRGS
Av. Bento Gonçalves 9500 – Bloco IV – Porto Alegre – RS – Brazil

Abstract

In this paper, we show how a Multi-Agent Systems (MAS) approach may be used to build an Inteligent Tutoring Systems (ITS). We model the ITS as a society of both reactive and cognitive agents that interact to implement the tutor. The purpose of this new paradigm is to try to overcome the weaknesses of the student models that are usual in current ITS. Our model is based on mental states that are used to describe both the tutor and the students, what, we believe, helps to decrease the gap between what is going on with the student during the learning process and what we are able to represent about that student.

Key words: Intelligent Tutoring Systems, Multi-agents Systems, Agent Modelling, Mental States.

Towards a New a Computational Model to build a Tutor

Abstract

In this paper, we show how a Multi-Agent Systems (MAS) approach may be used to build an Inteligent Tutoring Systems (ITS). We model the ITS as a society of both reactive and cognitive agents that interact to implement the tutor. The purpose of this new paradigm is to try to overcome the weaknesses of the student models that are usual in current ITS. Our model is based on mental states that are used to describe both the tutor and the students, what, we believe, helps to decrease the gap between what is going on with the student during the learning process and what we are able to represent about that student.

Key words: Intelligent Tutoring Systems, Multi-agents Systems. Agent Modelling, Mental States.

1. Introduction

In recent years a new paradigm arose for educational environments: to put more than one student interacting with the same environment under artificial tutor supervision. From the educational viewpoint the idea is not new, but became feasible through the development of hardware and software that allow us to connect people using computer networks and related technologies. The impact of these technologies on educational software research was immediate. Researches have begun to introduce these new possibilities to improve educational environments. Some Intelligent Tutoring Systems (ITS) were built using this approach like Balacheff [4], Colazzo [8], Elliot [14], Frasson and Aimeur [16, 2], and Leroux [23].

The student model remains the weak part of such systems. This situation imposes a great restriction for designing better tutoring systems to aid the student to build his/her own knowledge. We have many reasons to explain this (hardware, software, and techniques to model the student, knowledge representation, and others). However, the strong restriction comes from our imprecise knowledge about mental activities of the students during teaching/learning process. To build an ITS with a good student model we need to better understand what is happening in student's mind during the interaction. We need to understand the process and reproduce it into the machine. Much has been done to understand such process, according to different viewpoints: psychological, educational and from computer science. The work of Correa, Coelho e Viccari [9, 10, 11, 12]; Moussalle [29] and Self's work [1, 32] are examples of such improvements.

The state of the art of the technology and our restricted knowledge about the human information process still do not allow us to build the ideal ITS. However, we have now different tools and new possibilities. The multimedia techniques and agents programming paradigm are some of these new technologies that impact the way to design ITS. Nowadays we can find many experiments that try to use multimedia and/or Multi-Agent Systems (MAS) architectures to improve the traditional ITS. The state of art can be evaluated by the followings works: Andre [3],Costa [13], Frasson [15], Hayes-Roth [20, 21, 22], Lester [24] and Mitsuru [25].

We have been doing many developments in our group in order to explore the possibilities to build ITS using a MAS approach, and specially modelling the tutor using a set of mental sates, such as beliefs, intentions, motives, expectations, obligations and so on. In this paper we present a how we use the notions of agents and mental states to build a tutor. Specifically, we use the set of mental states composed by beliefs, desires, intentions, and expectations considered as future beliefs. Although our model includes both reactive and cogntive agents, we focus here on the aspects of the cognitive agents.

The paper is organised as follows: section 2 describes the agent model and how its concepts are used to build the tutor, section 3 describes the tutor architecture and shows how the interaction with the students occurs and finally, section 4 presents some conclusions and point out some future work.

2. ITS modelling using MAS approach

To build an ITS using a MAS architecture allows us to test another important aspect, namely to design educational systems with hybrid architecture, where reactive and cognitive agents are used. This mixture of different types of agents is important for some applications like the one in our system.

The hybrid architecture present the advantages of reactive environments, namely we can use reactive agents to exhibit the results of cognitive agents actions in a quick and clear way, and the cognitive agents can be designed and implemented using different way of the reactive one. The possibility to connect cognitive environment modelling with some agent's language or in this case with the mentalist approach where we use the computational model proposed by Mora et ali.[26, 27, 28] allow us to propose a new architecture for the tutor that can be adapt in different environment trough a interface that can be build with a low computational cost and low degree of complexity.

Our work has some similar items with those systems mentioned in the introduction that represent the state of the art in ITS using MAS approach, namely:

- the use of graphical and multimedia interface to improve the interaction between the tutor and the student;
- the distribution of knowledge among agents;
- communication among the agents (by messages or by sensors);
- the environment changes dynamically and it is possible see the results of the actions immediately.

We have explored the possibility to build cognitive agents using the mental state choreography to improve the student model and the interactions between the tutor and the students. This new possibility arose with Correa's [9, 10, 11] work. After this work another experiment built by Moussalle [29] showed that it is possible to trace some student's mental activities using this approach. Correa, Coelho e Viccari [12] recently improved the notation and the formalism to observe and test the mental activities that can happen during a teaching/learning session.

Previous work using this approach just considered two agents and a simple testbed with a specific teaching/learning situation using learning by explanation strategy. Just one Tutor and one student were modelled and observed. But results were very interesting because the dynamics of the interaction can be traced and observed. However, the tendency of educational environments is to consider more than one student working in a co-operative way based on WEB applications and the change of the traditional educational paradigm where the teacher was the focus to another one where the student is the focus (learn to learn). We believe in these two approaches (learn to learn, and to have at leats two students interacting with the system at the same time) and decided to join the previous study about the mental state choreography and to apply it to a more complex teaching/learning situation where more than two agents appear. Although the goal of our research group is to have a social environment, where several tutors and systems interact, we do not have achieved this situatuion at this moment. The cognitive aspect in our system has a co-operative behaviour between the two students, but it happens out off system control.

To better observe these behaviours and their consequences we decided to create an environment using agent's techniques, with a MAS architecture and Mental States modelling (Computer Science - Artificial Intelligence viewpoint) and a game-like interface, using simulation with the characteristics of problem solving (by the Educational viewpoint). The purpose of the problem solving is to provide a tool for students and teachers to discover the way s/he thinks and how to integrate conceptual and strategy knowledge to a specific situation (problem). Because we have computational limits we are able to model problems not concepts. Therefore, we designed a simulation environment not an executable environment where we simulated the behaviour of the participants as rational entities. We know it causes a pedagogical restriction in our system, but for the time being there is nothing we can do.

The choice of environment education for the domain area is based on the importance to educate children to preserve the nature and learn to control pollution and preserve the planet. More details about the system modelling and the application (MCOE system[1]) can be found in [16, 17, 18, 19].

3. The Tutor architecture

The proposal here is a tutor architecture for ITS modelling in MAS approach using the mentalist approach. The elements of the Tutoring agent architecture are:

- The tutor mental states. It is a BDI (Belief-Desires-Intentions) model that represents the desires and beliefs of the tutor. The knowledge about the students (beliefs about the student's beliefs and intentions) is in the tutor's beliefs. Intentions are derived by the agent system;
- Knowledge about what tool to use with specific foreign elements. This is also part of the tutor's beliefs, in the formal model;
- Knowledge about energy levels of scenario elements, a part of the tutor's beliefs, as well;
- Rules to help it to decide what action should be taken according to changes of the students' mental states. This is also part of the tutor's beliefs;
- Sensors for receiving data about the environment.

The BDI model is based on Bratman's analysis about intentions, its role in rational reasoning and how it relates beliefs and desires. According to Bratman [5], since agents are assumed to be resource-bounded, they cannot continuously evaluate their competing beliefs and desires in order to act rationally. After some reasoning, agents have to commit to some set of choices. It is this choice, followed by a commitment that characterises the intentions. Our model does not define a complete agent, but only the cognitive structure that is part of the agent model.

An *agent cognitive structure* is a tuple $\langle B, D, I, T \rangle$ where *B* is the set of agent's beliefs, *D* is the set of agent's desire, *I* is the set of agent's intentions and *T* is the set of time axioms, as defined above. The *desires* of the agent is a set of sentences *DES(Ag,P,Atr)*, where *Ag* is an agent identification, *P* is a property and *Atr* is a list of attributes. Desires are related to the state of affairs the agent eventually wants to bring about.

Beliefs constitute the agent's information attitude. They represent the information that agents have about the environment and about themselves. The set *B* contains sentences describing the problem domain using ELP[2]. An agent *A* believes that a property *P* holds at a time *T* if, from *B* and *T*, the agent can deduce *BEL(Ag,P)* for the time *T*. We assume that the agent continuously updates its beliefs to reflect changes that it detects in the environment. We assume that, whenever a new belief is added to the beliefs set, consistency is maintained.

Intentions are characterised by a *choice* of a state of affairs to be achieved, and a *commitment* to this choice. Thus, intentions are seen as a compromise, which the agent assumes with a specific possible future. This means that, differently from desires, an intention may not be contradictory with other intentions, since it would not be rational for an agent to intend something that it believes is impossible. Once an intention is adopted, the agent will pursue that intention, planning actions to accomplish it, re-planning when a failure occurs, and so on. Agents must also adopt these actions to

[1] MCOE is an educational game with characters represented by agents (with reactive and cognitive capabilities) and designed under a construtivist approach. The challenge for the students is control the pollution fighting against foreign elements using special tools.

[2] The formalism we are using is *logic programming extended with explicit negation* (ELP) with the ***Well-Founded Semantics eXtended for explicit negation*** (WFSX). ELP with WFSX (simply ELP) extends normal logic programs with a second negation named *explicit*, in addition to the usual *negation as failure* of normal logic programs, which is called *implicit negation* in the ELP context. More details in [26, 27, 28].

achieve their intentions.

The fundamental idea, according Corrêa [11, 12], is that intention are related to the belief and desire, i.e., beliefs represent the world and how it is possible to transform it. Internally, this is done through non-monotonic reasoning, belief revisions, and construction of strategies to achieve intention through planning.

The kernel of the Tutor architecture is the Behavioural Decision Centre (BDC) (figure 1).

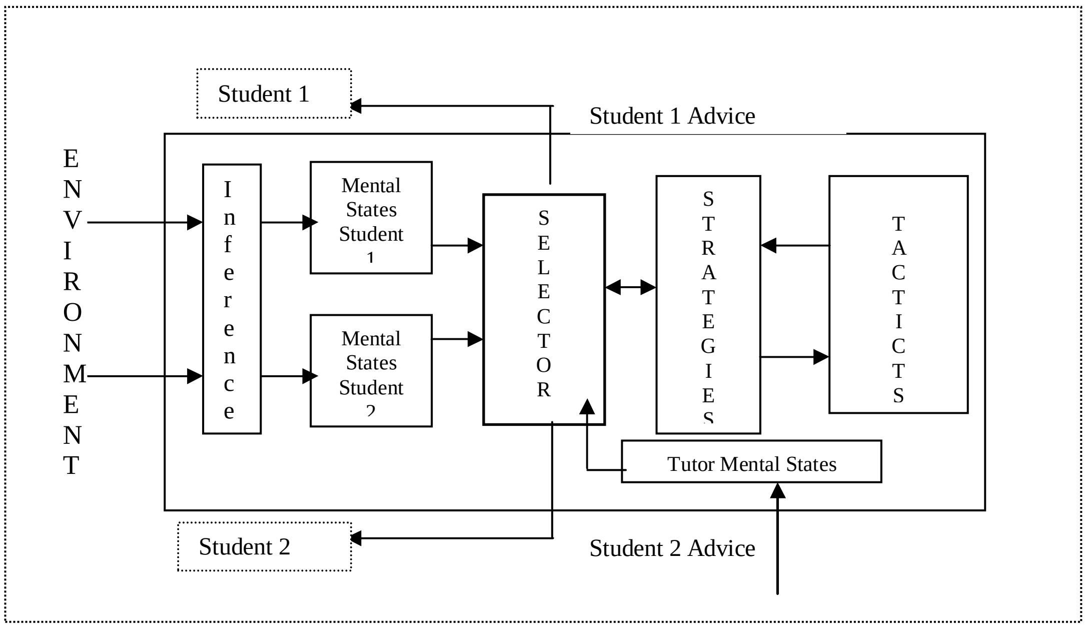

Figure 1: Behavioural Decision Centre

The tutor receives information about each student model. The information is composed by the mental states of each student, after they perform an action, and the environment energy levels. The tutor uses it to select a specific strategy with the associated tactics, according to the selector module.

The set of the students' mental states is obtained through a module of inference of attitudes, as presented in Quaresma[30]. The dialogues are seen as composed by a sequence of speech acts. According to [31], to speak is to act. When agents speak they accomplish speech acts. The speech acts, according to Cohen and Perrault [6], are actions whose effects affect both the originator and the receiver. Quaresma's work offers a formal and computational model of how to infer the mental states of the participants in a dialogue. Although this is not part of the implementation of the MCOE system, it is considered in the BDC project as an explanation of how to obtain the students' mental states.

The function of these selected strategies is to give to students (with different profiles) under the same tutor supervision a personal orientation so that they can better achieve their goals.

All students have the same desire: to win the game. All students have strong desires that move their actions, these desires having urgency attributes that compel them to achieve their intention. The differences among agents are on their beliefs about themselves, their own knowledge, and ability to solve the problem.

We selected a set of tactics to implement the tutor's actions related with the selected strategies. The mental states will be used to select the tutor behaviour. The tutor can behave as reactive, a coaching, or an assistant tutor. Each of the selected behaviour is connected with a set of rules, which will determine the specific tactics.

The tutor works through pedagogical actions and induces the student to reflect about his/her actions and create strategies with associated tactics. The tactics are the action that will be performed by the tutor and expected by the student. We selected a small piece from a dialogue, in order to illustrate how to use the architecture and the formalism in our application. Table 1 presents some examples about the mental states formalised using Mora's computational model.

Student 1 is playing with the Mayor character and student 2 is playing with the Mother Nature character. Both observe the visual gauge that aids them to have a clue about the environment condition and the colour is yellow. They have been playing for three minutes and there remains two minutes more. They just have few tools available in each character. Student 1 has two powerful tools reaming and student 2 has just one powerful tool to play.

Student 1 says to Student 2:

I am going to play with restore de bottom of the lake because the Ecometer is yellow and we have all kinds of fishes in the lake and we have many plants in the bottom. The situation is not dangerous.

He beliefs that if the time is more than 1 minute they must be careful because more pollution can come and they need to have a powerful tool to fight against it. He knows his colleague just have one powerful tool. He has few powerful tools, however his situation is better than his colleague's. The Ecometer is yellow and the number of fish and plants do not indicates a dangerous situation.
The tutor sent a message just warning about the number of reaming tool for this character. In the set of mental states of the tutor we find:

The ecometer now is green. Student 2 waits and she does not do anything.
She believes that if the Ecometer is green it is not necessary to do anything. The tutor does not have any reaction and wait. In the set of mental state that the tutor have about student 2, the environment and about the students behaviour we find:

Table 1: The mental sates expressed using the formalism

Some example of students mental states	
DES(student, exchange_message)) if BEL(student,energy(ecometer,Ee)), Ee<70, BEL(student, number_tool(changebottom, N)), N <= 2, BEL(student, number_tool(sunlight, N)), N <= 2, BEL(student, number_tool(put_plants, N)), N > 2, BEL(student, number_tool(decompsition, N)), N > 2, BEL(student, time(T)), T < 5, T > 3.	DES(student, exchange_message)) if BEL(student,energy(ecometer,Ee)), Ee<70, BEL(student, number_tool(prohibition, N)), N <= 2, BEL(student, number_tool(fine, N)), N <= 2, BEL(student, number_tool(campain, N)), N <= 2, BEL(student, number_tool(license, N)), N <= 2, BEL(student, time(T)), T < 5, T > 3.
DES (student1, play_changebottom) if BEL(student,energia(fishA,N)) if before(BEL(student,changebottom)), N=N+(N * 0.05). BEL(student,energia(fishB,N)) if before(BEL(student, changebottom)), N=N+(N * 0.05). BEL(student,energia(fundo,N)) if before(BEL(student,fishback)), N=N+(N * 0.05). BEL(student,energia(water,N)) if before(BEL(student, changebottom)),	INT (student, fight_foreign_elements) if BEL(student,energy(ecometer,Ee)), Ee<70, BEL(student, time(T)), T < 5.

N=N+(N * 0.15). BEL(student,energia(water,N)) if before(BEL(student, changebottom)), N=N+(N * 0.15).	
Some example of tutor mental states	
BEL(tutor, sent_message(7)) if BEL(tutor, is_a_character(mothernature)), BEL(tutor, number_tool(tchangebottom, N)), N <= 1, BEL(tutor, time(T)), T < 5, T > 2.	BEL(tutor, do_not_do_anything) if BEL(tutor,INT_THAT(student1, wait), BEL(tutor, energy(ecometer, Ee), Ee >= 70.

With this tutor'architecture we just describe the set of mental sates connected with the actions, describe the set of mental states that the tutor must have about the student and the application domain, and all the choreography will be held according the students actions. The use of mental states adds more information about the student (in a quantitative and a qualitative way). It allows us to try to reduce the gap between what the student is and what we can represent in the computer.

We have chosen to implement this application (MCOE system) to test the architecture because it is simple to program and reduces the complexity that exists in others agent's developing environments. Mora et ali's system allows the designers to write the set of mental sates in an easy way, as the formalism used is quite simple (similar to an action language). Also, the environment is compatible with different platforms (PC, Macintosh, and Unix). The cognitive kernel used in the MCOE system was linked to the graphical interface through sockets were the interface sends the set of mental states connected with students actions and the cognitive kernel sends back to the graphical interface the selected tactic chosen by the tutor.

4. Final considerations and future work

There is still a big gap in our knowledge of how natural mind work. However, by looking to the structure of mental states we probably may build replicas of these systems to help us to understand how these biological systems work. Such computer models are unlocking the secrets of a variety of artificial reactions. This can be the breakthrough we have been waiting for to design cognitive agents tuned to particular problem domains.

The agent's behaviour is based on the fact that the only thing the programmer has to do is to specify the agent's mental states. According to [12], the inclusion of expectations in the traditional BDI architectures enables bigger flexibility because, thoroughness, and more complex behaviours. The ideas behind this social approach introduce promising concepts to understand how to construct agents with special abilities, for example, to learn and teach [16, 17, 18].

What we want to achieve with our research group?
The acquisition of knowledge by the agents in a multi-agent society emerges from their interactions in which each agent can behave as a tutor or as a learner. As a matter of fact, there is no tutor and no learner, but only tutor and learner attitudes. Is important to highlight that the Tutor only has the personal model of each one student but does not have the collective model yet.

5. References

1. Akhras, F; Self, J. Modelling Learning as a Process. In: AI-ED97: Eighth World Conference on Artificial Intelligence in Education, 8., 1997. **Proceedings...** Kobe: Japan, 1997.

2. Aimuer, E.; Frasson, C. Analysing a new learning strategy according to different knowledge levels. In: **Computers Education**. London: Pergamon, 27(2), 1996.
3. Andre, E.; Muller, J.; Rist, T. Life-Like Presentations Agents: A New perspective for Computer Based Technical Documentation. In: AI-ED97: Eighth World Conference on Artificial Intelligence in Education - Workshop V: Pedagogical Agents, 8., 1997. **Proceedings...** Kobe: Japan, 1997.
4. Balacheff, N.;, Kaput J. (1997) Computer-Based Learning Environments in Mathematics, in: A. Bishop et al. (eds) **International Handbook in Mathematics Education** (pp.469-501). Dordrecht : Kluwer Academic Publisher.
5. Bratman, M.E.. What is intention? In P.R. Cohen, J.L. Morgan, and M. Pollack, editors, **Intentions in Communication.** The MIT Press, Cambridge, MA, 1990.
6. Cohen, P.R.; Perrault, R. **Elements of a Plan-Based Theory of Speech Acts.** Cognitive Science, 3:177-212, 1979.
7. Cohen, P.R; Levesque, H.J.. **Intention is choice with commitment**. Artificial Intelligence, 42,213--261, 1990.
8. Colazzo, L.; Silvestri, L. The pragmatics of the Tutor: A proposal of modelling. In: AI-ED97: Eighth World Conference on Artificial Intelligence in Education - Workshop V: Pedagogical Agents, 8.,1997. **Proceedings...** Kobe: Japan, 1997.
9. Corrêa,M. **A Arquitetura de Diálogos entre Agentes Cognitivos Distribuídos**. Rio de Janeiro: Universidade Federal do Rio de Janeiro, Pós-Graduação em Engenharia, Rio de Janeiro, 1994. (Ph.D. Thesis -Portuguese)
10. Corrêa, M.; Mendes, S. A Computational Approach to Situation Theory Based on Logic Programming to design Cognitive Agents. In: Brasilian Symposium on Artificial Intelligence, 11., 1995, Campinas. **Proceedings...** Campinas: Springer Verlag, 1995.
11. Corrêa, M; Coelho, H. A Framework for mental States and Agent Architecture. In :MASTA'97: Multi-Agent System: Theory and Applications, 1., 1997. **Proceedings...** Coimbra: DE-Universidade de Coimbra, 1997.
12. Corrêa,M.; Viccari,R.M.; Coelho,H. Dynamics in Transition Mental Activity. In: ICMAS'98 –International Conference on Multi-Agent Systems. **Proceedings…**Paris, 1998. (Submitted paper)
13. Costa, E.B.; Perkusich,A. A Multi-Agent Interactive Learning Environment Model. In: AI-ED97: Eighth World Conference on Artificial Intelligence in Education - Workshop V: Pedagogical Agents, 8., 1997. **Proceedings...** Kobe: Japan, 1997.
14. Elliott, C. Affective Reasoner Personality Models for Automated Tutoring Systems. In: AI-ED97: Eighth World Conference on Artificial Intelligence in Education - Workshop V: Pedagogical Agents, 8., 1997. **Proceedings...** Kobe: Japan, 1997.
15. Frasson, C.; Mengelle,T.; Aimeur,E. Using Pedagogical Agents in a Multi-Strategic Intelligent Tutoring System. In: AI-ED97: Eighth World Conference on Artificial Intelligence in Education - Workshop V: Pedagogical Agents, 8., 1997. **Proceedings...** Kobe: Japan, 1997.
16. Giraffa, L.M.M; Viccari, R.M. Tutor behaviour in a multi-agent ITS guided through mental states activities. Workshop: Pedagogical Agents. In: ITS'98- Fourth International Conference on Intelligent Tutoring Systems. **Proceedings…** San Antonio, Texas, 1998.
17. Giraffa, L.M.M; Mora, M.; Viccari, R.M Modelling the MCOE Tutor using a Computational Model. In: **Lectures Notes on Artificial Intelligence -** SBIA'98. Oliveira, F. (Ed.). Berlin: Springer Verlag, 1998.
18. Giraffa, L.M.M.; Viccari, R.M. Educação Ambiental suportada por um Ambiente de Ensino Inteligente RIBIE: Red Iberoamericana de Informatica Educativa, 1998. **Anais…**Brasilia: RIBIE, 1998.
19. Giraffa, L.M.M; Mora, M.; Viccari, R.M. Pedagogical game using ITS architecture. Workshop Multi-Agents Architectures -IBERAMIA'98. **Proceedings...** Toledo: Spain, 1998.
20. Hayes-Roth, B. et alli. **Directed Improvisation**. Stanford: CA, 1994. (Technical Report KSL-94-61)
21. Hayes-Roth, B. et alli. Directed Improvisation with animated puppets. CHI'95: Conference on Human-Computer Interaction, 1., 1995. **Proceedings...** Denver: 1995.
22. Hayes-Roth,B.; Van Gent,R.; Huber,D. Acting in Character. In: **Creating Personalities for Synthetic Actors:** towards autonomous personality agents. Trappl, R.; Petta,P. (Eds.). Berlin: Springer Verlag, 1997.
23. Leroux ,P. Co-operation between a Pedagogical Assistant, a Group of Learners and a Teacher. In: Euro AI-Ed - European Conference on Artificial Intelligence in Education, 1., 1996. Brna,P et Alii (Eds.). **Proceedings...**Lisboa: Colobri,1996.
24. Lester, J. et alli. Mixed Initiative Problem Solving with Animated Pedagogical Agents. In: AI-ED97: Eighth World Conference on Artificial Intelligence in Education - Workshop V: Pedagogical Agents, 8., 1997. **Proceedings...** Kobe: Japan, 1997.
25. Mitsuru,I. Et Alii. Opportunistic Group Formation. In: AI-ED97: Eighth World Conference on Artificial Intelligence in Education - Workshop V: Pedagogical Agents, 8., 1997. **Proceedings...** Kobe: Japan, 1997.

26. Móra, M.C.; Lopes, J.G.; Coelho, J.G.; Viccari, R. Modelling dynamic aspects of intentions. In E.Costa, editor, 8th Portuguese Conference on Artificial Intelligence. Springer-Verlag, **Proceedings...**, 1997.
27. Móra, M.C.; Lopes, J.G.; Coelho, J.G.; Viccari, R. Modelling agents with extended logic programme. In International Workshop on Engineering of Intelligent Systems. ICSC co., **Proceedings...** 1998.
28. Móra, M.C.; Lopes, J.G.; Coelho, J.G.; Viccari, R. BDI models and systems: Reducing the gap. In: Agents Theory, Architecture and Languages Workshop. **Lecture Notes on Artificial Intelligence**, Springer-Verlag, 1998.
29. Moussale, N.M.; Viccari, R.M.; Correa, M. Intelligent Tutoring Systems Modelled Through the Mental States. In: **Lectures Notes on Artificial Intelligence -** SBIA'96. Borges,D. and Kaestner,C.(Eds.). Berlin: Springer Verlag, 1996.
30. Quaresma, P. **Inferência de atitudes em diálogos.** Lisboa: FCT/UNL, 1997. (Ph.D. Thesis -Portuguese).
31. Searle, J.. What is an intentional state? In H.Dreyfuss and H.Hall, editors, Husserl, **Intentionality and Cognitive Science**, (42), 1984.
32. Self, J. Dormorbile: a Vehicle for Metacognition. In: **Emerging Computer Technologies in Education.** Self, J. (Ed.). Charlottesville, AACE, 1995.

11 ICCMA99 – “Modelling and Interactive ITS using a MAS Approach: from Design to Pedagogical Evaluation”

Este artigo, apresentado no ICCMA99, faz uma avaliação do tutor MCOE, já completo e utilizando a primeira versão da ferramenta de implementação de agentes resultante do modelo teórico apresentado, a *X-BDI*. O X-BDI é um kernel independenque, dado uma definição do agente em termos de suas crenças e desejos, é capaz de gerar planos e intenções, bem como fazer a revisão de intenções e crenças. Além de experimentar novamente com aspectos de implementação e projeto dos agentes segundo o modelo proposto, se faz uma análise de como esta implementação de agentes deve ser utilizada para satisfazer as necessidades pedagógicas nas classes de tutores trabalhadas.

Modelling an Interactive ITS Using a MAS Approach: From Design to Pedagogical Evaluation[1]

Lucia M. M. Giraffa
giraffa@inf.pucrs.br
FACIN/PUCRS[2]
CPGCC/UFRGS[3]

Michael da C. Móra
michael@inf.pucrs.br
CPGCC/UFRGS
FACIN/PUCRS

Rosa M. Viccari
rosa@inf.ufrgs.br
CPGCC/UFRGS

Abstract

The paper shows how a Multi-Agent Systems (MAS) approach may be used to build an interactive Intelligent Tutoring Systems (ITS) designed with a game. We model the ITS as a society of both reactive and cognitive agents that interact through a graphical interface. We present the experiment made in school in order to test our hypothesis about the architecture pedagogical potential, and the results obtained.

1. Introduction

We have been doing many developments in our group in order to explore the possibilities to build ITS using a MAS approach, and specially modelling the tutor using a set of mental sates, such as beliefs, intentions, motives, expectations, obligations and so on. Some results were published in Moussalle (1996), Mora et al (1997,1998), Giraffa, Viccari and Self (1998), Giraffa, Mora and Viccari (1998a, 1998b, 1998c), and Silveira and Viccari (1997). In this paper we present a different aspect from our research, the evaluation process made with teachers from schools of our city about the improvements that can be overcome using a MAS approach to design a educational game.

The paper is organised as follows: section 2 describes the games named Eco-Lógico and MCOE (Multi CO-operative

[1] Extended version of the paper accepted in thc ICCMA'99.
[2] Faculdade de Informática – Pontifical Catholic University of Rio Grande do Sul. Av. Ipiranga 6681 – prédio 16 – Porto Alegre – RS – Brazil 90610-900.
[3] Curso de Pós-Graduação em Ciência da Computação - Federal University of Rio Grande do Sul. Av. Bento Gonçalves 9500 – Bloco IV – Porto Alegre – RS – Brazil.

Environment), section 3 describes the methodology developed for the experiment, section 4 presents the results, section 5 point out our future work, and section 6 presents the references.

2. The games

After three years of research we have developed two pedagogical games with different software engineering approaches: with and without MAS architecture. They are not exactly the same game but they have the same domain, and the same working principles. The following sections present more details about the both games.

2.1 Eco-Lógico

The Eco-Lógico is an educational game developing using Toolbook Multimedia[4]. The domain application is Ecology (water pollution). In this ecological environment the student chooses a character to play using six options: Mother Nature, Mayor, Citizen, Fish, Ecologist, and Tourist. After that, the student defines the game configuration (foreign elements that will cause pollution) by a selecting process from a graphical menu. The student's first see a lake in equilibrium (represented trough a background bitmap) where there are different types of fish swimming. After 30 seconds the chosen polluter's appears which soon begins to show their action (take out energy from scenery elements).

There is a visual gauge (the Ecometer) that helps the user to observe the conditions of the lake. A well-balanced situation is shown when the gauge is full and green. As the equilibrium is lost, the colour changes gradually to yellow, finally changing to red when a dangerous situation is reached. The number of fish swimming in the lake indicates the situation. The student's challenge is to maintain the equilibrium and fight against the action of these foreign elements using a set composed by four tools. Each character has its own tools with different power to fight against the pollution (put energy on scenery elements). The period of time is five minutes.

[4] Toolbook Multimedia is a trade mark from Assimetrix Inc.

The game was modelling to be played by one student. There are sounds effects during the animations, and some information about the scenery elements can be obtained pressing the mouse right button. At the end the student receives a message explain if s/he achieve the goal or not.

2.2 MCOE

The conception of the ‘MCOE’ was based on an ITS architecture (Giraffa, Viccari, Self, 1998). This architecture is composed of a hybrid society of agents that work to achieve a common goal: to fight against the pollution resulting from foreign elements (pollutants) and to maintain the system equilibrium. We have *reactive agents* (bottom of the lake, micro-organisms - plankton, water, plants and three types of fish) and *cognitive agents* (the Tutoring agent, and students represented by characters).

The *cognitive agents* are modelled using a *mentalistic approach*, where the term agent means a computer system that can be viewed as consisting of mental states such as beliefs, intentions, motives, expectations, obligations and so on. In this case the question of which is an agent is replaced by the question of which entities can be viewed as possessing mental states. The model of each student and the tutor contains a set of basic beliefs, desires and expectations[5]. From this set emerges the dynamic selection done by Tutor to select a personal teaching strategy. The formalism using is *logic programming extended with explicit negation* (ELP) with the ***W**ell-**F**ounded **S**emantics e**X**tended for explicit negation* (WFSX). ELP with WFSX extends normal logic programs with a second negation named *explicit*, in addition to the usual *negation as failure* of normal logic programs, which is called *implicit negation* in the ELP context. This extension allows us to explicitly represent negative information (like a belief that a property *P* does not hold, or an intention that a property *P* should not hold) and increases the expressive power of the language[6]. The cognitive agent’s implementation was made using Prolog.

[5] The expectation is considered for us as a future belief.

[6] More details see the references, specially 1998b, and 1998c

We designed the system to be played by two students. The first student chooses a character to play using four options: Mother Nature, Mayor, Citizen, and Tourist. After that, the second student can choose one of the three remaining characters. The system defines the game configuration (foreign elements that will cause pollution) by a sorting process using a random function. Their challenge is to maintain the equilibrium and fight the action of these foreign elements for a period of time previous defined by the teacher or the students.

The three types of fish, micro-organisms, and plants compose the reactive agents. So, when the game begins the student see a lake with the select scenery elements (sorting by the system trough a random function) which soon begins to show the action of the polluters. The number of plants, micro-organisms and fish swimming in the lake indicate the situation. There is a visual gauge (the Ecometer) that helps the user to observe the conditions of the lake. A well-balanced situation is shown when the gauge is full and green. As the equilibrium is lost, the colour changes gradually to yellow, finally changing to red when a dangerous situation is reached.

The system equilibrium is based on the energy level, the occasional predatory actions that can happen in the environment and the number of elements that remain in the system after the pollutants impact. The fish, plankton, and plants also reproduce using a function based on the energy level. However, when they reproduce they naturally loose energy, because they give energy to their descendants. If an agent dies (zero energy), it is removed from the system.

The reactive environment was built with Visual C++. The game’s main window is implemented with DirectX, and the scenery and its elements are implemented with Direct3D Retained Mode. The controls that the player can select during the game are implemented as bitmaps controlled by DirectDraw. The objects were models in 3D Studio MAX 1.0. The relationship between the agent and the environment is affected by the occurrence of external incentives (polluters) that cause some partial reaction of the agent. Each example of an agent can have reaction that might differ even if the same incentives is present and will always change during the lifespan of each one. Initially, we made a set of actions rules for each agent’s reaction. The agents are repositories of actions and data. The system control is always tracing each event for the respective object-action that was instanced to treat that event.

The reactive environment is connected with the cognitive kernel trough an interface made using socks that allow both parts exchange data. The cognitive kernel was implemented using Mora's environment (Mora et al, 1997; Mora et al, 1998).

There is a help system wich allows the student solve some doubts about the game rules, the domain, and a small dictionary (because we use some technical words to explain the polluters).

3. The experiment

We built an instrument under to evaluate some aspect of the system according to the point of view of the teacher. It is a form with 26 questions to put a stick for all aspects studied and 2 open questions that allows teacher put comments and suggestions about the systems and possible improvements.

The goals of this questionnaire are:

- Verify how the two interfaces influences the teachers perception about the educational possibilities of the two games (Eco-Lógico e MCOE) as an auxiliary tool to aid to teach their students;
- Verify the possibilities of domain exploration (water pollution) increased using agent's techniques, since the two systems (Eco-Lógico and MCOE) have the same goals and the same contents.

The hypotheses are:

1) The possibilities of domain exploration (educational point of view) in the ITS building using as MAS architecture using multiple strategies are bigger than the conventional architecture;

2) The understanding of goals of the game increase using the MAS architecture;

3) The agent's oriented interface is more dynamic and interactive than the interface implemented with Toolbook.

Six teachers, from third and fourth level of three elementary schools compose the sample for our experiment.

The experiment was applied in two phases. First phase we explain the two games for the teachers, and explain how the both games work. After that, we gave a little printed overview about the domain (water pollution), the games user manual, put the teachers interact with the games in order to answered their doubts about the way how the two systems work. We explain also, that the main goal is obtain their impressions about the pedagogical potential of the two environments, and the positive and negative aspect that they found in each version. During this phase we took notes about their first impressions and comments.

The second phase was the instrument application. We gave the instruments for the teachers full fill in the laboratory after playing with the two games. They full filled the instruments alone without our presence. The identifications data just have the initials of the teacher's name, age, if is a private or public school, if it is in the city or surroundings, and the teachers' origin.

The instrument was analysed and the results will be presented on the next section.

4. The results

The teacher's behaviour profile when they selected activities to aid to teach their students indicates that:

- Preferentially they chose interactive activities in the environment (school or surroundings) for introduce the study of pollution (43%), and they selected artificial environments (software or videos) as a complementary activity in order to reinforce the knowledge of their students about pollution (60%);
- 66% of the teachers use educational software more than 5 times a year[7];
- 83% of the teachers have preference for educational software in Portuguese;
- 83% of the teachers have preference for games when they select a educational software;
- 50% of the select educational software by the teachers have sound resources and animations;
- 83% of the teachers think that interaction possibility of the educational software is important for the work with the

[7] A year means a school year in Brazil, from March to December, with 15 vacations days on July.

students;

- 50% of he select educational software by the teacher's record the student work during the interaction. They said that if is possible have the log file of the student's interactions it will be very useful for them in order to better analyse the student learning process;
- 66% of the select educational software by the teachers allow time control of the activity;
- 83% the select educational software by the teachers did not modify their interface when student chose play with them again. However, they said it is something very important that allow them to organise better the activities according their personal goals with the students.
- 83% the select educational software by the teachers have some kind of help system in order to understand how the program works;
- 66% the select educational software by the teachers do not have a help system to aid the student to understand the domain application (contents). Some of them give emphasis that it is very important in order to help the students solve some small doubts that can arise during the interaction.

These results are very important for designers that want to develop educational software for our reality. We must to know how the teacher select their software, and if the thinks we believe that are important in an educational game have a tuning with the teacher's expectation.

The comparison between the Eco-Lógico and MCOE made by the teachers gave us very important results that verify our hypothesis:

- Suitable representation of the lake ecosystem: MCOE (33%), Eco-Lógico (33%), both systems (16%);
- Better represents the life in a lake: MCOE (100%);
- Significant contents to be explore by the teachers and students: MCOE (100%);
- Dynamically of the interface elements: MCOE (33%), Eco-Lógico (33%), both systems (16%);
- Easier to be understood in the first time the student play with: MCOE (83%), Eco-Lógico (17%),
- Suitable for work with their students: both systems

(100%);

One of the most significant results for us was the 100% achieved on significant contents. The domain is the same, and the internal rules are the same as well. However, all the teachers had the perception that MCOE has more significant contents than the other game. The Eco-Lógico just use the strategy created by the expert (the teacher used to model the domain and the game). The MCOE has multiple strategies in its design conception. Using multiple strategies for modelling the tutor behaviour we need to sophisticate the interface in order to support this intrinsical pedagogical aspect. The use of agent's techniques helps us to achieve this degree of sophistication necessary to our project. In case of ITS and in Intelligent Learning Environments (ILE) we can consider agents as Pedagogical Agents (Frasson, 1998; Lewis, 1998). Pedagogical agents have some fundamental properties: autonomously, social ability, proactiveness, and persistence. Some of them can be reactive, continuously performing, capable to learn and a character represents most of them. The MCOE agent's has all this properties, except capable to learn (at this moment).

This set of properties gave characteristics for the pedagogical agents that are very important under pedagogical point of view. In the reactive agents this properties improve the quality of domain representation, increase the feedback for the student's actions and their strategies in order to solve the problem. We must to remember that the game (MCOE) is a problem-solving environment where the students have multiple challenges during the same section. The possibility to show the environment in different aspect using agent's creates the feeling that each time the system is loaded you have a new challenge with unpredictable situations to solve. Even the characters are the same, the lake and its elements will not be equal as previous section. The user has a quick feedback that allows him/her to make reflections about his/her actions and its consequences. By the side of cognitive agents, the autonomy guarantees the conditions in order to achieve their personal goals. In an educational environment the agent has to analyse its own actions and the reflections of this on the environment (proactiveness property), and took in account to act again. The social ability are implicit in the keyboard actions (press a selected tool from a character), and codify in the set of beliefs connected wit the student's actions.

The inclusion of the student model built using BDI architecture (it is an internal MAS composed by the mental states represented as an agent) give more that just sensorial information for the tutor. It describes the current sate of the student and at the same time did not labelled the student according previous stereotypes. The student is considering the way hi/her is.

5. Future work

The results allow us to have important elements to continue our research with pedagogical games using MAS architecture. The use of MAS and Multimedia allow us to explore different possibilities that we cannot do with traditional techniques. We can build a more powerful and realistic interface for simulations environments like games designed as an ITS. The actions on screen are more dynamic and express better what is going on in the system.

We are preparing a new experiment to be running in the schools with the students in order to measure how the MCOE aids the learning process. The goal of this experiment is verify how much knowledge the student retain or remember about the pollution control. We have a multidisciplinary team that work together to achieve this goal. Concurrent this, a new project is on the way to put reinforcement learning in the reactive agents (fish) in order to create a more interesting environment for student's exploration.

Acknowledgements
We would like to thank Dr. Milton Madeira (from faculty of Psychology/PUCRS), Leandro Bernsmüller (implementation and Computer Graphics support), and André Raabe (to help us with the evaluation process at schools).

6. References

Frasson, C.; Mengelle,T.; Aimeur,E. Using Pedagogical Agents in a Multi-Strategic Intelligent Tutoring System. AI-ED97: Eighth World Conference on Artificial Intelligence in

Education - Workshop V : Pedagogical Agents, 8.,1997. **Proceedings...** Kobe: Japan, 1997.

Giraffa, L.M.M.; Nunes, M.A.; Viccari, R.M. (1997) Multi-Ecological: an Intelligent Learning Environment using Multi-Agent architecture. MASTA'97: Multi-Agent System: Theory and Applications. **Proceedings...** Coimbra: DE-Universidade de Coimbra.

Giraffa, L.M.M; Viccari, R.M.; Self, J. (1998) Improving tutoring activities using a Multi-Agents system Architecture. Twenty-ninth SIGCSE Technical Symposium on Computer Science Education, 29,1998. **Proceedings...** Atlanta: Georgia.

Giraffa, L.M.M; Viccari, R.M. (1998a) Tutor behaviour in a multi-agent ITS guided through mental states activities. Workshop: Pedagogical Agents. In: ITS'98- Fourth International Conference on Intelligent Tutoring Systems. **Proceedings…** San Antonio, Texas.

Giraffa, L.M.M; Mora, M.; Viccari, R.M (1998b) Modelling the MCOE Tutor using a Computational Model. In: **Lectures Notes on Artificial Intelligence** - SBIA'98. Oliveira, F. (Ed.). Berlin: Springer Verlag.

Giraffa, L.M.M.; Viccari, R.M. (1998c) Educação Ambiental suportada por um Ambiente de Ensino Inteligente RIBIE: Red Iberoamericana de Informatica Educativa, 1998. **Anais…**Brasilia: RIBIE.(Portuguese)

Giraffa, L.M.M; Mora, M.; Viccari, R.M. (1998d) Pedagogical game using ITS architecture. Workshop Multi-Agents Architectures -IBERAMIA'98. **Proceedings...** Toledo: Spain.

Lewis, W J.; Shaw,E. Using Agents to Overcome Deficiencies in Web-Based Courseware. AI-ED97: Eighth World Conference on Artificial Intelligence in Education - Workshop V: Pedagogical Agents, 8.,1997. **Proceedings...** Kobe: Japan, 1997.

Móra, M.C.; Lopes, J.G.; Coelho, J.G.; Viccari, R. (1997) Modelling dynamic aspects of intentions. In E.Costa, editor, **8th Portuguese Conference on Artificial Intelligence**. Springer-Verlag.

Móra, M.C.; Lopes, J.G.; Coelho, J.G.; Viccari, R. (1998) Modelling agents with extended logic programme. In **International Workshop on Engineering of Intelligent Systems**. ICSC co.

Móra, M.C.; Lopes, J.G.; Coelho, J.G.; Viccari, R. (1998) BDI models and systems: Reducing the gap. In: Agents Theory, Architecture and Languages Workshop. **Lecture Notes on Artificial Intelligence**, Springer-Verlag.

Moussale, N.M.; Viccari, R.M.; Correa, M. (1996) Intelligent Tutoring Systems Modelled Through the Mental States. In: **Lectures Notes on Artificial Intelligence** - SBIA'96. Borges,D. and Kaestner,C.(Eds.). Berlin: Springer Verlag.

Raabe, A.L.A; Javimczik, A .M.; Giraffa, L.M.M. Eco-Lógico: Ambiente interativo para suporte ao ensino de educação ambiental. Simpósio Brasileiro de Informática na Educação, 7., 1996, **Anais...** Belo Horizonte: DCC/UFMG, 1996. (Portuguese)

Silveira,R.A.; Viccari,R.M.(1997) Projeto Eletrotutor: Desenvolvimento e Avaliação de Ambientes Inteligentes de Ensino-Aprendizagem. CLEI-PANEL'97:XXIII **Conferencia Latinoamericana de Informática**. Valpariso, Chile.

12 Conclusões

Este trabalho propôs um modelo formal de agentes que, simultaneamente, apresenta as vantagens dos modelos formais existentes:

- pode ser usado como ferramenta de especificação e validação de agentes;
- reduz a distância entre especificação e implementação de agentes, na medida que o modelo formal do agente é utilizado para implementar o mesmo agente.

A teoria proposta permite descrever agentes e comportamentes autônomos e flexíveis, como aqueles mostrados no exemplo 1. O trabalho segue a abordagem intencional, ou seja, os agentes são descritos pelos seus estados mentais. O modelo formal define quais são os estados mentais que compõem o agente, qual a relação entre estes estados mentais e como os comportamentos dos agentes são produzidos a partir destes estados mentais. Além disto, o modelo adota uma *perspectiva do agente* [MÓR 95] [SIN 94], ou seja, ao invés de enxergar o modelo simplesmente como uma ferramenta de especificação a ser utilizada pelo projetista, este também é visto como uma ferramenta a ser utilizada pelo agente para representar os estados mentais e raciocinar sobre eles.

O formalismo utilizado para construir o modelo é a *programação em lógica estendida com negação explícita* com a semântica bem fundada com negação explícita (*WFSX* – *W*ell *F*ounded *S*emantics for e*X*tended programs), baseando-se no trabalho de Pereira et al. [ALF 96]. Esta semântica para programas em lógica estendida com negação explícita foi adotada devido essencialmente às suas propriedades e devido a existência de procedimentos de prova corretos e completos com relação à *WFSX*, bem como de procedimentos que permitem a detecção e a manipulação de contradições e a implementação de diversas formas de raciocínio não monotônico.

A principal contribuição desta tese é a definição de uma teoria BDI de agente formal e executável que permite tanto definir e analisar formalmente agentes e propriedades destes agentes, bem como executar estes agentes, a partir de sua definição BDI. O desenvolvimento desta teoria deu origem a outras contribuições: extensão do cálculo de eventos para suportar ações concorrentes e eventos expontâneos, o que permite utilizar este modelo para representar ambientes dinâmicos e onde múltiplos agentes atuem; definição formal e computável de processos associados aos estados mentais, nomeadamente: criação de intenções a partir de desejos, criação de ações a partir das intenções, resolução de conflitos entre desejos. Estes processos não são comtemplados nos modelos formais existentes; integração de diversas formas de raciocínios na definição dos agentes, nomeadamente dedução, abdução e raciocínio revogável; implementação de um ambiente de teste de agentes onde o modelo formal é executado, de modo que a arquitetura BDI passa a ser um paradigma de implementação de agentes.

Claramente, este trabalho não se esgota em sí só. Muito há ainda por fazer como trabalhos futuros:

- no desenvolvimento da ferramenta, não houve preocupação com performance. Se, por um lado, a implementação deriva diretamente dos axiomas do modelo formal, ela hoje está feita como um meta-interpretador sobre o interpretador Prolog. É importante elaborar algorítmos eficientes e que sejam verificados contra a teoria apresentada;

- o modelo não trata da questão da adaptabilidade do agente que envolve processos de aprendizagem. É necessário estender o modelo de modo que aprendizagem possa fazer parte do agente construído;

- embora o modelo permita tratar de mais de um agente (como foi o caso no tutor desenvolvido), ele não está customizado para lidar com múltiplos agentes. É necessário estender a teoria de modo que a representação de múltiplos agentes, bem como aspectos usuais de sistemas multiagentes como comunicação e coordenação façam parte da teoria, e se propaguem para a ferramenta de desenvolvimento;

- por fim, há diversas questões referentes a aspectos de engenharia de software que devem ser mais profunamete analisados: qual o papel de um modelo BDI como este em uma metologia mais de desenvolvimento de sistemas multiagentes (ou uma metodologia orientada a agentes); dado o modelo teórico e a ferramenta desenvolvidas, qual a metodologia para definição do agente; verificação se questões como reusabilidade e modificação de sistemas podem ser bem resolvidadas quando a abordagem é orientada a agentes, com uma ferramenta de descrição formal como a que foi aqui desenvolvida.

Bibliografia

[ALF 95] ALFERES, J.; DAMÁSIO, C. A logic programming system for non-monotonic reasoning. **Journal of Automated Reasoning**, [S.l.], v.14, p.93–147, 1995.

[ALF 96] ALFERES, J. **Reasoning with Logic Programming**. Berlin, DE.:Springer-Verlag, 1996. Lecture Notes in Artificial Intelligence Series (LNAI 1111).

[BEL 95] BELL, J. A planning theory of practical reasoning. In: PROCEEDINGS OF THE AAAI'95 FALL SYMPOSIUM ON RATIONAL AGENTS CONCEPTS, 1995. [s.n.], 1995.

[BEL 97] BELL, J. Dynamic goal hierarchies. In: PROCEEDINGS OF THE SECOND WORKSHOP ON PRACTICAL REASONING AND RATIONALITY, 1997. **Proceedings...** London, England:[s.n.], 1997.

[BRA 87] BRATMAN, M. **Intentions, plans and practical reasoning**. Cambridge, MA:Harvard University Press, 1987.

[BRA 88] BRATMAN, M.; ISRAEL, D. Plans and resource bounded practical reasoning. **Computational Intelligence**, [S.l.], v.4, p.349–355, 1988.

[BRA 90] BRATMAN, M. What is intention? In: Cohen, P.; Morgan, J., editors, INTENTIONS IN COMMUNICATION, chapter1. The MIT Press, Cambridge, MA, 1990.

[BUR 92] BURMEITER, B. Cooperative problem solving guided by intentions and perceptions. In: Werner, E., editors, DESCENTRALIZED AI 3 – PROCEEDINGS OF THE THIRD EUROPEAN WORKSHOP ON MODELLING AUTONOMOUS AGENTS AND MULTI-AGENTS WORLDS (MAAMAW'91), 1992. **Proceedings...** Amsterdam, The Netherlands:Elsevier Science Publishers, 1992.

[CAS 97] CASTELFRANCHI, C. To be or not to be an “agent”. In: PROCEEDINGS OF THE INTELLIGENT AGENTS III: AGENT THEORIES, ARCHITECTURES AND LANGUAGES, 1997. **Proceedings...** Budhapest, Hungary:[s.n.], 1997. Lecture Notes in Artificial Intelligence (LNAI 1193).

[CHE 80] CHELLAS, B. **Modal Logic: an Introduction**. Cambridge, UK:Cambridge University Press, 1980.

[CHU 81] CHURCHLAND, P. Eliminative materialism and the propositional attitudes. **Journal of Philosophy**, [S.l.], v.78, p.67–90, 1981.

[COH 90] COHEN, P. Intention is choice with commitment. **Artificial Intelligence**, [S.l.], v.42, p.213–261, 1990.

[COR 93] CORRêA, M. Around the architectural approach to model conversations. In: PROCEEDINGS OF THE FIFTH EUROPEAN WORKSHOP ON MODELLING AUTONOMOUS AGENTS AND MULTI-AGENTS WORLDS (MAAMAW’93), 1993. [s.n.], 1993.

[COR 95] CORRêA, M. A computational approach to situation theory based on logic programming to design cognitive agents. In: ADVANCES IN ARTIFICIAL INTELLIGENCE: PROCEEDINGS OF THE 12TH BRAZILIAN SYMPOSIUM ON ARTIFICIAL INTELLIGENCE, 1995. Springer-Verlag, 1995. (Lecture Notes on Artificial Intelligence 991).

[COR 98] CORRêA, M.; VICCARI, R. Agent behaviors from mental states dynamic. In: ACTAS DO 6TO CONGRESSO IBEROAMERICANO DE INTELIGÊNCIA ARTIFICIAL (IBERAMIA’98), 1998. [s.n.], 1998.

[COS 92] COSTA, M. **Introdução à Lógica Modal Aplicada a Computação**. Rio de Janeiro, BR:SBC, 1992. (in portuguese).

[DAM 94] DAMÁSIO, C.; NEJDL, W. Revise: An extended logic programming system for revising knowledge bases. In: KNOWLEDGE REPRESENTATION AND REASONING. Morgan Kaufmann inc., 1994.

[DAV 80] DAVIDSON, D. **Essays on actions and events**. New York, NY:Oxford University Press, 1980.

[DEN 87] DENNET, D. **The intentional stance**. Cambridge, MA:The MIT Press, 1987.

[DEV 91] DEVLIN, K. **Logic and Information**. Cambridge University Press, 1991.

[DOY 79] DOYLE, J. A truth maintenance system. **Artificial Intelligence**, [S.l.], v.12(2), p.231–272, 1979.

[FRA 97] FRANKLIN, S. Is it and agent or just a program?: A taxonomy for autonomous agents. In: PROCEEDINGS OF THE INTELLIGENT AGENTS III: AGENT THEORIES, ARCHITECTURES AND LANGUAGES, 1997. **Proceedings...** Budhapest, Hungary:[s.n.], 1997. Lecture Notes in Artificial Intelligence (LNAI 1193).

[GÄR 88] GÄRDENFORS, P. **Knowledge in Flux: Modeling the Dynamics of Epistemic States**. London, UK:The MIT Press, 1988.

[GEN 87] GENESERETH, M. **Logical Foundations of Artificial Intelligence**. Los Altos, CA:Morgan-Kaufmann co., 1987.

[GEO 91] GEORGEFF, M. Reactive reasoning and planning. In: PROCEEDINGS OF THE NATIONAL CONFERENCE OF ARTIFICIAL INTELLIGENCE (AAAI'91), 1991. [s.n.], 1991.

[GOL 93] GOLDMAN, R. Intentions in time. 1993. Relatório técnico.

[HAD 96] HADDADI, A. **Communication and Cooperation in Agent Systems: a Pragmatic Theory**. Berlin, DE.:Springer-Verlag, 1996. Lecture Notes in Artificial Intelligence Series (LNAI 1056).

[HAL 92] HALPERN, J. A guide to completeness and complexity for modal logics of knowlwdge and belief. **Artificial Intelligence**, [S.l.], v.54(3), p.319–379, 1992.

[HAL 96] HALPERN, J. **PEGAR O TITULO CERTO**. LOCAL.:EDITOR CERTO, 1996.

[HIN 62] HINTIKA, J. **Knowledge and Belief**. Ithaca, US:Cornell Unievrsity, 1962.

[KIS 92] KISS, G. Variable coupling of agents to their environments: combining situated and symbolic automata. In: Werner, E., editors, DESCENTRALIZED AI 3 – PROCEEDINGS OF THE THIRD EUROPEAN WORKSHOP ON MODELLING AUTONOMOUS AGENTS AND MULTI-AGENTS WORLDS (MAAMAW'91), 1992. **Proceedings...** Amsterdam, The Netherlands:Elsevier Science Publishers, 1992.

[KON 86] KONOLIGE, K. **A Deduction Model of Beliefs**. London, UK:Pitman Publishing, 1986.

[KON 93] KONOLIGE, K. A representationalist theory of intentions. In: PROCEEDINGS OF THE XII INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE (IJCAI'93), 1993. **Proceedings...** Chambéry, France:[s.n.], 1993.

[KOW 86] KOWALSKI, R. A logic-based calculus of events. **New Generation Computing**, [S.l.], v.4, p.67–95, 1986.

[MCC 78] MCCARTHY, J. Ascribing mental qualities to machines. Menlo Park, CA, 1978. Relatório técnico.

[MES 92] MESSIAEN, L. **Localized abductive planning with the event calculus**. Leuven (Heverlee):Katholieke Universiteit Leuven, 1992. Tese de Doutorado.

[MÓR 95] MÓRA, M.; LOPES, J. Modelling intentions with extended logic programming. In: Wainer, J., editors, ADVANCES IN ARTIFICIAL INTELLIGENCE: PROCEEDINGS OF THE 12TH BRASILIAN SYMPOSIUM ON ARTIFICIAL INTELLIGENCE, 1995. **Proceedings...** Berlin, Germany:Springer-Verlag, 1995. Lecture Notes in Artificial Intelligence (LNAI 991).

[MÓR 96] MÓRA, M. et al. Motive processing and its role in rational reasoning. In: 1ST IBEROAMERICAN WORKSHOP ON ARTIFICIAL INTELLIGENCE AND DISTRIBUTED ARTIFICIAL INTELLIGENCE, 1996. Springer-Verlag, 1996.

[MÓR 97] MÓRA, M. et al. Affecting the stability of intentions. In: Costa, E., editor, 8TH PORTUGUESE CONFERENCE ON ARTIFICIAL INTELLIGENCE, 1997. Springer-Verlag, 1997.

[MÓR 98a] MÓRA, M. et al. BDI models and systems: Reducing the gap. In: SUBMITTED TO ATAL'98, 1998. Springer-Verlag, 1998.

[MÓR 98b] MÓRA, M. et al. Modelling agents with extended logic programa. In: INTERNATIONAL WORKSHOP ON ENGINEERING OF INTELLIGENT SYSTEMS, 1998. ICSC co., 1998.

[NAK 81] NAKASHIMA, H.; OHSAWA, I. Inference with mental situation. Umezono, Tsukuba, Ibaraki, Japan, 1981. Relatório técnico.

[QUA 95] QUARESMA, P. Unified logical programming approach to the abduction of plans and intentions in information-seeking dialogues. **Journal of Logic Programming**, [S.l.], v.24, n.1&2, 1995.

[QUA 97] QUARESMA, P. **Inferência de Atitudes em Dialogos**. Lisbon, Portugal:Universidade Nova de Lisboa, june, 1997. Tese de Doutorado.

[RAO 91] RAO, A. Modelling rational agents within a BDI-architecture. In: Fikes, R., editors, PROCEEDINGS OF THE KNOWLEDGE REPRESENTATION AND REASONING'91 (KR&R'91), 1991. **Proceedings...** San Mateo, CA.:Morgan Kauffman Publishers, 1991.

[RAO 96] RAO, A. Agentspeak(1): BDI agents speak out in a logical computable language. In: PROCEEDINGS OF THE EUROPEAN WORKSHOP ON MODELLING AUTONOMOUS AGENTS AND MULTI-AGENTS WORLDS 1996 (MAAMAW'96), 1996. **Proceedings...** Berlin, Germany:Springer-Verlag, 1996.

[RUS 95] RUSSEL, S. **Artificial Intelligence: A Modern Approach**. Prentice Hall, 1995.

[SCH 99] SCHILD, K. On the relationship between BDI logics and standard logics of concurrency. In: Müller, J.; Singh, M., editors, INTELLIGENT AGENTS V — PROCEEDINGS OF THE FIFTH INTERNATIONAL WORKSHOP ON AGENT THEORIES, ARCHITECTURES, AND LANGUAGES (ATAL-98), Lecture Notes in Artificial Intelligence. Springer-Verlag, Heidelberg, 1999.

[SEA 84] SEARLE, J. What is an intentional state? In: Dreyfuss, H., editors, HUSSERL, INTENTIONALITY AND COGNITIVE SCIENCE, v.42, p.213–261. 1984.

[SHO 93] SHOHAM, Y. Agent-oriented programming. **Artificial Intelligence**, [S.l.], v.60, p.51–92, 1993.

[SHO 94] SHOHAM, Y. Logics of mentao attitudes in ai. In: Lakemeyer, G., editors, FOUNDATIONS OF KNOWLEDGE REPRESENTATION AND REASONING. Springer-Verlag, 1994.

[SIC 92] SICHMAN, J. When can knowledge-based systems be called agents? In: PROCEEDINGS OF THE NINTH BRAZILIAN SYMPOSIUM ON ARTIFICIAL INTELLIGENCE (SBIA'92), 1992. [s.n.], 1992.

[SIN 94] SINGH, M. **Multiagent systems: a theoretical framework for intentions, know-how, and communications**. Heidelberg, Germany:Springer-Verlag, 1994. Lecture Notes in Artificial Intelligence (LNAI 799).

[SMI 91] SMITHERS, T. Taking eliminative materialism seriously: a methodology for autonomous system research. In: Varela, F., editors, PROCEEDINGS OF THE 1ST EUROPEAN CONFERENCE ON ARTIFICIAL LIFE, 1991. [s.n.], 1991.

[V.D 95] V.D., P. An emergent approach to systems of physical agentes. In: PROCEEDINGS OF THE AAAI SPRING SYMPOSIUM ON SOFTWARE ARCHITECTURES FOR PHYSICAL AGENTS, 1995. [s.n.], 1995.

[vL 96] VAN LINDER, B.; VAN DER HOEK, W. Formalizing motivational attitudes of agents: On preferences, goals, and commitments. In: Wooldridge, M.; Müller, J. P., editors, INTELLIGENT AGENTS II — PROCEEDINGS OF THE SECOND INTERNATIONAL WORKSHOP ON AGENT THEORIES, ARCHITECTURES, AND LANGUAGES (ATAL-95), v.1037 of **Lecture Notes in Artificial Intelligence**, p.17–32. Springer-Verlag, Heidelberg, 1996.

[WAI 94] WAINER, J. Yet another semantics of goals and goal priorities. In: PROCEEDINGS OF THE ELEVENTH EUROPEAN CONFERENCE OF ARTIFICIAL INTELLIGENCE (ECAI'94), 1994. **Proceedings...** Amsterdam, The Netherlands:[s.n.], 1994.

[WOO 95] WOOLDRIDGE, M. Agent theories, architectures and languages: a survey. In: PROCEEDINGS OF ECAI'94 WORKSHOP ON AGENT THEORIES, ARCHITECTURE AND LANGUAGES (ATAL'94), 1995. **Proceedings...** Amsterdam, NE:[s.n.], 1995. Lecture Notes in Artificial Intelligence (LNAI) 890.

[WOO 97] WOOLDRIDGE, M. Agents as a rorschach test: A response to franklin and graesser. In: PROCEEDINGS OF THE INTELLIGENT AGENTS III: AGENT THEORIES, ARCHITECTURES AND LANGUAGES, 1997. **Proceedings...** Budhapest, Hungary:[s.n.], 1997. Lecture Notes in Artificial Intelligence (LNAI 1193).