Paulo Quaresma

Inferência de atitudes em diálogos

Dissertação apresentada para obtenção do
Grau de Doutor em Engenharia Informática,
pela Universidade Nova de Lisboa,
Faculdade de Ciências e Tecnologia.

LISBOA
1997

Agradecimentos

Em primeiro lugar gostaria de agradecer ao meu orientador Doutor Gabriel Lopes pelo seu apoio e incentivo. De facto, durante a elaboração deste trabalho esteve sempre disponível para o analisar e para sugerir possíveis soluções para os problemas detectados. Por outro lado, sempre me deu o espaço necessário para seguir *à minha maneira* os caminhos que foram definidos. Por tudo isto, o meu obrigado.

Gostaria também de agradecer ao Prof. Doutor Luís Moniz Pereira pela sua disponibilidade para analisar, comentar e sugerir melhoramentos ao trabalho realizado. A utilização do ambiente de programação em lógica estendida que tem vindo a ser desenvolvido pelo seu grupo também foi um ponto fundamental na elaboração desta tese.

Um agradecimento especial aos colegas que, ao longo do tempo, me tiraram diversas dúvidas e comentaram e corrigiram este trabalho, em especial a Irene, o Sérgio, o Michael, o Damásio e o Alferes.

Agradeço, ainda, ao Departamento de Informática da Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa e ao Centro de Inteligência Artificial do Uninova pelo apoio dado durante estes anos.

A um outro nível, gostaria de agradecer à minha família, com um obrigado especial para o Pai Quaresma.

Finalmente, à Cati pela sua paciência e amor (e à Beatriz e à Francisca por, normalmente, me deixarem trabalhar à noite).

Resumo

Nesta tese proponho uma metodologia e um ambiente de trabalho que permitem a inferência das atitudes dos agentes participantes em diálogos. Estas inferências permitem a cada agente uma participação activa e inteligente em diversos tipos de diálogos, desde os diálogos cooperativos para obtenção de informação até aos diálogos não necessariamente cooperativos com diversos tipos de interlocutores.

O ambiente de trabalho proposto utiliza a programação em lógica estendida com negação explícita para a representação do conhecimento, sendo a inferência das diferentes atitudes dos agentes modelada através de um processo abdutivo. A semântica utilizada é basedada na semântica bem fundada com negação explícita (WFSX) desenvolvida por Pereira et al.

A principal inovação deste trabalho reside na integração, sob um ambiente de programação em lógica, com uma semântica bem definida, dos componentes necessários a uma correcta participação em diálogos: representação de conhecimento (incluindo a representação do conhecimento temporal), modelação de agentes, reconhecimento de actos de fala, inferência de atitudes, revisão de crenças, geração de planos e geração de actos de fala. O processo de integração da totalidade destes componentes sob um ambiente formal é uma característica inovadora em sistemas de processamento de diálogos e permite lidar com um conjunto bastante mais vasto de situações.

O sistema que proponho permite resolver os principais problemas existentes nos actuais sistemas de diálogo, nomeadamente o recurso a processos não formais para a inferência de atitudes, a representação e revisão dessas atitudes e a detecção e correcção de situações de ambiguidade e de erro nos diálogos.

O ambiente proposto poderá, ainda, ser utilizado como parte integrante de uma arquitectura de suporte a interacções entre agentes, recorrendo a módulos especializados que permitam o reconhecimento de actos de fala e a geração de frases em Língua Natural e recorrendo a outros módulos quando a interacção em Língua Natural não é necessária (sistemas multi-agentes).

Summary

A framework for supporting the inference of attitudes in dialogues is proposed. These inferences allow each agent to have an active and intelligent participation in several kinds of dialogues, namely cooperative information-seeking dialogues and non-cooperative dialogues.

The proposed framework uses extended logic programming to represent knowledge and to enable abductive reasoning in order to model the inference of attitudes. The used semantics is based on the Well Founded Semantics augmented with eXplicit negation (WFSX) from the work of Pereira et al.

The main innovation of this work is the integration under a logic programming framework with a specific semantics of several modules needed for a correct (natural and intelligent) participation in dialogues: knowledge representation, agent modelling, speech acts recognition, inference of attitudes, belief revision, plan generation, and speech acts generation. The integration of these modules is a new feature in dialogue processing systems and it allows us to handle a much wider range of situations.

The proposed system allows the resolution of the main problems detected in the previous dialogues systems, namely the use of non-formal processes to infer attitudes, the representation, revision and updating of the attitudes, and the detection and correction of dialogue error situations.

The proposed framework may also be used as an engine for a more global arquitecture supporting agents interaction (multi-agent systems). Whenever communication in Natural Language is necessary, it will use specialized modules to recognize speech acts and to generate utterances in Natural Language.

Índice de matérias

Índice de figuras

Índice de tabelas

Glossário Português-Inglês

Abdução ponderada	Weighted abduction
Cálculo de Eventos	Event Calculus
Cálculo de Situações	Situation Calculus
Coerente (procedimento)	Sound (procedure)
CFXSM (Modelo Estável Não Contraditório)	Contradiction Free eXtended Stable Model
Derivação linear selectiva para programas extendidos	Selected Linear resolution for eXtended programs (SLX)
Descendente	Top-down
Diálogos cooperativos para busca de informação	Cooperative information seeking-dialogues
Hipótese do mundo fechado	Closed World Assumption
Literais por omissão ou supletivos	Default literals
Modelo bem fundado	Well founded model
Modelo parcial estável	Partial Stable Model (PSM)
Pergunta de resposta Sim-Não	YN-Question
Pergunta Quem-Qual	Wh-Question
Persistência	Commitment
Programa admissível	Allowed Program
Programa com termos limitados	Bounded term program
Programa instanciado	Grounded Program
Programação em lógica estendida com negação explícita	Logic programming extended with explicit negation
Programas Abdutivos em Lógica	Abductive Normal Logic Programs
PCFXSM (Modelo Estável Não Contraditório Preferido)	Preferred Contradiction Free eXtended Stable Model
Raciocínio por omissão ou supletivo	Default reasoning
Raciocínio revogável	Defeasible reasoning
Resolução de Problema(s)	Problem Solving
Semântica bem fundada	Well founded semantics
Semântica bem fundada com negação explícita	WFSX (Well founded semantics with explicit negation)
Semântica de completude de predicados	Predicate completion semantics
Semântica de conjuntos-resposta	Answer-set semantics

Glossário Inglês-Português

Abductive Normal Logic Programs	Programas Abdutivos em Lógica
Allowed Program	Programa admissível
Answer-set semantics	Semântica de conjuntos-resposta
Bounded term program	Programa com termos limitados
Closed World Assumption	Hipótese do mundo fechado
Commitment	Persistência
Cooperative information seeking-dialogues	Diálogo cooperativo para busca de informação
CFXSM (Contradiction Free eXtended Stable Model)	Modelo Estável Não Contraditório
CWA (Closed World Assumption)	Hipótese do mundo fechado
Defeasible reasoning	Raciocínio revogável
Default literals	Literais por omissão ou supletivos
Default reasoning	Raciocínio por omissão ou supletivo
Event Calculus	Cálculo de Eventos
Grounded Program	Programa instanciado
Logic programming extended with explicit negation	Programação em lógica estendida com negação explícita
Partial Stable Model (PSM)	Modelo parcial estável
Predicate completion semantics	Semântica de completude de predicados
Problem solving	Resolução de Problema(s)
PCFXSM (Preferred Contradiction Free eXtended Stable Model)	Modelo Estável Não Contraditório Preferido
Situation Calculus	Cálculo de Situações
SLX (Selected Linear resolution for eXtended programs)	Derivação linear selectiva para programas extendidos
Sound (procedure)	Coerente (procedimento)
Top-down	Descendente
Weighted abduction	Abdução ponderada
Well founded model	Modelo bem fundado
Well founded semantics	Semântica bem fundada
WFM (Well founded model)	Modelo bem fundado
WFSX (Well founded semantics with explicit negation)	Semântica bem fundada com negação explícita
Wh-Question	Pergunta Quem-Qual
YN-Question	Pergunta de resposta Sim-Não

Capítulo 1

Introdução

Neste capítulo apresento a motivação para esta tese, bem como os objectivos que, com o trabalho desenvolvido, pretendo atingir. É feita uma descrição dos principais sistemas de reconhecimento de planos e intenções em diálogos, focando os seus pontos fortes e as suas desvantagens. Posteriormente, descrevo a abordagem proposta para a inferência de atitudes em diálogos, destacando os problemas que a metodologia apresentada permite resolver.

Finalmente, apresento a organização da tese, descrevendo o conteúdo dos seus capítulos, bem como as suas dependências.

1.1 Motivação da tese

Uma participação inteligente em diálogos requer que cada agente participante tenha capacidade para inferir as atitudes (crenças, intenções e objectivos) dos seus interlocutores, de modo a poder actualizar as suas próprias atitudes, planear as suas acções e realizar essas acções.

O diálogo seguinte (adaptado de [LA87]), passado numa estação de caminho de ferro, tipifica o tipo de problemas que o sistema proposto tenta solucionar:

Passageiro: O comboio das 8:30?

Empregado: Para o Porto? Sai da linha 5.

Passageiro: Não. O de Leiria.

Empregado: Chega na linha 7, mas está 15m atrasado.

Um sistema computacional para desempenhar o papel de empregado, e participar activamente num diálogo deste tipo, tem de ser capaz de resolver os seguintes problemas:

- Representar adequadamente o domínio da aplicação (capítulo 2);

  Este aspecto está directamente relacionado com a necessidade de representar as acções passíveis de serem realizadas pelo sistema e os eventos que podem ocorrer no mundo circundante. No diálogo apresentado, seria necessário representar as acções do domínio (por exemplo, fazer uma viagem de comboio, comprar um bilhete, esperar alguém, etc.) e os eventos relevantes (actos de fala, chegadas e partidas de comboios).

- Representar adequadamente no tempo os eventos ocorridos e as acções associadas, bem como as atitudes (crenças, intenções e objectivos) que o sistema possui (capítulo 2);

  A representação temporal dos eventos e das atitudes do sistema permite a construção do modelo mental do agente-sistema, incluindo a sua *interpretação* do mundo que o rodeia.

- Modelar o seu comportamento e o dos outros agentes (capítulo 3);

  A correcta modelação dos agentes é um aspecto crucial do processo de participação em diálogos. Fundamentalmente, é necessário representar as atitudes próprias e alheias (crenças, objectivos e intenções) e modelar o tipo de comportamento (sinceridade, credulidade, cooperatividade, actividade). No exemplo apresentado, o empregado deverá ser um agente cooperativo, sincero, reactivo e crédulo.

- Reconhecer as acções dos outros interlocutores (nomeadamente os actos de fala), as suas pré-condições e os seus efeitos (capítulo 2);

  Num diálogo, os eventos estão, normalmente associados a actos de fala que transmitem informação. Como consequência, existe a necessidade de reconhecer esses actos, utilizando a informação que eles veiculam de modo a actualizar o estado mental do agente-sistema computacional. No diálogo anterior, o empregado deverá reconhecer o pedido do passageiro (*request*) de uma determinada informação (*informref*) sobre comboios. Posteriormente, deverá ter a capacidade de reconhecer que a sua resposta não fora correcta e que existe um novo pedido de informação.

- Inferir atitudes (crenças, objectivos e intenções) próprias e alheias (capítulo 4);

  Com base nos actos de fala dos outros agentes, deverá ser inferido um conjunto de novas atitudes consistentes com os eventos ocorridos anteriormente e com o estado mental do agente-sistema computacional. No exemplo apresentado, após a primeira frase do passageiro, o empregado deverá inferir a crença de que o passageiro deseja ser informado sobre uma dada propriedade de um dado comboio que chega ou parte às 8:30; em face desta crença, o empregado adopta a crença de que o interesse é sobre um comboio que parte (pedido de informação mais provável); e, finalmente, o empregado adopta a intenção de informar o passageiro sobre a informação que crê que ele deseja.

- Actualizar e rever as suas atitudes (capítulo 5);

  Tendo em conta os novos actos de fala dos seus interlocutores, poderá ser necessário o agente-sistema actualizar as suas atitudes, cancelando atitudes anteriores e iniciando novas atitudes. De notar que o processo de inferência de atitudes poderá provocar a necessidade de efectuar uma revisão de atitudes, devido à possível criação de contradições no modelo mental do agente-sistema (crenças contraditórias, por exemplo). Neste caso, será necessário efectuar a revisão do modelo mental do agente, obtendo um modelo não contraditório. No exemplo apresentado, após a segunda frase do passageiro, será necessário cancelar a crença de que o passageiro pretendia informações sobre um comboio que partia e iniciar a crença de que, pelo contrário, pretende informações sobre o comboio que chega de Leiria.

- Planear acções (capítulo 5);

  A partir do novo estado mental do agente-sistema, deverão ser planeadas as acções que permitam a satisfação dos objectivos existentes. No caso apresentado, o empregado deverá criar o plano de informar o passageiro sobre o comboio em causa.

- Gerar actos de fala (capítulo 5);

  Os planos construídos deverão ser gerados. No exemplo apresentado, o empregado informará o passageiro sobre a propriedade que considera que foi requerida. Note-

se que, no entanto, a geração de frases em Língua Natural não será analisada em detalhe no âmbito deste trabalho.

- Raciocinar (capítulo 4 e 5).

  O processo de participação em diálogos implica a capacidade de efectuar diversos tipos de raciocínio. De facto, o agente-sistema computacional necessita de efectuar raciocínio por omissão quando assume que, por omissão, o pedido de informação é sobre um comboio que parte. O agente-sistema necessita, também, de possuir capacidade de efectuar raciocínio abdutivo quando planeia as acções necessárias para satisfazer os seus objectivos.

A análise dos mecanismos envolvidos nestes diversos processos tem vindo a ser objecto de estudo de diferentes áreas da Ciência, desde a psicologia cognitiva até à inteligência artificial, passando pela linguística computacional. De facto, a necessidade de recorrer a métodos de representação de conhecimento, de modelação de agentes, de inferência e revisão de atitudes, de planeamento e de execução dos planos construídos e de raciocínio tem provocado a necessidade de reconstruir e integrar diversas teorias criadas anteriormente com o objectivo de resolver problemas específicos e parciais.

Nesta tese proponho uma abordagem com base na programação em lógica que permite o reconhecimento, a inferência e a revisão das atitudes (crenças, intenções, objectivos) do agente sobre si próprio e sobre os seus interlocutores e o planeamento e a geração de actos de fala. A abordagem integra, sob um ambiente de programação em lógica, a representação de conhecimento, a modelação de agentes, a inferência de atitudes, o planeamento e a realização de actos de fala. Um protótipo foi desenvolvido utilizando o sistema proposto por Pereira et al. ([ADP95, Alf93, DNP94]). No capítulo 7 apresento em detalhe os aspectos relacionados com a construção do protótipo construído.

Na próxima secção é apresentado o modo como os principais sistemas existentes abordam os problemas levantados por este tipo de diálogos, realçando os problemas que não foram resolvidos e para os quais apresento uma solução mais robusta.

1.2 Sistemas para a inferência de atitudes em diálogos

Nesta secção são descritos alguns dos sistemas que têm vindo a ser desenvolvidos com o objectivo de suportar o reconhecimento de planos e de atitudes em discursos e em diálogos.

Os primeiros trabalhos a relacionar planeamento e discurso foram apresentados por Hobbs e Evans ([HE80]), por Cohen e Perrault ([CP79]) e por Allen e Perrault ([AP80]). A ideia base era integrar técnicas de planeamento desenvolvidas em Inteligência Artificial com a teoria de actos de fala desenvolvida por Austin e Searle ([Aus62]). Os sistemas propostos utilizavam o modelo clássico de representação de planos STRIPS ([FN71]) e NOAH ([Sac77]). Neste modelo cada plano é definido por uma sequência de acções e cada acção é composta por um cabeçalho, pré-condições, restrições e efeitos. Em cada

instante, cada acção poderá ser executada se as pré-condições se verificarem e as restrições de integridade não forem violadas. Com base numa biblioteca de acções e de planos, e utilizando algumas regras heurísticas de inferência é possível efectuar a inferência de planos.

Um dos principais problemas destas abordagens, e que o trabalho descrito nesta tese soluciona, é a inexistência de uma semântica clara para o processo de inferência. De facto, estes sistemas não suportam uma representação que possua uma semântica definida e procedimentos de prova coerentes e completos. O comportamento desejado é modelado através do recurso a regras incluídas no próprio processo de inferência. No trabalho apresentado nesta tese esta estratégia não é utilizada, dado que, ao utilizar-se um ambiente de programação em lógica, ir-se-á obter um sistema de inferência com uma semântica bem definida (semântica bem fundada). As regras de comportamento são modeladas através da utilização de regras de programação em lógica e não são parte integrante da máquina de inferência.

Um outro problema com as abordagens clássicas é a sua incapacidade para modelar raciocínios não-monótonos que permitem lidar com informação incompleta. Na proposta desta tese, o raciocínio não-monótono é modelado através da programação em lógica estendida com negação explícita.

O trabalho realizado por Allen foi, posteriormente, estendido por Litman ([Lit85]) para suportar diálogos cooperativos de busca de informação. De facto, em diálogos deste tipo, a mesma abordagem utilizada por Allen para o reconhecimento de planos implícitos em discursos pode ser utilizada, dado que um agente interveniente num diálogo pode efectuar a geração dos seus actos de fala com base nos actos de fala já realizados e nos seus próprios objectivos.

No entanto, o mesmo tipo de críticas apresentado para o trabalho de Allen também se aplica ao trabalho de Litman em diálogos, dado continuar a ter como base processos em que não existe uma semântica bem definida e em que não é possível modelar raciocínios não-monótonos. Além destes factores, esta abordagem não tem em conta a necessidade de modelar o estado mental do agente-sistema, nomeadamente a necessidade de representar as suas atitudes em relação a si próprio e aos agentes que o rodeiam.

Uma outra abordagem para reconhecimento de planos implícitos em diálogos foi utilizada por Pollack ([Pol86]). Um dos pontos essenciais desta abordagem é a constatação de que são necessários modelos mais sofisticados de representação de atitudes (crenças e intenções) para suportar e resolver alguns dos problemas originados num processo de diálogo. Nomeadamente, no caso de diálogos em que um dos participantes demonstra ter planos incorrectos é necessário recorrer a modelos mais poderosos de representação e revisão de atitudes, de modo a possibilitar a revisão do plano que permite satisfazer os objectivos desejados. A abordagem de Pollack interpreta os planos como sendo atitudes mentais e o processo de reconhecimento de planos como sendo um processo de inferir as atitudes dos diversos agentes. No seu trabalho inicial, Pollack recorria a processos não formais de representação de conhecimento pelo que os resultados obtidos careciam também

de semânticas bem definidas. Posteriormente, em conjunto com Appelt ([AP92]), Pollack propôs um sistema que utilizava a abdução ponderada para a inferência das atitudes, e com Konolige [KP93] propôs a utilização de um sistema de argumentação para suporte do processo de inferência.

Embora estes sistemas (abdução ponderada e sistema de argumentação) já possuam uma representação formal, têm como lacuna principal a inexistência de uma representação temporal adequada para permitir inferir e rever as atitudes dos agentes. Com efeito, a metodologia utilizada não integra teorias de representação temporal e, como tal, não consegue suportar diálogos em que seja necessário rever estados mentais, mantendo, simultaneamente, conhecimento sobre os estados anteriores.

A abordagem proposta nesta tese tenta solucionar este problema e integrar a visão dos planos como atitudes mentais, na sequência da proposta de Pollack, com a necessidade de representar e raciocinar sobre os diversos tipos de acções (na sequência da proposta de Allen), incluindo uma representação temporal baseada no Cálculo de Eventos. Deste modo, consigo solucionar grande parte dos problemas de cada uma das abordagens de Pollack, de Litman e de Allen.

Grosz e Sidner ([GS90]) utilizaram também a noção de planos como conjuntos de atitudes (crenças e intenções) em diálogos em que diversos agentes colaboram entre si. No entanto, e mais uma vez, esta abordagem possui as mesmas lacunas da abordagem de Pollack e também não permite raciocínios sobre a localização temporal dos estados mentais, sobre o seu início e sobre o seu fim, sobre a sua persistência e a relação com outros eventos ocorridos no mundo que rodeia o agente-sistema computacional.

Uma abordagem distinta é a seguida por Carberry ([Car85, Car88]) no seu trabalho, em que propõe um sistema que suporta diálogos cooperativos para busca de informação. O modelo é baseado numa árvore em que os nós são objectivos a satisfazer e que têm associados planos para serem satisfeitos. Carberry demonstra que o sistema proposto permite suportar alguns tipos de diálogos complexos, em que existem interrogações e frases mal formadas. Posteriormente, Lambert ([LC92]) e Ramshaw ([Ram89]), em trabalhos distintos, estendem o modelo de Carberry para um modelo com três níveis (domínio, resolução do problema e discurso). No entanto, e embora as abordagens propostas permitam resolver alguns dos problemas que surgem em diálogos, não estão completamente formalizadas e não integram, sob o mesmo ambiente, os diversos componentes fundamentais à participação em diálogos (representação temporal, representação e revisão de atitudes e reconhecimento, planeamento e realização de actos de fala).

Finalmente, uma das abordagens que estará mais relacionada com a seguida nesta tese é a resultante do trabalho de Ferguson ([Fer95]) em que é proposto um sistema de raciocínio não-monótono sobre planos que integra tempo, estados e acções. A representação de planos é efectuada com base num sistema de raciocínio revogável em que são construídos argumentos com base em regras revogáveis. Os planos são vistos como argumentos em que uma certa sequência de acções, sob determinadas condições, atingirá determinados objectivos. O planeamento é um processo de construção de argumentos e de

contra-argumentos. Esta abordagem tem como característica fundamental a apresentação de um sistema formal de raciocínio para planeamento com vários agentes. No entanto, ao focar essencialmente as acções do domínio, não considera as componentes específicas dos actos de fala dos agentes e das suas atitudes. O trabalho de Ferguson tem, assim, como principal lacuna a não existência das componentes relativas à interpretação dos actos de fala e à sua interligação com os estados mentais. O seu objectivo é a construção de um sistema formal de planeamento multiagente e não a construção de um sistema que permita um processo de participação activa em diálogos. A abordagem que proponho nesta tese tenta suprir estas falhas, integrando sob um ambiente de programação em lógica todos estas componentes: acções (incluindo os actos de fala), estados (incluindo as atitudes) e a representação temporal.

Na próxima secção será apresentada a metodologia proposta nesta tese para suportar a inferência de atitudes em diálogos (crenças, objectivos e intenções) e o processo de planeamento e geração de actos de fala.

1.3 A abordagem proposta

Conforme foi descrito na secção anterior, a maioria das abordagens existentes têm o grande problema de não recorrerem a linguagens com semânticas bem definidas que permitem a existência de processos formais operacionais que suportem a participação activa e intencional de sistemas computacionais em diálogos. As abordagens que definem um processo formal não são suficientemente abrangentes, não integrando componentes fundamentais como a representação temporal explícita dos eventos e das atitudes do agente-sistema ([Fer95, AP92]).

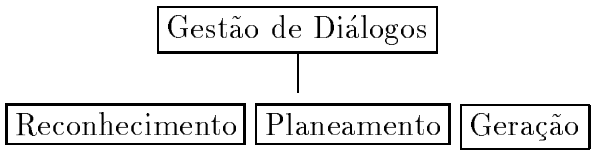

Figura 1.1: Arquitectura proposta

Conforme se pode observar na figura 1.1, a abordagem proposta nesta tese integra as diversas componentes necessárias ao processamento de diálogos: o reconhecimento dos actos de fala; a inferência e a revisão de atitudes e o planeamento das acções a realizar; e a geração dos actos de fala.

O ambiente utilizado para a representação e integração destas componentes é a programação em lógica estendida com negação explícita. A semântica utilizada é a semântica bem fundada com negação explícita (WFSX[1]), baseando-se no trabalho de Pereira et al.

[1]Well Founded Semantics for eXtended programs

([ADP95]). Esta semântica para programas em lógica estendida com negação explícita foi adoptada devido essencialmente às suas propriedades (simplicidade, cumulatividade, racionalidade, relevância e avaliação parcial — ver [AP96] para uma análise detalhada) e devido à existência de procedimentos de prova descendentes e ascendentes.

O ambiente de programação em lógica adoptado permite modelar raciocínios não-monótonos, como por exemplo os raciocínios abdutivos e por omissão, dando uma semântica declarativa e operacional aos programas em lógica. Além destes pontos permite, ainda, a experimentação do trabalho desenvolvido, dado existir um demonstrador de teoremas para programas em lógica com semântica bem fundada ([DNP94]).

Como teoria de representação de eventos utilizou-se uma versão alterada do Cálculo de Eventos, a partir do trabalho de [Esh88, Mis91, DMB82]. A versão proposta permite a representação de eventos não atómicos (com duração no tempo), bem como a existência de simultaneidade de eventos. Deste modo é possível representar os eventos relativos às acções de diálogo e às acções referidas nesses mesmos diálogos, bem como efectuar raciocínio temporal sobre a informação representada.

As acções, tanto as do domínio como as dos diversos actos de fala, são representadas recorrendo a uma linguagem de representação de acções, A_{EC}, derivada da linguagem proposta por Gelfond e Lifschitz (linguagem A - ver [GL92a]). As alterações propostas nesta tese à linguagem A permitem representar acções mais complexas do que as da proposta original de Gelfond e Lifschitz, tendo sido definidos os modelos e a semântica desta linguagem A_{EC}. Com base na linguagem aqui proposta foi definida uma conversão para programas em lógica estendidos com a negação explícita e foi provada a completude e coerência dessa transformação (apêndice A). A opção de utilizar uma linguagem de descrição de acções geral e independente de um formalismo de representação temporal tem como vantagem a possibilidade de utilizar um outro formalismo (o cálculo de situações em vez do cálculo de eventos, por exemplo), bem como a possibilidade de utilizar resultados teóricos já existentes que relacionam diferentes abordagens. Dado que Kartha ([Kar93]) provou no seu trabalho a completude e a coerência de traduções da linguagem A para os formalismos de Pednault, Reiter e Baker e, como provo nesta tese a equivalência entre os modelos da linguagem A e da linguagem proposta, A_{EC}, é possível relacionar o formalismo aqui proposto com os formalismos acima citados. Renwei Li e Pereira [LP96a] utilizaram uma abordagem semelhante para estender a linguagem A de Gelfond e Lifschitz para suportar acções concorrentes e a observação de proposições. No seu trabalho provaram a coerência e completude de uma tradução para programas abdutivos em lógica (*abductive normal logic programs*). A abordagem seguida nesta tese tem algumas semelhanças com este trabalho sendo, no entanto, a tradução efectuada para programas em lógica estendida com a negação explícita e com semântica bem fundada.

Os actos de fala subjacentes às frases que constituem os diálogos foram definidos com base nos trabalhos de Allen, Cohen, Perrault e Levesque ([AP80, CP79, CL90a]). Estes actos de fala foram no entanto descritos utilizando a linguagem de representação de acções e, posteriormente, traduzidos para programas em lógica. A representação dos actos de fala

foi alterada de modo a ter em conta os pré-requisitos e os efeitos nos diversos interlocutores em termos de atitudes (crenças, objectivos e intenções). Foi ainda necessário tomar em consideração o efeito do modelo dos utilizadores na representação utilizada, dado que, por exemplo, uma acção de informar tem um efeito diferente consoante o destinatário seja crédulo ou não.

As acções passíveis de serem executadas num dado domínio de aplicação foram descritas utilizando a linguagem de descrição de acções referida anteriormente.

As diversas atitudes representativas do estado mental de cada agente, sejam elas crenças, intenções ou objectivos, foram representadas utilizando a linguagem referida (A_{EC}) e têm como base o trabalho desenvolvido previamente por diversos investigadores, nomeadamente o trabalho mais recente de [Pol86, AP92].

Como consequência de todas as acções e mudanças de estado terem sido caracterizadas através de uma linguagem independente do qualquer formalismo temporal, obteve-se um sistema que poderá ser mais facilmente comparado com outros já existentes e que poderá ser utilizado em diversos ambientes de raciocínio temporal.

De modo a integrar todas estas componentes, foi necessário representar o modelo dos agentes participantes nos diálogos. A representação que proponho permite modelar diversos tipos de agentes, cruzando características tão distintas como a cooperactividade vs não cooperactividade, pró-actividade vs reactividade, credulidade vs cepticismo, ou sinceridade vs falsidade. Através da separação entre os modelos dos agentes e o processo de inferência das atitudes é possível simular no sistema proposto comportamentos distintos, pela simples alteração dos modelos subjacentes a cada agente. Esta característica é inovadora em sistemas de processamento de diálogos e permite a construção de sistemas bastante mais poderosos. Esta representação foi efectuada também utilizando a linguagem de descrição referida anteriormente (A_{EC}).

Com base no modelo dos utilizadores e nos actos de fala realizados é possível efectuar a inferência das atitudes válidas para cada agente participante em diálogos. Nesta tese proponho que a inferência seja efectuada recorrendo ao cálculo do modelo bem fundado do programa em lógica que modela o estado mental dos agentes. O processo de inferência utiliza raciocínio dedutivo e abdutivo para a obtenção das atitudes válidas em cada instante de tempo (processo descrito em detalhe no capítulo 4).

A partir do estado mental de cada agente é possível efectuar o planeamento das acções a realizar. Este processo é feito com base nas intenções do agente válidas em cada instante de tempo e abduzindo as acções que poderão levar à satisfação dos seus objectivos. A metodologia que proponho nesta tese modela, portanto, o planeamento através da abdução das acções que levarão à concretização dos objectivos requeridos.

Finalmente, a geração dos actos de fala é efectuada com base no planeamento efectuado e após a selecção dos planos existentes. Posteriormente, a partir dos actos de fala seleccionados, poderão ser geradas as frases em Língua Natural correspondentes. O aspecto da geração e da análise da Língua Natural não é tratado nesta tese. De facto, a ligação entre as diversas teorias semânticas e pragmáticas que têm vindo a ser desenvolvidas,

nomeadamente a semântica de situações [BP83, Coo93, CG94], a teoria de representação do discurso [KR93, AL94] e a semântica dinâmica [GS86a], só recentemente começaram a analisar o diálogo como área de aplicação. No entanto, convém referir que todas as abordagens adoptam um ponto de vista de uma terceira pessoa, reduzindo o problema a um texto descritivo do diálogo. Nesta área há trabalho que tem vindo a ser desenvolvido, nomeadamente o trabalho de Ginzburg [Gin95, J.G97], de Asher [AL97], Poesio [Poe97], e também trabalho desenvolvido no DI/FCT/UNL por Lopes et al. [LQR97]. Só agora começa a ser tratado o problema da ligação entre as diversas teorias semânticas com a teoria dos actos de fala, subjacente ao trabalho desta tese. Neste sentido, optou-se por não abordar nesta tese esta problemática, e tratar essencialmente das questões relacionadas com a participação em diálogos, simultaneamente intencional e, tanto quanto possível, natural, procurando resolver problemas na área da transferência de atitudes, produção de novas atitudes e término de atitudes antigas, do planeamento do comportamento e da concretização do planeado.

Como súmula, poder-se-á referir que a abordagem proposta nesta tese pretende integrar, sob um sistema de programação lógico formal e com uma semântica bem definida, as diversas componentes necessárias a uma participação activa e intencional em diálogos.

1.4 Principais contribuições da tese

Na secção anterior foi descrita a abordagem proposta para modelar o processo de participação de agentes em diálogos.

A abordagem proposta integra diversos aspectos inovadores em relação a abordagens anteriores, que lhe permitem resolver um conjunto vasto de problemas que surgem tradicionalmente em diálogos: informação incompleta, ambiguidade e situações de erro e planeamento incorrecto.

Sumarizando, as principais contribuições inovadoras desta tese para uma teoria de participação em diálogos são:

- Utilização de um ambiente de programação em lógica com semântica bem fundada para a modelação do processo de participação em diálogos;
- Extensão do Cálculo de Eventos para suportar eventos não instantâneos;
- Definição de uma linguagem de representação de acções e estabelecimento das suas relações com outros formalismos já existentes;
- Definição dos actos de fala com base nas atitudes dos agentes;
- Definição do modelo lógico dos agentes;
- Capacidade de efectuar inferências e de actualizar e rever as atitudes dos agentes após cada acto de fala, de acordo com o estado mental dos agentes representado

através do modelo bem fundado do programa em lógica estendida com negação explícita;

- Capacidade de planear e gerar actos de fala com base no modelo bem fundado que representa o estado mental dos agentes.

Cada um destes pontos será abordado em detalhe nos capítulos seguintes.

Na secção seguinte far-se-á a apresentação da organização desta tese, bem como as dependências existentes para a sua leitura.

1.5 Organização da tese

- No capítulo 2 é apresentada a metodologia utilizada para representar o conhecimento necessário à inferência das atitudes do agente-sistema participante em diálogos.

  O capítulo é constituído por uma descrição do sistema de suporte à representação do conhecimento, pela descrição da metodologia de representação temporal, de acções do domínio, de actos de fala e das atitudes dos agentes.

  É efectuada uma descrição detalhada das teorias utilizadas, bem como das alterações efectuadas de modo a adequá-las ao ambiente de programação em lógica proposto e aos requisitos do sistema de participação em diálogos (por exemplo, a existência de eventos concorrentes e não instantâneos).

- O capítulo 3 descreve o modo como é efectuada a modelação do agente-sistema computacional participante em diálogos.

  O processo utilizado permite modelar agentes com diferentes características, desde o agente cooperativo que pretende satisfazer os objectivos dos seus interlocutores, até aos agentes mentirosos que pretendem enganá-los, passando pelos agentes activos/não activos que apresentam posturas distintas durante os diálogos.

  Neste capítulo pretendo demonstrar as capacidades de representação do sistema proposto, em comparação com os sistemas já existentes, e que possibilitam a simulação de ambientes multiagentes com diferentes características.

- No capítulo 4 é descrito o processo de inferência e da actualização das atitudes dos diversos agentes.

  A inferência é efectuada com recurso a diversos tipos de raciocínio, nomeadamente o raciocínio abdutivo e o dedutivo. Este processo poderá levar à necessidade de efectuar revisão das atitudes existentes, devido à introdução de inconsistências pelo processo de abdução.

  Neste capítulo pretende-se demonstrar que o processo utilizado para suportar as inferências de atitudes permite lidar com um conjunto bastante vasto de situações que ocorrem em diálogos.

O capítulo inclui exemplos de diálogos tipo, realçando os problemas existentes e o modo como são resolvidos pelo sistema proposto.

- O capítulo 5 faz a integração de todas as ferramentas descritas nos capítulos anteriores de modo a permitir um processo de participação dos diversos agentes em diálogos.

  Assim, é descrito o modo com foi efectuada a integração do processo de inferência de atitudes, com o processo de modelação dos agentes e com a representação de conhecimento. Apresenta, ainda, o processo de planeamento e geração de actos de fala realizado com base nas atitudes inferidas. Neste capítulo é abordada a necessidade de efectuar a revisão do modelo de um agente sempre que seja introduzida uma atitude contraditória com as já existentes.

  É dada uma especial relevância aos resultados obtidos, aos problemas resolvidos e aos problemas em aberto.

- No capítulo 6 são apresentados exemplos de aplicação da teoria desenvolvida.

  O primeiro exemplo pertence ao domínio de diálogos cooperativos sobre comboios. Estes exemplos têm servido de base a vários sistemas de diálogos ([Lit85, GAT93, HA95]) e permitem aferir da capacidade do sistema proposto para resolver diversos tipos de problemas típicos em sistemas de gestão de diálogos.

  O segundo exemplo representa uma situação em que diversos agentes com diferentes características interagem entre si com vista à satisfação de determinados objectivos. Nesta situação existem acções que não são actos de fala e que podem provocar alterações nas atitudes dos agentes e afectar o seu comportamento futuro. Este exemplo é importante, dado demonstrar a capacidade do sistema em efectuar planeamento das acções a realizar, num ambiente com agentes heterogéneos e em que existem eventos externos que podem condicionar o modelo mental dos agentes.

- No capítulo 7 é descrito o desenvolvimento e a construção efectuada de um protótipo do sistema proposto.

  Como ambiente de base utilizou-se o protótipo desenvolvido por Pereira et al. ([PAA94, Alf93, DNP94]) no âmbito da programação em lógica estendida com semântica bem fundada (WFSX).

  Neste capítulo são apresentados os programas utilizados para a aplicação das teorias referidas nos capítulos anteriores.

  Para cada um dos programas em lógica utilizados são apresentados os fundamentos teóricos que suportam a correcção dos resultados obtidos. Em algumas situações estes fundamentos teóricos são apresentados em apêndice.

- Finalmente, no capítulo 8 são apresentadas as conclusões do trabalho realizado.

É dado um especial realce às características do sistema proposto, aos resultados obtidos e às principais inovações propostas nesta tese.

Na parte final deste capítulo é efectuada uma análise aos problemas existentes no sistema proposto e são apontados caminhos possíveis para a sua efectiva resolução.

- Os apêndices A e B apresentam provas de alguns teoremas referidos na tese.

  Nomeadamente é apresentada a relação entre a linguagem de representação de acções proposta (linguagem A_{EC}) e linguagem A e a prova de completude e coerência da transformação da linguagem A em cálculo de Eventos.

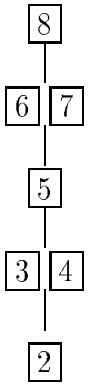

Figura 1.2: Dependências dos capítulos da tese

Trajectos de leitura

Na figura 1.2 são apresentadas as dependências existentes entre os diversos capítulos desta tese. Assim, o capítulo 2 apresenta a base do trabalho desenvolvido pelo que é requisito para a leitura de qualquer outro capítulo. Os capítulos 3 e 4 podem ser lidos por qualquer ordem, dado a sua independência lógica. O capítulo 5 mostra a integração das teorias apresentadas anteriormente e, como tal, tem como pré-requisito a leitura de todos os capítulos anteriores. Os capítulos 6 e 7 são independentes, dado apresentarem exemplos de aplicação da teoria e detalhes da construção do protótipo. Têm, no entanto, como requisito a leitura do capítulo 5. Finalmente, o capítulo 8 deverá ser lido no final, embora a dependência dos capítulos 6 e 7 seja somente aconselhada.

Capítulo 2

Representação de Conhecimento

Neste capítulo apresento a abordagem utilizada para representar o conhecimento necessário à inferência de atitudes dos diversos agentes participantes em diálogos. Este processo permitirá, também, suportar o processo de raciocínio não-monótono sobre esse mesmo conhecimento e servir de base a uma participação efectiva e inteligente em diálogos.

No início do capítulo é efectuada uma descrição do ambiente de programação em lógica utilizado, sendo posteriormente descrita a metodologia de representação de eventos, de acções e de atitudes. Para cada uma das teorias descritas é efectuada uma apresentação geral onde são salientadas as características inovadoras deste trabalho

Finalmente, na última secção, será efectuado uma análise global à abordagem proposta.

2.1 Introdução

O processo de inferência de atitudes dos agentes participantes em diálogos, sejam elas as suas crenças, as suas intenções de realizar acções ou os seus objectivos a satisfazer, necessita, como suporte, de uma representação de conhecimento que contemple a representação adequada dos eventos que possam suceder, estejam a suceder, ou tenham sucedido no tempo, nomeadamente os actos de fala dos diversos agentes e a representação das acções passíveis de serem realizadas nos diversos domínios de aplicação. Além disso, é necessária a capacidade de modelar os agentes, representando as suas atitudes e o processo de raciocínio utilizado para inferir novas atitudes e planear as acções a realizar.

Neste capítulo serão apresentados os mecanismos propostos para efectuar a representação do conhecimento necessário ao processo de participação em diálogos.

Assim, serão abordados neste capítulo os seguintes pontos:

- Ambiente geral utilizado

  Como ambiente de suporte à representação de conhecimento e ao processo de raciocínio necessário a uma efectiva e inteligente participação em diálogos foi utilizado a programação em lógica estendida com a negação explícita com a semântica bem fundada para programas em lógica estendida WFSX ([AP96]). Este ambiente tem como principais vantagens ser adequado a um processo de raciocínio não-monótono (necessário a um sistema de diálogos, conforme exemplo apresentado no capítulo 1), possuir procedimentos de derivação coerentes e completos para determinados tipos de programas em lógica e permitir a revisão de programas contraditórios (ver secção seguinte). Como consequência destas características, este ambiente permite ultrapassar o problema de alguns dos sistemas já existentes que não possuem processos formais de raciocínio (ver a análise efectuada no capítulo anterior aos sistemas de Litman e Allen, Carberry e Pollack).

- Representação de acções, incluindo os actos de fala

  As acções, sejam elas actos de fala (informar, requerer, perguntar) ou acções específicas do domínio necessitam ser representadas adequadamente. Para tal, é necessário representar as suas pré-condições e os seus efeitos. Na secção 2.4 é proposta uma linguagem de representação de acções A_{EC} e é apresentada uma tradução para programação em lógica estendida. Esta linguagem tem como vantagem adicional a possibilidade de serem efectuadas comparações com sistemas já existentes de representação de acções (linguagem A [GL92a]).

- Representação de eventos

  Os eventos são instâncias de acções associadas a intervalos de tempo, ou seja, são acções que ocorrem num determinado intervalo de tempo. É necessário representar no ambiente de programação em lógica estendida os diversos eventos que sucedem

no tempo. Estes eventos podem ser associados a actos de fala ou a acções do domínio (realizadas pelo agente-sistema computacional a modelar ou por outros agentes intervenientes no diálogo). O formalismo proposto é uma versão alterada do Cálculo de Eventos que permite a representação de eventos não instantâneos e, eventualmente, simultâneos.

- Representação de atitudes

  As atitudes (crenças, intenções e objectivos) do agente-sistema computacional a modelar devem ser suportadas por uma teoria que defina as suas características e as diversas relações existentes entre si. Na secção 2.5 é proposto um conjunto de regras necessário a uma correcta modelação de atitudes.

Para o processo de participação em diálogos, além do processo de representação de conhecimento analisada neste capítulo, é fundamental modelar o agente-sistema computacional (capítulo 3) e suportar raciocínios que permitam a actualização e revisão das atitudes do agente e o planeamento e a geração de actos de fala (capítulos 4 e 5).

2.2 Ambiente de programação em lógica

O ambiente utilizado para satisfazer os requisitos definidos na secção anterior, é a programação em lógica estendida com negação explícita, que fornece um sistema formal adequado à representação de conhecimento e ao suporte a diversos tipos de raciocínio ([AP96, ADP95, Alf93, Dam96, DNP94])

A existência de um procedimento de derivação descendente SLX[1] para programas em lógica estendidos com negação explícita e com a utilização da semântica bem fundada (*WFSX*) permite, também, o recurso a processos de interrogação em que se prova se determinados literais pertencem ou não ao modelo do programa. Este é um aspecto fundamental, dado que, no processo de inferência de atitudes, irá ser necessário saber se determinados literais pertencem à semântica bem fundada dos programas em lógica que modelam os agentes. O objectivo é poder obter as atitudes (crenças, objectivos e intenções) de um agente em relação a uma determinada propriedade sem ter necessidade de calcular a semântica global do programa.

Pereira et al. demonstraram no seu trabalho que o procedimento SLX é coerente e completo para programas instanciados podendo, no entanto, ser aplicado a programas não instanciados desde que sejam admissíveis e com termos limitados. Em termos gerais, é necessário impedir que haja recursividade infinita e literais por omissão não instanciados. Deste modo, é possível garantir a terminação dos procedimentos SLX e a sua coerência e completude.

Programas admissíveis são programas em que todas as variáveis que aparecem no corpo de uma regra, também aparecem na cabeça; isto é, não existem variáveis locais às

[1] *SLX - Selected Linear resolution for eXtended programs*

regras. Por exemplo, a regra seguinte não está de acordo com esta característica (y é uma variável local) e o programa em lógica que a contenha não é admissível:

- $p(x) \leftarrow q(x,y),\ r(y)$

Programas com termos limitados são programas em que a complexidade dos termos diminui segundo uma determinada métrica. A regra abaixo não tem termos limitados, dado que a complexidade de $p/1$ não diminui:

- $p(x) \leftarrow p(f(x))$

Como regra geral, a terminação do procedimento SLX garante a sua completude e coerência.

No domínio deste trabalho é possível garantir que não existem programas com termos não limitados e que, embora haja a utilização de variáveis locais, não existe recursividade infinita, nem literais por omissão não instanciados. Deste modo é garantida a terminação dos procedimentos SLX e, consequentemente, a sua coerência e completude. No capítulo 7 serão apresentadas as opções tomadas de modo a garantir esta característica.

Em suma, recorrendo à WFSX e utilizando o procedimento SLX tem-se um sistema de programação em lógica com procedimentos de prova coerentes e completos.

2.2.1 Definição da WFSX e WFSX$_p$

Nesta secção apresento a sintaxe de programas em lógica estendida com a negação explícita. A semântica utilizada é a semântica bem fundada para programas em lógica com negação explícita (WFSX). No caso dos programas serem contraditórios é utilizada a versão paraconsistente da WFSX, a WFSX$_p$, que está definida para programas contraditórios e que permite a definição de processos de revisão que eliminem a contradição.

As definições que utilizo foram descritas em [AP96, Dam96] e são baseadas no cálculo do ponto fixo do operador Γ ([GL88]) e numa lógica parcial a 2-valores. Nos trabalhos referidos, Pereira et al., apresentam definições alternativas para a WFSX e para a WFSX$_p$ que são baseadas em lógica a 3-valores.

Os programas em lógica estendida são conjuntos de regras, com a seguinte forma:

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

onde H, $B_1, \ldots, B_n$, C_1,..., C_m são literais objectivos. Um literal objectivo é um átomo A ou a sua negação explícita $\neg A$; *not* representa a negação por omissão ou supletiva; *not* L designa-se por literal por omissão ou supletivo. Os literais ou são objectivos ou supletivos e $\neg\neg L \equiv L$. Nas regras de integridade, H é o símbolo $\bot$ (contradição).

O conjunto de todos os literais objectivos de um programa P designa-se por base estendida de Herbrand de P e representa-se por $H(P)$.

Definição 1 *Uma interpretação I de um programa em lógica P é representada por $T \cup not\ F$, onde T e F são subconjuntos disjuntos de $H(P)$. Os literais objectivos que pertencem a T são verdadeiros em I; os literais objectivos que pertencem a F são falsos por omissão em I; os literais objectivos que pertencem a $H(P) - (T \cup F)$ são indefinidos em I. Uma interpretação é coerente sse para cada L em T, $\neg L$ pertencer a F.*

Definição 2 *Seja I uma interpretação de um programa em lógica P.*

I satisfaz um literal L objectivo ou supletivo ($I \models L$), sse $L \in I$.

I satisfaz a conjunção $L_1, ..., L_m, not\ G_1, ..., not\ G_n$

$$I \models L_1, ..., L_m, not\ G_1, ..., not\ G_n$$

sse $I \models L_1, ..., I \models L_m$, e $I \models not\ G_1, ..., I \models not\ G_n$.

I satisfaz a regra $L_0 \leftarrow L_1, ..., L_m, not\ G_1, ...not\ G_n$

$$I \models (L_0 \leftarrow L_1, ..., L_m, not\ G_1, ...not\ G_n)$$

se sempre que o corpo da regra for satisfeito em I, L_0 também for satisfeito em I.

Definição 3 *Uma interpretação I é um modelo de um programa em lógica estendida P sse todas as regras de P são satisfeitas em I.*

A definição da semântica dos programas em lógica estendida é baseada no operador Γ, definido por Gelfond e Lifschitz ([GL88]).

Definição 4 *Operador Γ.*

Seja P um programa estendido e I uma interpretação a 2-valores Ie.e. onde $T \cup F = H(P)$. A transformação-GL[2] $\frac{P}{I}$ é o programa obtido a partir de P através da supressão das regras que contêm o literal por omissão not A, tal que $A \in I$, e pela supressão dos restantes literais por omissão de P. Por definição. $\Gamma_P I$ é o modelo mínimo a 2-valores de $\frac{P}{I}$.

O operador Γ tem como característica que os pontos fixos de um programa P também são modelos de P ($\Gamma(I) = I$). Esta característica serviu de base à definição da semântica bem fundada (ver definição 6). Como exemplo, suponhamos o seguinte programa:

Exemplo 1 *Programa P ([AP96]):*

$$\begin{aligned} a &\leftarrow not\ b \\ b &\leftarrow not\ a \\ c &\leftarrow not\ d \\ d &\leftarrow not\ e \\ p &\leftarrow a \\ p &\leftarrow b \end{aligned}$$

[2]Gelfond-Lifschitz

A transformação-GL $\frac{P}{I}$*, onde* $I = \{p, a, d\}$ *é dada por:*

$$\begin{aligned} a &\leftarrow \\ d &\leftarrow \\ p &\leftarrow a \\ p &\leftarrow b \end{aligned}$$

e $\Gamma(I) = \{p, a, d\} = I$.

A versão semi-normal de um programa garante que o requisito de coerência é satisfeito (para cada literal objectivo L, se $\neg L$ é verdadeiro na semântica, então $not\ L$ também o é):

Definição 5 *Versão semi-normal de um programa*

A versão semi-normal de um programa P *é o programa* P_S *obtido a partir de* P *acrescentando o literal por omissão* $not\ \neg L$ *ao corpo de cada regra do tipo* $L \leftarrow ...$

Exemplo 2 *A versão semi-normal do programa*

$$P = \{a, b \leftarrow a, \neg c\}$$

é dada por

$$P_S = \{a \leftarrow not\ \neg a, b \leftarrow a, not\ \neg b, \neg c \leftarrow not\ c\}$$

Por questões de simplificação da designação, utilizar-se-á Γ_S para denotar Γ aplicado a P_S (a versão semi-normal de P).

O modelo bem fundado de um programa P não-contraditório é dado por:

Definição 6 *Modelo WFSX*

Seja P *um programa em lógica estendida não-contraditório e* T *o ponto fixo mínimo de* $\Gamma\Gamma_S$*. O modelo* $M = T \cup not\ (H_P - \Gamma_S T)$ *é o modelo bem fundado de* P*.*

Teorema 1 *Semântica WFSX*

Cada programa P *não contraditório tem um ponto fixo mínimo de* $\Gamma\Gamma_S$*, que é designado por modelo bem fundado de* P *(*$WFM(P)$*).*

Exemplo 3 *O modelo WFSX do programa ([AP96]):*

$$P = \{a \leftarrow not\ b, b \leftarrow not\ a, \neg a\}$$

é dada por

$$\{\neg a, b, not\ a, not\ \neg b\}$$

No entanto, devido à introdução da negação explícita, é possível que os programas sejam contraditórios e não tenham modelo dado pela WFSX.

Por exemplo, no programa seguinte

- $P = \{a \leftarrow not\ b, \neg a, c\}$ (a partir de [PAA94])

  $not\ b$ é verdadeiro por omissão e, pela segunda regra, existe uma contradição (a e $\neg a$ são verdadeiros).

A contradição poderá ser removida através da alteração dos valores de um subconjunto de literais revisíveis. O processo é efectuado através da revisão do estado lógico de literais negativos verdadeiros no modelo do programa. Pereira et al. em [PAA94] propuseram uma semântica para a remoção de contradições (CRSX) que assumia que os literais revisíveis eram literais supletivos cujo complemento não possuia regras. Em [AP96] esta limitação é suprimida e os literais revisíveis não estão sujeitos a qualquer limitação, assumindo-se que são definidos pelo utilizador em conjunto com o programa.

No exemplo apresentado, supondo que o conjunto dos revisíveis é:

$$\{not\ b\}$$

então o modelo de P já não inclui a, dado que $not\ b$ deixou de ser verdadeiro e passou a estar indefinido (foi revisto o valor lógico de b).

O processo de revisão pode ser descrito do seguinte modo (a partir de [AP96]): em primeiro lugar é definida uma versão paraconsistente da WFSX, a WFSX$_P$ (definida para programas eventualmente contraditórios); de seguida, é definido declarativamente um processo de revisão que permite alterar assumpções já efectuadas; finalmente, é definido a revisão minimal de um programa contraditório.

A definição da versão paraconsistente de um programa em lógica estendida é obtido a partir da definição da WFSX, simplesmente pela sua utilização no âmbito mais alargado de programas possivelmente contraditórios (para uma maior análise ver [Dam96]):

O conjunto paraconsistente WFSX$_P$ de um programa P é dado por:

Definição 7 *WFSX$_P$*

Seja P um programa em lógica estendida e T o ponto fixo mínimo de $\Gamma\Gamma_S$. $T \cup not\ (H_P - \Gamma_S T)$ é o conjunto WFSX$_P$ paraconsistente de P.

Teorema 2 *Semântica WFSX$_P$*

Cada programa P tem um ponto fixo mínimo de $\Gamma\Gamma_S$, que é designado por conjunto paraconsistente de P ($WFM_P(P)$).

A revisão de um programa contraditório pode ser efectuada a através de regras de inibição que impedem que literais sejam falsos nos modelos do programa:

Definição 8 *Regra de inibição (a 3 valores). A regra de inibição para um literal supletivo not L é:*

$$L \leftarrow not\ L.$$

As regras de inibição para o conjunto S de de literais supletivos é:

$$IR(S) = \{L \leftarrow not\ L \mid not\ L \in S\}$$

Definição 9 *Submodelo de um programa. Um submodelo de um programa (contraditório) P com um conjunto de revisíveis Rev, é um par $< M, R >$, onde R é um subconjunto de Rev:*

$$M = WFSX_p(P \cup \{L \leftarrow not\ L \mid not\ L \in R\})$$

R designa a revisão do submodelo e M são as consequências da revisão do submodelo. Um submodelo é contraditório sse M for contraditório.

Definição 10 *Submodelo minimal não-contraditório. Um submodelo $< M, R >$ é um submodelo minimal não-contraditório (MNS) de um programa P sse é não contraditório e não existe nenhum outro submodelo $< M', R' >$ tal que $R' \subset R$.*

A revisão de um programa contraditório P é definida com base nos sub-modelos minimais não-contraditórios desse programa.

Definição 11 *Revisão minimal de um programa. Seja P um programa com um conjunto de literais revisíveis Rev e $< M, R >$ um submodelo minimal não-contraditório de P. A revisão minimal de P (MRP) é:*

$$MRP(P) = P \cup IR(R).$$

Definição 12 *Revisão céptica. O submodelo céptico de um programa P é a junção $< M_J, R_J >$ de todos os MNS de P. A revisão céptica de um programa P é o programa obtido de P através da adição de uma regra de inibição para cada elemento R_J.*

Como exemplo de revisão céptica de um programa contraditório vejamos o seguinte exemplo ([AP96])

Exemplo 4 *Seja P o programa:*

$$p \leftarrow not\ q$$
$$\neg p \leftarrow not\ r$$
$$a \leftarrow not\ b$$

com literais revísiveis $Rev = \{not\ q, not\ r, not\ b\}$. Este programa tem duas revisões minimais possíveis: rever q ou rever r

$$MRP_1 = \{p \leftarrow not\ q, \neg p \leftarrow not\ r, a \leftarrow not\ b, q \leftarrow not\ q\}$$
$$MRP_2 = \{p \leftarrow not\ q, \neg p \leftarrow not\ r, a \leftarrow not\ b, r \leftarrow not\ r\}$$

A revisão céptica revê q e r.

Os procedimentos necessários para obter a revisão minimal e a revisão céptica de um dado programa em lógica estendida estão descritos e exemplificados em detalhe em [AP96] e são baseados na noção de suporte de contradição e de conjuntos de supressão de contradição.

No âmbito deste trabalho a contradição é resolvida através da obtenção da revisão a 2 valores dos programas em lógica estendida que modelam os agentes, conforme o definido no sistema REVISE ([SDP96]). Na revisão a 2-valores o valor lógico de literais supletivos verdadeiros ($not\ A$) é revisto para falso (acrescentando A ao programa em lógica). Utilizou-se a revisão a 2 valores, dado pretender-se obter modelos em que os revisíveis verdadeiros devido à hipótese do mundo fechado (eventos que não ocorreram) sejam revistos para falso e não para indefinido; isto é, pretendem-se modelos em que os eventos ou ocorreram ou não (ver secção 2.3).

Além disto, a selecção do modelo preferido é efectuda entre os modelos estáveis estendidos (XSM) dos sub-modelos minimais não contraditórios (MNS), designados por $CFXSM$ — *Contradiction Free eXtended Stable Model* (ver [PAA91a] para uma descrição detalhada). O modelo preferido será designado por $PCFXSM$ — *Preferred Contradiction Free eXtended Stable Model* — e é obtido a partir da utilização de regras de preferência de revisíveis (sistema REVISE).

2.2.2 Tipos de raciocínio

O ambiente de programação em lógica estendida utilizado permite a modelação de diversos tipos de raciocínio não-monótono, nomeadamente, raciocínio por omissão, raciocínio hipotético e raciocínio abdutivo. Nos capítulos 4 e 5 será demonstrada a aplicação do raciocínio não-monótono ao processo de inferência de atitudes em diálogos.

O raciocínio por omissão pode ser modelado através da inserção de regras do seguinte tipo ([PAA91b]):

Normalmente $A(X)$ implica $B(X)$

que pode ser traduzida por

$$B(X) \leftarrow A(X),\ not\ ab(X)$$

Esta regra significa que, se não for possível provar a não normalidade de X, e se $A(X)$ for verdadeiro, então $B(X)$ deve ser verdadeiro.

Como exemplo de aplicação aos actos de fala de que trataremos mais tarde, poderemos ter:

Normalmente, $inform(A, B, P)$ implica $bel(B, P)$

que significa que o acto de informar causa normalmente no ouvinte a crença naquilo de que for informado. A regra correspondente será:

$$bel(B, P) \leftarrow inform(A, B, P),\ not\ ab(inform(A, B, P))$$

representando que, se um agente A informa outro agente B sobre alguma proposição P, então B passará a acreditar em P, i.e., por omissão os agentes acreditam no que lhes é dito.

Os programas em lógica com negação explícita também podem suportar raciocínio hipotético. Retomando o exemplo anterior:

Os agentes podem ou não acreditar no que lhes é dito.

Esta regra poderá ser reescrita como:

$$\begin{aligned} bel(Y,P) &\leftarrow inform(X,Y,P),\ hipotese\_verdadeiro(X). \\ hipotese\_verdadeiro(X) &\leftarrow not\ \neg hipotese\_verdadeiro(X). \\ \neg hipotese\_verdadeiro(X) &\leftarrow not\ hipotese\_verdadeiro(X). \end{aligned}$$

que significa que um agente acredita no que lhe é dito na hipótese de o emissor ser *verdadeiro*. O emissor é *verdadeiro* se não for possível provar que ele é *falso* (e vice-versa). Note-se que, embora as hipóteses sejam indefinidas no modelo bem fundado, existem modelos parciais que contêm a hipótese e outros que contêm a sua negação.

O raciocínio abdutivo, essencial ao processo de participação em diálogos conforme será apresentado nos capítulos seguintes, pode ser descrito do seguinte modo:

Definição 13 *Dada uma teoria T e um conjunto de observações O, encontrar uma teoria Δ, tal que*

$$T \cup \Delta \models O,$$
$$T \cup \Delta \textit{ é consistente (não contraditório).}$$

Δ é considerada a justificação das observações.

Um dos problemas do raciocínio abdutivo é a escolha das melhores justificações. O critério mais utilizado é o de escolher as soluções básicas e minimais:

1. Uma justificação é básica se nenhum dos factos na justificação poder ser explicado pela teoria.

2. Uma justificação é minimal se não existir nenhuma outra justificação que seja um subconjunto dos seus factos.

Kowalski e Eshgi ([EK89, Esh88]) propõem um esquema abdutivo para a programação em lógica que permite encontrar justificações: $< P, Abd, IC >$, onde P é um programa em lógica com negação por omissão, Abd é um conjunto de literais que podem ser abduzidos e IC é um conjunto de restrições de integridade.

Utilizando a programação em lógica com negação explícita e a semântica bem fundada é possível obter um esquema abdutivo com características idênticas:

Definição 14 *[PAA91b]Seja P' o programa em lógica obtido a partir da reunião do programa P com as restrições de integridade IC. Para todos os literais L que pertencem à lista dos abdutíveis Abd, duas regras deverão ser acrescentadas a P':*

$$L \leftarrow not\ \neg L.$$
$$\neg L \leftarrow not\ L.$$

Estas regras significam que se não se poder provar $\neg L$, então L deverá ser verdadeiro (e vice versa). De notar que na ausência de qualquer regra adicional, L e $\neg L$ serão indefinidos (não existe nenhum facto que suporte qualquer uma das possibilidades).

Como exemplo, no caso de um agente/sistema computacional que esteja a conversar com um interlocutor a, que pode ser ou não mentiroso (propriedade abdutível), pode-se acrescentar:

$$\begin{aligned} mentiroso(a) &\leftarrow not\ \neg mentiroso(a). \\ \neg mentiroso(a) &\leftarrow not\ mentiroso(a). \end{aligned}$$

Estas regras permitem abduzir a propriedade de a ser (ou não) mentiroso, no caso de existir uma observação que seja uma consequência dessa propriedade.

O raciocínio abdutivo pode ser definido do seguinte modo:

Definição 15 *Seja P' o programa em lógica estendida correspondente ao esquema abdutivo $< P, Abd, IC >$ de acordo com a transformação descrita na definição anterior. Nestas condições, a observação O, que pode ser vista como uma restrição de integridade adicional $O \Leftarrow$[3], tem uma explicação abdutiva Δ sse:*

$$\begin{aligned} P' \cup \Delta &\models_{WFSX} O \\ P' &\not\models_{WFSX} \Delta \end{aligned}$$

Isto é, a observação O é suportada pelo modelo bem fundado do programa em lógica estendida com negação explícita constituído pelo programa P' e pelas explicações abduzidas Δ; as explicações Δ não pertencem ao modelo bem fundado do programa P'.

No domínio deste trabalho, será permitida a abdução de eventos de modo a permitir o planeamento de acções a realizar. Nomeadamente, será possível abduzir eventos que venham a permitir a satisfação dos objectivos do agente-sistema computacional. Este processo será realizado através do uso deste ambiente de programação e de uma versão alterada do Cálculo de Eventos (descrita na próxima secção)e será descrito em detalhe nos capítulos 4 e 5.

[3]As restrições de integridade serão identificadas pelo uso do símbolo $\Leftarrow$.

2.3 Eventos

A representação temporal das acções, sejam actos de fala ou acções do domínio (apanhar comboios, comprar bilhetes, etc.), é fundamental para uma participação em diálogos eficaz. Na realidade, o factor tempo desempenha um papel crucial no processo de inferência e revisão das atitudes dos agentes e no seu comportamento para com os outros interlocutores.

De modo a ser possível raciocinar sobre as acções realizadas (ou a realizar) propõe-se uma versão alterada do Cálculo de Eventos, que tem como base a proposta de Missiaen [Mis91] (por sua vez baseada em [Esh88]) e que foi alterada de modo a suportar acções não instantâneas e simultâneas. O Cálculo de Eventos tem como vantagem a capacidade para representar explicitamente os eventos ocorridos e a possibilidade de obter as propriedades válidas em qualquer instante de tempo (passado ou futuro).

Na proposta de Missiaen, os eventos (E) são instâncias de acções e não têm duração temporal. São representados pelo instante de tempo em que ocorrem (T), sendo os instantes temporais ordenados pela relação de precedência $<$. Não existe, portanto uma distinção entre a representação de um evento e a do instante de tempo em que ele ocorreu, sendo utilizado indistintamente qualquer uma das representações (E ou T). As mudanças de estado são provocadas pela ocorrência de eventos e os estados não têm uma identificação própria. Os eventos (E) estão relacionados com as acções (A) das quais são instâncias pelo predicado $act(E, A)$; o predicado $happens(E)$ significa que o evento E ocorreu no instante de tempo E; $initiates(E, P)$ significa que o evento E inicia a propriedade P; $terminates(E, P)$ significa que a propriedade P é terminado pelo evento E; $enabled(E)$ significa que o evento E pode vir a ocorrer, isto é, as suas pré-condições estão satisfeitas (este predicado é diferente do predicado $happens$, que significa que o evento realmente ocorreu); $holds\_at(P, T)$ significa que a propriedade P é válida no instante de tempo T.

No entanto, estas versões do Cálculo de Eventos têm como lacunas fundamentais a incapacidade de representar acções simultâneas e acções com duração temporal não instantânea, devido a representarem os eventos através dos instantes temporais em que ocorrem, confundindo a representação de duas entidades distintas.

No âmbito deste trabalho estes pressupostos são demasiado fortes, pelo que é necessário aumentar o poder de representação do Cálculo de Eventos, de modo a suportar a representação de acções com duração temporal não instantânea, bem como acções concorrentes (no mesmo instante de tempo poderão estar a acontecer várias acções em simultâneo). Deste modo, será possível suportar a participação em diálogos com um grau de complexidade superior e não sendo necessário efectuar simplificações temporais.

Como exemplo da necessidade de representar acções simultâneas, analisemos a primeira frase do exemplo apresentado no capítulo 1:

- O comboio das 8:30?

Esta frase deverá ser traduzida em dois actos de fala simultâneos: um pedido de

informação sobre um comboio (request) e uma informação adicional de que é sobre o comboio das 8:30 (informref).

Tendo em conta estes requisitos, utilizo neste trabalho uma redefinição do Cálculo de Eventos (baseada em [Mis91]). Em concreto, os eventos passam a ter uma identificação própria e a poder decorrer num determinado intervalo de tempo. Como consequência, passam a poder haver acções a decorrer em simultâneo (basta que estejam associadas a eventos que acontecem em intervalos de tempo que se sobrepõem).

Uma propriedade P verifica-se num dado instante de tempo T se:

$$\begin{aligned}
holds_at(P,T) &\leftarrow happens(E,T_i,T_f), && (2.1)\\
&\quad\ initiates(E,T_P,P),\\
&\quad\ T_P < T,\ T_i \leq T_P,\\
&\quad\ persists(T_P,P,T).\\
persists(T_P,P,T) &\leftarrow not\ clipped(T_P,P,T). && (2.2)\\
clipped(T_P,P,T) &\leftarrow happens(C,T_{ci},T_{cf}), && (2.3)\\
&\quad\ terminates(C,T_C,P),\\
&\quad\ not\ out(T_C,T_P,T).\\
out(T_C,T_P,T) &\leftarrow T \leq T_C. && (2.4)\\
out(T_C,T_P,T) &\leftarrow T_C < T_P. && (2.5)
\end{aligned}$$

Note-se que nesta proposta o predicado $happens(E,T_i,T_f)$ significa que o evento E ocorreu entre o instante de tempo T_i e o instante de tempo T_f; $initiates(E,T,P)$ significa que o evento E inicia a propriedade P no instante de tempo T; o predicado $terminates(E,T,P)$ significa que o evento E termina a propriedade P no instante de tempo T; $persists(T_i,P,T)$ significa que a propriedade P persiste desde T_i até T. Existe ainda o predicado $enabled(E,T_i)$ que significa que o evento E pode ocorrer no instante T_i.

O predicado $holds\_at(P,T)$ significa que a propriedade P é válida no instante de tempo T se houver um evento anterior a T que tenha iniciado a propriedade P e se P persistir até ao instante de tempo T. A propriedade persiste até ao instante T se não for possível provar (por negação por omissão) a existência de um evento que termine a propriedade P antes do tempo T. Note-se que se utiliza uma estratégia conservadora sobre a análise do instante de tempo T_C que terminou a propriedade P, estar ou não dentro do intervalo $]T_P,T]$: no caso de não se conseguir provar que está fora, então assume-se que T_C está dentro do intervalo e que a propriedade P foi terminada.

De notar que é necessário adicionar uma a regra de integridade que garanta que uma propriedade e a sua negação não são válidas no mesmo instante de tempo:

$$\Leftarrow holds\_at(P,T),\ holds\_at(\neg P,T).$$

Esta regra pode ser substituída pela representação explícita da relação entre a validade da negação de uma propriedade e a negação da sua validade:

$$\neg holds\_at(P,T) \leftarrow holds\_at(\neg P,T) \quad (2.6)$$

$$\neg holds\_at(\neg P,T) \leftarrow holds\_at(P,T) \quad (2.7)$$

Por outro lado, é necessário lidar com a existência de informação incompleta e modelar quais as propriedades válidas num instante de tempo. Para tal é necessário relacionar a negação explícita com a negação por omissão. A semântica bem fundada para programas em lógica com negação explícita garante a coerência, isto é, se $\neg holds\_at(P,T)$ pertencer à semântica do programa então $not\ holds\_at(P,T)$ também pertence. No entanto, no caso do predicado $holds\_at/2$ é necessário garantir que uma propriedade não é válida (explicitamente) se não se puder provar por omissão a sua validade, ou seja, na inexistência de informação assume-se a negação (CWA- hipótese do mundo fechado). Esta regra será representada por:

$$\neg holds\_at(P,T) \leftarrow not\ holds\_at(P,T). \quad (2.8)$$

Existe, ainda, a necessidade de definir regras de integridade adicionais que serão utilizadas para restringir os modelos possíveis dos programas em lógica estendida:

1. Impossibilidade de um evento iniciar e terminar uma propriedade no mesmo instante de tempo:

$$\Leftarrow initiates(E,T,P),\ terminates(E,T,P). \quad (2.9)$$

2. Impossibilidade de um evento iniciar uma propriedade e a sua complementar no mesmo instante de tempo:

$$\Leftarrow initiates(E,T,P), initiates(E,T,\neg P). \quad (2.10)$$

3. Impossibilidade de um evento estar associado a intervalos de tempo distintos:

$$\begin{aligned} \Leftarrow\ & happens(E,T_{1i},T_{1f}), \\ & happens(E,T_{2i},T_{2f}), \\ & not(T_{1i}=T_{2i},T_{1f}=T_{2f}). \end{aligned} \quad (2.11)$$

4. Impossibilidade de um evento ter duração *negativa*:

$$\Leftarrow happens(E,T_i,T_f),\ T_f < T_i. \quad (2.12)$$

5. Impossibilidade de um evento não ter uma acção associada:

$$\begin{aligned} \Leftarrow\ & happens(E,T_i,T_f), \\ & not(act(E,A)). \end{aligned} \quad (2.13)$$

6. Impossibilidade de uma propriedade ser válida sem existir um evento que iniciou a sua validade (conforme será apresentado posteriormente nesta secção, esta regra possibilita a abdução de eventos, $happens/3$, que permitem explicar as propriedades verificadas num instante de tempo):

$$\begin{aligned} & \Leftarrow && holds\_at(P,T), \\ & && not(ev\_gen(P,T)). \\ ev\_gen(P,T) & \leftarrow && happens(E,T_i,T_f), \\ & && initiates(E,T_p,P), \\ & && T_i \leq T_p < T, \\ & && persists(T_p,P,T). \end{aligned} \tag{2.14}$$

   De notar a criação de um predicado adicional $ev\_gen(P,T)$ que é válido quando existe um evento que gera a propriedade P e em que essa propriedade é válida pelo menos até o instante de tempo T.

7. Impossibilidade de um evento ocorrer sem ter as suas pré-condições satisfeitas:

$$\Leftarrow happens(E,T_i,T_f),\ not\ enabled(E,T_i). \tag{2.15}$$

Um outro problema está relacionado com a possibilidade das acções se sobreporem no tempo e surgir a questão da resolução da interferência dessas acções, dado que elas poderão não ser independentes entre si. De facto, a execução simultânea de acções pode iniciar/terminar propriedades diferentes das iniciadas/terminadas pela execução linear dessas mesmas acções e podem, mesmo, vir a ser cancelados os efeitos das acções individuais.

Móra no seu trabalho ([MLC95]) propõe a introdução de um novo termo $conc(E1,E2)$, de modo a poder representar a concorrência de dois eventos, e a criação de um novo predicado $cancels(Act1,Act2)$ para representar a interferência entre as acções (a acção $Act1$ cancela os efeitos da acção $Act2$, caso sejam realizadas em simultâneo).

No entanto, o programa em lógica proposto anteriormente já suporta as características relacionadas com as interferências de acções simultâneas.

O início de propriedades pela execução simultânea das acções poderá ser representado por regras do seguinte tipo:

$$\begin{aligned} initiates(E_1,T,P_1) & \leftarrow && act(E_1,A_1), \\ & && act(E_2,A_2), \\ & && happens(E_1,T_i,T_f), \\ & && happens(E_2,T_i,T_f). \end{aligned}$$

que significa que se as acções A_1 e A_2 ocorrerem simultaneamente, então a propriedade P_1 será iniciada. O término de propriedades poderá ser efectuado de um modo idêntico (utilizando o predicado $terminates/3$).

O cancelamento de efeitos de uma acção por uma outra acção que ocorra simultaneamente pode ser representado através do recurso a regras de raciocínio supletivo. Como exemplo, suponhamos que a acção de abrir uma janela tem como consequência que essa janela fique aberta, excepto se houver uma acção simultânea que cancele esse efeito (fechar a janela):

$$\begin{aligned} initiates(E, T_f, janela\_aberta) \leftarrow\ & act(E, abrir\_janela), \\ & happens(E, T_i, T_f), \\ & not\ cancelled(E). \\ cancelled(E) \leftarrow\ & act(E, abrir\_janela), \\ & happens(E, T_i, T_f), \\ & act(E_1, fechar\_janela), \\ & happens(E_1, T_i, T_f). \end{aligned}$$

Deste modo o efeito *janela_aberta* é cancelado caso existisse uma execução simultânea das acções de abrir e fechar a janela.

Note-se que esta abordagem pressupõe uma completa descrição dos conflitos possíveis de existir devido à execução simultânea de acções. Pretende-se, somente, ilustrar as capacidades de representação da abordagem utilizada.

Por outro lado, Alferes et al. [ALP95b] propuseram uma linguagem de descrição de acções concorrentes que considera indefinidos os efeitos de acções simultâneas em que existam conflitos (isto é, um efeito é iniciado e terminado por sub-acções simultâneas). A metodologia que é proposta, ao cancelar os efeitos de acções com conflitos, mantém o valor lógico anterior das propriedades. No exemplo apresentado, após uma acção simultânea de abrir e fechar uma janela, ela permaneceria no estado anterior a essa acção.

2.3.1 Planeamento com o Cálculo de Eventos

A representação proposta permite definir e construir planos recorrendo à utilização dos predicados *happens* e *act*. Utilizando estes predicados é possível representar os eventos que ocorrem e as acções associadas a estes eventos. Por exemplo, o conjunto seguinte pode representar o plano de fazer uma viagem de comboio (comprar o bilhete e, posteriormente, embarcar no comboio):

$$\begin{aligned} & happens(e_1, t_{1i}, t_{1f}). \\ & act(e_1, comprar\_bilhete). \\ & happens(e_2, t_{2i}, t_{2f}). \\ & act(e_2, embarcar\_comboio). \\ & t_{1i} \leq t_{if} < t_{2i} \leq t_{2f}. \end{aligned}$$

O planeamento das acções a obter, de modo a satisfazer determinados objectivos (propriedades que se pretendem sejam válidas num instante de tempo posterior), é efectuado através do recurso à abdução dos eventos a realizar que permitirão a sua satisfação.

Assim, planear pode ser encarado como um processo abdutivo em que o conjunto de predicados abdutíveis é composto por:

$$Ab = \{happens/3,\ act/2\}.$$

Este conjunto de abdutíveis permite a abdução de eventos e da sua relação com as acções. Com base nestes abdutíveis e nas restrições de integridade apresentadas anteriormente é possível obter um planeamento que satisfaça os objectivos existentes.

De facto, dada uma teoria T (que inclui as regras definidas nesta secção) e um objectivo G (do tipo $holds\_at(P,T)$), o plano Pl para satisfazer o objectivo G é definido pelo conjunto Δ de predicados abduzidos ($happens/3$ e $act/2$), tal que:

$$T \cup \Delta \models_{WFSX} G$$

Ou seja, o plano Pl é definido pelo conjunto de eventos que foram abduzidos de modo a satisfazer o objectivo. De acordo com o apresentado na secção anterior, o objectivo G pode ser visto como a restrição de integridade adicional $G \Leftarrow$.

A título de exemplo iremos analisar o modo como o Cálculo de Eventos, com as alterações propostas nesta tese, suporta uma situação em que há eventos não instantâneos e simultaneidade de acções:

Exemplo 5 *O problema de Glasgow (adaptado de [Lif93]):*

Na situação inicial, o agente está em Glasgow, tem bilhete de avião e existem voos de Glasgow para Londres e de Londres para Moscovo. Além disso, o agente pretende estar em Moscovo num determinado instante de tempo futuro.

$$\begin{aligned}
initiates(e_1, t_0, existsFlight(glasgow, londres)) &\leftarrow happens(e_1, t_0, t_0).\\
initiates(e_1, t_0, existsFlight(londres, moscovo)) &\leftarrow happens(e_1, t_0, t_0).\\
initiates(e_1, t_0, hasTicket) &\leftarrow happens(e_1, t_0, t_0).\\
initiates(e_1, t_0, at(glasgow)) &\leftarrow happens(e_1, t_0, t_0).\\
happens(e_1, t_0, t_0).&\\
act(e_1, start).&
\end{aligned}$$

Esta representação assume um evento e_1 e uma acção $start$ responsáveis pela satisfação das propriedades iniciais.

A acção $fly(X,Y)$ tem a seguinte descrição (simplificada):

$$
\begin{aligned}
enabled(E, T_i) \leftarrow\ & act(E, fly(X, Y)), \\
& holds\_at(at(X), T_i), \\
& holds\_at(hasTicket, T_i), \\
& holds\_at(existsFlight(X, Y), T_i). \\
initiates(E, T_f, at(Y)) \leftarrow\ & happens(E, T_i, T_f), \\
& act(E, fly(X, Y)). \\
terminates(E, T_i, at(X)) \leftarrow\ & happens(E, T_i, T_f), \\
& act(E, fly(X, Y)).
\end{aligned}
$$

Representando que a acção de voar de um local X para um local Y inicia a propriedade de o agente estar em Y (e deixar de estar em X), tendo como pré-condições que o agente está no local X, tem um bilhete de avião, existe um voo de X para Y e que voar é a acção associada ao evento.

O objectivo do agente pode ser representado por:

$$holds\_at(at(moscovo), t), t_0 < t$$

Utilizando os axiomas temporais apresentados anteriormente, bem como as restrições de integridade descritas, o plano do agente e as relações temporais dos eventos abduzidos é dado por:

$$
\begin{aligned}
& happens(e_2, t_{2i}, t_{2f}). \\
& act(e_2, fly(glasgow, londres). \\
& t_{2i} \leq t_{2f}. \\
& t_0 \leq t_{2i}. \\
& happens(e_3, t_{3i}, t_{3f}). \\
& act(e_3, fly(londres, moscovo). \\
& t_{3i} \leq t_{3f}. \\
& t_{2f} < t_{3i}
\end{aligned}
$$

Note-se que, de forma a efectuar a implementação do processo abdutivo sobre a semântica bem fundada de programas em lógica estendida, será necessário limitar os eventos possíveis de serem realizados e os instantes de tempo existentes (ver capítulo 7 para uma análise deste problema em detalhe).

De modo a exemplificar a utilização de acções concorrentes poderemos supor que durante o voo de Glasgow para Londres o agente escreve uma carta:

$$
\begin{aligned}
& act(e_4, writeLetter). \\
& happens(e_4, t_{2i}, t_{2f}).
\end{aligned}
$$

Existe uma regra que define os efeitos de escrever uma carta:

$$initiates(E, T_f, letterWritten) \leftarrow happens(E, T_i, T_f), \\ act(E, writeLetter).$$

Com base nestas regras, a semântica bem fundada do programa suporta as seguintes inferências :

$\neg holds\_at(letterWritten, t_0).$

$holds\_at(letterWritten, t_{2f}).$

$holds\_at(letterWritten, t_{3f}).$

Conforme se pode observar, as alterações propostas ao Cálculo de Eventos permitem suportar situações mais complexas, em que existe simultaneidade de acções e acções não instantâneas (situações não suportadas pelas propostas originais de Eshgi e Missiaen). Estas características são necessárias a uma participação eficaz em diálogos, dado permitirem modelar interacções em que os agentes realizam actos de fala em simultâneo e/ou em intervalos de tempo de duração não instantânea.

2.4 Acções

Conforme foi referido anteriormente, existe a necessidade de representar, no formalismo adoptado, as diferentes acções, nomeadamente os actos de fala e as acções específicas do domínio. O formalismo proposto (programação em lógica estendida com negação explícita), além de permitir a modelação de raciocínio não monótono, fornece as ferramentas necessárias para uma correcta representação dessas acções, tanto das suas pré-condições, como dos seus efeitos.

No entanto, devido à existência de um conjunto bastante distinto de abordagens à problemática do raciocínio sobre acções, tem havido um esforço na tentativa de adoptar uma linguagem declarativa de descrição de acções que permita comparar as diversas abordagens.

Gelfond e Lifschitz ([GL92a]) propuseram a linguagem A como base de trabalho para o desenvolvimento da teoria de acções. No seu trabalho apresentaram a semântica dessa linguagem e demonstraram a coerência de uma tradução para programação em lógica estendida com a semântica de conjuntos-resposta (*answer-sets*). Posteriormente, Kartha ([Kar93]) propôs, e provou a coerência e completude de traduções para três formalismos distintos: a abordagem utilizando lógica de primeira ordem para raciocinar acerca de acções de Pednault ([Ped89]) e de Reiter ([Rei80, Rei91]), e a abordagem utilizando raciocínio não-monótono de circunscrição de Baker ([Bak91]). Baral e Gelfond ([BG93]) e Renwei Li ([LP96a]) apresentaram uma extensão da linguagem A para suportar acções

concorrentes. Denecker e De Schreye ([DS92]) demonstraram a coerência e a completude de uma tradução para programas abdutivos em lógica, com a semântica de completude de predicados (*predicate completion semantics*).

Além destes factores, a utilização de uma linguagem declarativa de alto nível permite uma descrição mais simples e mais compreensível das acções a representar.

Devido a todos estes factores, proponho neste trabalho uma versão alterada da linguagem A, que permite trabalhar com acções concorrentes e não instantâneas e apresento uma tradução para programas em lógica estendida com semântica bem fundada.

No anexo A apresento a demonstração formal da completude e coerência da tradução entre a linguagem proposta e os programas em lógica estendida com semântica bem fundada. No anexo B demonstro que todos os modelos da linguagem A de Gelfond e Lifschitz são modelos da linguagem que proponho.

2.4.1 Linguagem A

De modo a definir com exactidão a linguagem de representação de acções que eu proponho, é necessário apresentar primeiro a linguagem A, conforme proposta de Gelfond e Lifschitz.

A linguagem A definida por Gelfond e Lifschitz propunha dois tipos de proposições:

$$A \quad causes \quad F \; if \; P_1, \ldots, P_n \text{ (proposição-e — efeito)}$$
$$F \quad after \quad A_1, \ldots, A_n. \text{ (proposição-v — valor)}$$

A primeira regra significa que a acção A causa o efeito (fluente) F se as pré-condições $P_1, \ldots, P_n$ se verificarem; a segunda regra significa que o fluente F é uma consequência observada da sequência de acções $A_1, \ldots, A_n$. Se $n = 0$ então escreve-se *initially* F.

Uma proposição é uma proposição-e ou uma proposição-v. Uma descrição do domínio é um conjunto de proposições.

A descrição da semântica de A é feita a partir da definição dos modelos da descrição do domínio (ver definição 16) e pelas proposições-v que são suportadas pelos modelos desse domínio.

Um estado é um conjunto de fluentes. Dado um fluente F e um estado σ, considera-se que F é válido em σ se $F \in \sigma$; $\neg F$ é válido em σ se $F \notin \sigma$.

Uma função de transição é um mapeamento Φ de um conjunto de pares (A, σ) para um conjunto de estados, onde A é uma acção e σ é um estado.

Uma estrutura M é um par (σ_0, Φ), onde σ_0 é um estado (o estado inicial) e Φ uma função de transição.

Para cada estrutura $M = (\sigma_0, \Phi)$ e para a sequência de acções $A_1, ..., A_m$, por $M^{A_1;...;A_m}$ designa-se o estado

$$\Phi(A_m, \Phi(A_{m-1}, ..., \Phi(A_1, \sigma_0)...))$$

Uma proposição-v $F \; after \; A_1, ..., A_m$ é verdadeira numa estrutura M se F for válido no estado $M^{A_1;...;A_m}$ e é falsa em caso contrário.

Em particular, uma proposição do tipo

$$initially\ F.$$

é válida em M sse o fluente F é válido no estado inicial de M.

Definição 16 *Modelos de D*

Uma estrutura (σ_0, Φ) é um modelo da descrição do domínio D se todas as proposições-v de D são verdadeiras em (σ_0, Φ) e, para cada acção A, para cada fluente F, e cada estado σ, as condições seguintes se verificarem:

1. Se D incluir uma proposição-e que descreve F como efeito de A, e em que as pré-condições se verifiquem em σ, então $F \in \Phi(A, \sigma)$;

2. Se D incluir uma proposição-e que descreve $\neg F$ como efeito de A, e em que as pré-condições se verifiquem em σ, então $F \notin \Phi(A, \sigma)$;

3. Se D não incluir proposições-e com estas características, então $F \in \Phi(A, \sigma)$ sse $F \in \sigma$. Esta condição permite representar a persistência dos fluentes, caso não haja acções que o contrariem explicitamente.

Gelfond e Lifschitz ([GL92a, Lif93]) provaram que só existe uma função de transição Φ que satisfaz as condições referidas.

Uma descrição de domínio D é consistente se tiver um modelo e é completa se o modelo for único.

Uma proposição-v é suportada pela descrição de domínio D se for verdadeira em todos os modelos de D.

2.4.2 Linguagem A_{EC}

Neste trabalho alterou-se a linguagem proposta inicialmente por Gelfond e Lifschitz, de modo a contemplar a representação de eventos não instantâneos e com acções possivelmente concorrentes. Este requisito é fundamental dado, consoante foi referido na secção anterior, ser necessária a capacidade de representação deste tipo de eventos no sistema de participação em diálogos que proponho nesta tese.

A linguagem A_{EC} (EC significa a possibilidade de representar eventos que ocorrem em intervalos de tempo e a concorrência de acções) é consitituída por proposições do seguinte tipo (a partir do trabalho de Lifschitz [Lif93]):

1. $A\ causes\ F\ if\ P_1, ..., P_n$ (proposição-e — efeito)

2. $F\ after\ E_1(T_{1i}, T_{1f}), ..., E_m(T_{mi}, T_{mf})$ (proposição-v — valor)

3. $A\ occurs\_in\ E(T_i, T_f)$ (proposição-o — ocorrência)

4. $E_1(T_i, T_f), ..., E_m(T_i, T_f)$ $precedes$ $E'_1(T'_i, T'_f), ..., E'_n(T'_i, T'_f)$ (proposição-p — precedência)

onde A representa uma expressão de acções $A_1 \| ... \| A_n$, com $n \geq 1$, significando a ocorrência em simultâneo de acções (T_i e T_f idênticos); $E(T_i, T_f)$ representa um evento que ocorre entre os instantes de tempo T_i e T_f; F representa um fluente e $P_1, ..., P_n$ são pré-condições a serem verificadas.

A primeira regra é idêntica à da linguagem A, com a possibilidade adicional de permitir a representação de um conjunto de acções a realizar em simultâneo. A segunda regra substitui as regras de valor propostas na linguagem A, definindo o valor de um fluente não após uma sequência de acções, mas após um conjunto de eventos associados a intervalos de tempo eventualmente sobrepostos. A terceira regra significa que a acção A está associada ao evento E. A quarta regra representa a relação temporal existente entre conjuntos de eventos: o final dos eventos simultâneos $E_1, ..., E_m$ ocorre num instante de tempo anterior ao início dos eventos simultâneos $E'_1, ..., E'_n$, i.e. $T_f < T'_i$.

Note-se que as proposições-v F $after$ $A_1, ..., A_m$ de Gelfond e Lifschitz podem ser representadas por:

$$
\begin{aligned}
& A_1 \ occurs\_in \ E_1(T_{1i}, T_{1f}) \\
& ... \\
& A_m \ occurs\_in \ E_m(T_{mi}, T_{mf}) \\
& E_1(T_{1i}, T_{1f}) \ precedes \ E_2(T_{2i}, T_{2f}) \\
& ... \\
& E_{m-1}(T_{m-1i}, T_{m-1f}) \ precedes \ E_m(T_{1i}, T_{1f}) \\
& F \ after \ E_1(T_{1i}, T_{1f}), ..., E_m(T_{mi}, T_{mf})
\end{aligned}
$$

A semântica desta linguagem é efectuada a partir da definição dos modelos da descrição do domínio e pelas proposições-v suportadas por esses modelos.

Assim:

- Uma estrutura $\Psi = (\Phi, \sigma_0)$ é um par onde σ_0 é o estado inicial e Φ é um mapeamento do conjunto de pares (E^n, σ) para o conjunto de estados (E^n é um conjunto de eventos e σ é um estado);

- Um fluente F é válido num estado σ se $F \in \sigma$; $\neg F$ é válido em σ se $F \notin \sigma$;

- $F \ after \ E_1(T_{1i}, T_{1f}), ..., E_m(T_{mi}, T_{mf})$ é verdadeira numa estrutura Ψ se F for válida no estado $\Phi(E_1(T_{1i}, T_{1f}), ..., E_m(T_{mi}, T_{mf}), \sigma_0)$.

De modo a definir os modelos da descrição do domínio é necessário introduzir uma nova definição:

Definição 17 *$Effects(E^n, \sigma)$ — Efeitos de um conjunto de eventos simultâneos num dado estado*

Para qualquer conjunto de eventos simultâneos $E_1(T_i, T_f), ..., E_n(T_i, T_f)$ (designado por $E^n(T_i, T_f)$) e para qualquer estado σ, seja $Effects(E^n(T_i, T_f), \sigma)$ o conjunto de fluentes F tal que, para cada expressão de acções A e fluentes $P_1, ..., P_m$, a descrição do domínio D contém as proposições:

A *occurs_in* $E_i(T_i, T_f)$, onde $E_i(T_i, T_f) \in E^n(T_i, T_f)$;

A *causes* F *if* $P_1, ..., P_m$, sendo $P_1, ..., P_m$ válidos em σ.

e não existe nenhum evento $E_j(T_i, T_f) \in E^n(T_i, T_j)$ tal que:

A' *occurs_in* $E_j(T_i, T_j)$

A' *causes* $\neg F$ *if* $P'_1, ..., P'_k$, sendo $P'_1, ..., P'_k$ válidos em σ.

Esta definição representa os fluentes válidos num dado estado σ, após um conjunto de eventos simultâneos (eliminando os efeitos contraditórios).

É, também, necessário definir se uma proposição de ordem é, ou não, redundante:

Definição 18 *Proposição de ordem redundante. Uma proposição de ordem E precedes E' num domínio D é redundante se for possível relacionar os eventos referidos através de outras proposições de ordem do domínio. Nomeadamente, a proposição é redundante se existir um conjunto de eventos E_1, tal que E precedes E_1 pertence ao domínio D e se existir um conjunto de proposições de ordem que relacionam E_1 com E':*

$$E \text{ precedes } E_1$$
$$E_1 \text{ precedes } ...$$
$$...$$
$$E_n \text{ precedes } E'$$

Com base nesta definição é possível definir os modelos de descrição de um domínio:

Definição 19 *Modelos de D*

Uma estrutura $\Psi = (\Phi, \sigma_0)$ é um modelo de D se todas as proposição-v incluídas em D forem verdadeiras em Ψ e se para qualquer proposição de ordem $E_1, ..., E_n$ *precedes* E'_1,...,E'_m de D que seja não redundante, e para todos os fluentes F:

1. Se $F \in Effects(E'^m, \Psi(E^n, \sigma_0))$ então $F \in \Psi(E'^m, \sigma_0)$
2. Se $\neg F \in Effects(E'^m, \Psi(E^n, \sigma_0))$ então $F \notin \Psi(E'^m, \sigma_0)$

3. Se nenhuma das situações anteriores se verificar, então $F \in \Psi(E'^m, \sigma_0)$ sse $F \in \Psi(E^n, \sigma_0)$.

De acordo com a semântica apresentada, uma proposição-v é suportada pela descrição de domínio D se for verdadeira em todos os modelos de D.

Esta definição da semântica de A_{EC} tem como objectivo o suporte a um raciocínio não-monótono. De facto, existem quatro assunções base na definição da semântica:

1. Se não houver informação sobre se num determinado evento ocorreu uma dada acção, então assume-se que a acção não ocorreu;

2. Se não se souber se uma acção afecta o estado de um fluente, então assume-se que não o afecta;

3. Se um evento preceder outro e se não se souber se ocorreram acções entre eles, então assume-se que não ocorreram.

4. Se um conjunto de eventos simultâneos tem acções contraditórias acerca de F, então assume-se que F não é alterado.

Devido a estes factores, o facto de se adicionar proposições de ordem, de ocorrência ou de efeito não implica um aumento do conjunto de proposições suportadas pela descrição do domínio (não monotonicidade).

2.4.3 Tradução para programação em lógica estendida

O passo seguinte foi definir um mecanismo de tradução da linguagem A_{EC} para a programação em lógica estendida, com semântica bem fundada, e utilizando o Cálculo de Eventos proposto.

Definição 20 *Proposições-e*

As proposições-e do tipo:

A *causes* F *if* $P_1, ..., P_n$, onde $A = A_1 \| ... \| A_m$

são traduzidas para as seguintes regras:

$$\begin{aligned} enabled(E, T_i) \leftarrow\ & act(E, A_1),\ ...,\ act(E, A_m), \\ & holds\_at(P_1, T_i), \ldots, holds\_at(P_n, T_i). \\ initiates(E, T_f, F) \leftarrow\ & happens(E, T_i, T_f), \\ & act(E, A_1),\ ...,\ act(E, A_m), \\ & holds\_at(P_1, T_i), \ldots, holds\_at(P_n, T_i). \end{aligned}$$

Este esquema define que um evento E associado com uma expressão de acções A é possível se as pré-condições se verificarem no instante inicial desse evento e que, como consequência da sua ocorrência, a propriedade F passará a ser válida. É de salientar a necessidade de validar os pré-requisitos das acções no processo de validação dos seus efeitos, devido a ser possível que a mesma acção tenha efeitos distintos, consoante as pré-condições que são válidas.

Note-se que no caso do efeito ser negativo, isto é,

$$A\ causes\ \neg F\ if\ P_1, ..., P_n$$

então a segunda regra será o predicado $terminates/2$.

$$\begin{aligned} terminates(E, T_f, F) \ \leftarrow\ & happens(E, T_i, T_f), \\ & act(E, A_1),\ ...,\ act(E, A_m). \\ & holds\_at(P_1, T_i), \ldots, holds\_at(P_n, T_i). \end{aligned}$$

Se os efeitos de uma acção não forem relevantes no domínio a modelar, as proposições-e poderão ser escritas por:

$$A\ causes\ true\ if\ P_1, \ldots, P_n$$

ou, de um modo abreviado,

$$A\ if\ P_1, \ldots, P_n$$

e traduzidas apenas por:

$$\begin{aligned} enabled(E, T_i) \ \leftarrow\ & act(E, A_1),\ ...,\ act(E, A_m), \\ & holds\_at(P_1, T_i), \ldots, holds\_at(P_n, T_f). \end{aligned}$$

Se só se pretender representar relações entre propriedades, as proposições-e poderão ser representadas por:

$$A^0\ causes\ F\ if\ P_1, \ldots, P_n$$

onde A^0 representa a acção vazia. No caso de não haver ambiguidade com a situação anterior, as regras poderão ser representadas por

$$F\ if\ P_1, \ldots, P_n$$

e traduzidas em:

$$holds\_at(F, T) \leftarrow\ holds\_at(P_1, T), \ldots, holds\_at(P_n, T).$$

Definição 21 *Proposições-p*

As proposições-p do tipo:

$E_1(T_i, T_f), ..., E_n(T_i, T_f)\ precedes\ E'_1(T'_i, T'_f), ..., E'_m(T'_i, T'_f)$

são traduzidas para a seguinte restrição de integridade:

$$\begin{aligned} \Leftarrow\ & happens(E_1, T_i, T_f), \\ & ... \\ & happens(E_n, T_i, T_f), \\ & happens(E'_1, T'_i, T'_f), \\ & ... \\ & happens(E'_m, T'_i, T'_f), \\ & T_f > T'_i. \end{aligned}$$

Esta restrição de integridade significa que a ocorrência dos eventos tem de respeitar a relação de ordem definida.

Definição 22 *Proposições-o*

As proposições-o do tipo:

$A\ occurs\_in\ E(T_i, T_f)$

são traduzidas para as seguintes restrições de integridade:

$$\begin{aligned} happens(E, T_i, T_f) &\Leftarrow\ . \\ &\Leftarrow\ happens(E, T_i, T_f), \\ &\quad\ not\ act\_ev(E). \\ act\_ev(E) &\leftarrow\ act(E, A_1), ..., act(E, A_n). \end{aligned}$$

que significa que o evento E tem de existir e estar associado à expressão de acções $A_1, ..., A_n$.

Definição 23 *Proposições-v*

As proposições-v que relacionam fluentes com eventos (assumindo uma ordenação dos eventos em que $T_{1f} \leq ... \leq T_{nf}$):

$F\ after\ E_1(T_{1i}, T_{1f}), ..., E_n(T_{ni}, T_{nf})$

são traduzidas para a seguinte regra:

$$\begin{aligned} initiates(E_n, T_{nf}, F) \leftarrow\ & happens(E_1, T_{1i}, T_{1f}), \\ & \ldots \\ & happens(E_n, T_{ni}, T_{nf}), \\ & T_{1i} \leq \ldots \leq T_{ni}, \\ & T_{1f} \leq \ldots \leq T_{nf}. \end{aligned}$$

No caso das proposições do tipo:

$$\neg F\ after\ E_1(T_{1i}, T_{1f}), \ldots, E_n(T_{ni}, T_{nf})$$

serão traduzidas para regras

$$\begin{aligned} terminates(E_n, T_{nf}, F) \leftarrow\ & happens(E_1, T_{1i}, T_{1f}), \\ & \ldots \\ & happens(E_n, T_{ni}, T_{nf}), \\ & T_{1i} \leq \ldots \leq T_{ni}, \\ & T_{1f} \leq \ldots \leq T_{nf}. \end{aligned}$$

Conforme foi referido anteriormente a demonstração da completude e a coerência da tradução da linguagem A_{EC} para a programação em lógica estendida está apresentada no anexo A.

Foi, também, estabelecida uma tradução entre a linguagem A e a linguagem A_{EC} que demonstra que todas as proposições-v (que permitem aferir do valor de um dado fluente) que são suportadas em A, também o são em A_{EC}. Deste modo, é possível relacionar os resultados de abordagens diferentes sobre representação de acções com a abordagem proposta, desde que as abordagens referidas tenham já resultados teóricos em relação à linguagem A. Devido à sua extensão, apresenta-se a tradução proposta e a demonstração dos resultados no apêndice B.

Como exemplo, apresenta-se a tradução de uma regra do domínio do mundo dos blocos para regras de programação em lógica.

A regra

$$move\_to\_top\_of(X,Y)\ causes\ on(X,Y)\ if\ on\_hand(X),\ free(Y).$$

é traduzida para

$$\begin{aligned} enabled(E, T_i) \leftarrow\ & act(E, move\_to\_top\_of(X,Y)), \\ & holds\_at(on\_hand(X,Y), T_i), \end{aligned}$$

$$
\begin{aligned}
 & & holds\_at(free(Y), T_i). \\
initiates(E, T_f, on(X,Y)) & \leftarrow & happens(E, T_i, T_f), \\
 & & act(E, move\_to\_top\_of(X,Y)), \\
 & & holds\_at(on\_hand(X), T_i), \\
 & & holds\_at(free(Y), T_i).
\end{aligned}
$$

2.5 Atitudes

A representação das atitudes dos diversos agentes é um factor fundamental para apoiar uma correcta participação dos agentes em diálogos. Como atitudes fundamentais considero as crenças, os objectivos e as intenções.

A frase seguinte é demonstrativa da necessidade de representar e raciocinar sobre as atitudes dos agentes:

- Está demasiado calor!

Um agente-sistema computacional a quem esta informação seja comunicada deve ter a capacidade de representar não só as suas próprias atitudes (estados mentais segundo Pollack [Pol90]), mas também as suas crenças sobre as atitudes dos seus interlocutores. Neste exemplo, a crença de que o seu interlocutor acredita que está calor e que tem como objectivo alterar esse estado.

2.5.1 Operadores epistémicos

A representação das diversas atitudes é efectuada com base em operadores epistémicos. Os operadores básicos utilizados no âmbito deste trabalho são baseados no trabalho existente nesta área ([AP92, CMP90]) e permitem representar as atitudes básicas de um agente (as suas crenças, intenções e objectivos) e são os definidos em [AP92, CMP90] :

1. bel(a, p): o agente a acredita que a proposição p é verdadeira
2. int(a, α): o agente a tem a intenção de que a acção α seja executada
3. ach(a, p): o agente a tem como objectivo p, isto é, o agente a pretende que a proposição p se torne verdadeira como consequência das acções de algum agente (incluindo as suas próprias acções)
4. knowif(a, p)$\leftarrow$ bel(a,p).

   knowif(a,p)$\leftarrow$ bel(a,$\neg p$): o agente a acredita que conhece o valor lógico da proposição p

5. exp(a, p) $\leftarrow$ bel(a, p).

   exp(a, p)$\leftarrow$ ach(a, p): o agente a tem a expectativa de que a proposição p seja, ou venha a ser, verdadeira.

De notar que as intenções de um agente são as acções que ele pretende que sejam realizadas, enquanto os objectivos são os estados que ele pretende que se venham a atingir.

No âmbito deste trabalho os operadores epistémicos não são definidos como operadores modais, mas sim como predicados que serão (ou não) válidos no modelo bem fundado dos programas em lógica estendida que modelam os agentes (ver capítulo 3).

2.5.2 Regras de relação entre os operadores epistémicos

De modo a ser possível efectuar inferências sobre as atitudes dos agentes, é necessário definir uma teoria de atitudes que inclua as relações existentes entre elas e as restrições de integridade existentes. Com esta finalidade, proponho neste trabalho um conjunto de regras de programação em lógica estendida que permitem a representação desta teoria.

A definição completa da teoria de atitudes proposta é apresentada no capítulo 3, devido a estar inserida no processo mais global de modelação de agentes. No entanto, e de modo a ilustrar o processo de definição de relação entre atitudes, apresento nesta secção algumas restrições básicas de integridade e a regra de necessitação das crenças.

Definição 24 *Restrições de integridade*

O operador $bel/2$ possui uma restrição de integridade que representa a impossibilidade de um agente acreditar numa proposição e na sua negação:

$$\Leftarrow holds\_at(bel(A, P), T), holds\_at(bel(A, \neg P), T). \tag{2.16}$$

O operador $ach/2$ possui uma restrição de integridade que representa a impossibilidade de um agente pretender que uma proposição e a sua negação se tornem verdadeiras:

$$\Leftarrow holds\_at(ach(A, P), T), holds\_at(ach(A, \neg P), T). \tag{2.17}$$

É necessária, ainda, uma restrição de integridade que represente a inconsistência entre os operadores bel e ach:

$$\Leftarrow holds\_at(bel(A, P), T),\ holds\_at(ach(A, P), T). \tag{2.18}$$

De facto não é correcto ter uma situação em que um agente acredita que a proposição P é verdadeira e, simultaneamente, pretender que P se torne verdadeira.

Definição 25 *Regra de necessidade das crenças*

É necessário definir, ainda, regras que representem a coerência dos agentes:

$$holds\_at(bel(A, P), T) \leftarrow holds\_at(P, T). \tag{2.19}$$

Esta regra representa que todo o agente acredita no que é verdade e, em particular, nas tautologias.

Este tipo de regras introduz o problema de encaixamento infinito de crenças. Este problema é resolvido a nível de implementação, limitando o nível máximo desse encaixamento. Note-se que no domínio dos diálogos não é necessário definir um nível de recursividade de crenças superior a 2 (ver [TCM96]).

De notar que a regra de necessidade de crenças define, também, que um agente acredita nas suas atitudes, quer sejam crenças, objectivos ou intenções. A regra poderia ser refinada de modo a suportar a existência de intenções e crenças inconscientes, isto é, situações em que o agente não acredita que possui uma dada intenção ou crença por dificuldades de introspecção. Esta situação está fora do âmbito deste trabalho.

Há, no entanto, que distinguir estas situações de uma outra situação em que os agentes não acreditam na informação que estão a transmitir. Neste caso o agente é mentiroso, e esta é uma situação que será analisada no próximo capítulo.

O conjunto de atitudes e de regras de co-relação propostas nesta secção e no próximo capítulo são a base do processo de inferência das atitudes dos agentes. É, no entanto, necessário caracterizar correctamente os actos de fala em relação às atitudes que lhes estão subjacentes.

2.6 Actos de Fala

Num processo de participação em diálogos é, também, fundamental representar os actos de fala que estão subjacentes a cada intervenção.

Desde Austin e Searle ([Aus62, Sea90]) que tem sido referido que falar é agir e que, ao falar, os agentes estão a realizar actos de fala. Cohen e Perrault ([CP79]) definiram os actos de fala como acções cujos efeitos afectam tanto o emissor como o receptor. Posteriormente, Perrault ([Per90]) referiu que a partir da análise das frases não é possível determinar com exactidão os efeitos dos actos de fala, pois é necessário ter em conta o estado mental dos interlocutores. De facto, existe a necessidade de recorrer a raciocínio não-monótono que possibilite a alteração das atitudes do agente-sistema computacional perante um acto de fala, consoante a informação veiculada pelo referido acto.

A definição proposta nesta tese para os actos de fala permite ultrapassar os problemas existentes na abordagem de Allen e Litman [AL86, LA87] em que os actos de fala não eram definidos com base em operadores epistémicos e nas suas consequências em termos de estados mentais dos agentes. Por outro lado, ao integrar a componente temporal, resolve também alguns dos problemas das abordagens de Cohen e Perrault, que propuseram uma teoria geral de actos de fala, mas que não tem a capacidade de raciocinar adequadamente

sobre os referidos actos e os seus efeitos (efectuando introspecções sobre os efeitos dos actos de fala num dado instante de tempo).

De facto, a teoria de actos de fala proposta está integrada num ambiente que permite:

1. raciocínio não-monótono. Como o agente-sistema computacional não detém informação completa sobre o mundo que o rodeia e sobre os seus interlocutores, terá necessidade de assumir determinados pressupostos que, posteriormente, se podem vir a revelar incorrectos.

2. raciocínio temporal, que inclui a possibilidade de raciocinar sobre eventos passados e sobre as proposições válidas num dado instante de tempo. O agente deve ter a capacidade de fazer introspecção e analisar os eventos, as suas atitudes e a sua evolução ao longo do tempo.

3. raciocínio sobre as atitudes dos agentes. O agente deve poder actualizar as suas atitudes (terminando umas e iniciando outras) ou revê-las, em caso de detecção de inconsistências.

4. a modelação dos agentes, com as suas diferentes características. O agente deve ser capaz de modelar o seu comportamento e de possuir um modelo dos seus interlocutores.

Conforme já foi apresentado nas secções anteriores, a abordagem proposta nesta tese suporta os dois primeiros itens, sendo o terceiro e quarto item apresentados nos dois capítulos seguintes.

É de referir que não é objectivo primordial desta tese o reconhecimento dos diversos actos de fala subjacentes às frases em Língua Natural. É assumido que haverá um módulo autónomo que será responsável por essa funcionalidade. De modo idêntico, também no processo de geração dos actos de fala, não é analisado em detalhe o processo de conversão dos actos de fala planeados em frases em Língua Natural.

Em relação à análise das acções envolvidas num processo de diálogos entre vários interlocutores, e de acordo com Traum ([TH92, Tra94]), são necessários quatro níveis de acção para expressar o conteúdo e para manter a coerência das conversas:

1. Actos de argumentação — Elaboração, sumarização, clarificação, ...

2. Actos de fala — Informação, pedido, sugestão, pergunta SimNão, ...

3. Actos fundacionais (*Grounding*) — Iniciar, continuar, cancelar, confirmar, ...

4. Actos de mudança de emissor — Obter-vez, manter-vez, passar-vez, ...

Embora concordando com esta análise, no âmbito deste trabalho só irão ser analisados os actos de fala propriamente ditos, dado que o objectivo principal é analisar o processo

de transferência de informação entre os agentes e as suas consequências e pré-condições em termos dos seus estados mentais. Os restantes actos (argumentação, fundacionais e de mudança de emissor) são utilizados para a gestão do processo de diálogo ou para a organização da informação no processo de geração.

No âmbito deste trabalho defino os seguintes actos de fala em termos dos seus efeitos e pré-condições:

1. inform(s, h, p) : o emissor s informa o receptor h acerca da proposição p
2. informref(s, h, t, p) : o emissor s informa o receptor h acerca do termo t da proposição p
3. informif(s, h, p) : o emissor s informa o receptor h acerca do valor lógico da proposição p.
4. request(s, h, a) : o emissor s pede ao receptor h para realizar a acção a.
5. askif(s, h, p) : o emissor s pede ao receptor h para o informar acerca do valor lógico da proposição p.
6. checkif(s, h, p) : o emissor s pede ao receptor h para lhe confirmar o valor lógico da proposição p.
7. accept(s, h,a) : o emissor s aceita efectuar a acção a.
8. reject(s, h,a) : o emissor s rejeita efectuar a acção a.

Estes actos de fala foram definidos em termos das suas pré-condições e efeitos nos interlocutores e, posteriormente, foram descritos usando a linguagem A_{EC} definida anteriormente. De referir que assumi que os actos de fala são operadores de um interlocutor para outro interlocutor, e não de um para vários. Por outro lado, os actos de fala têm pré-condições e efeitos distintos consoante se trate do emissor ou do receptor do acto de fala. Existe um predicado auxiliar — $eu(X)$ — que define a identificação do agente que está a ser modelado. Conforme será apresentado em detalhe no próximo capítulo, cada agente é uma entidade autónoma que será modelado através de um programa em lógica estendida. Para agentes distintos haverão programas em lógica distintos que modelam os respectivos estados mentais. Em cada programa em lógica que modela um agente haverá predicados que permitem identificar os diversos agentes: $eu(X)$, $voce(X)$.

Como exemplo, suponhamos um diálogo entre dois agentes: o João e o Manuel. No programa em lógica que modela o João, os seguintes factos serão verdadeiros: $eu(joao)$, $voce(manuel)$.

No modelo do Manuel, os factos verdadeiros serão: $voce(joao)$, $eu(manuel)$.

Nesta secção será apresentada uma definição dos actos de fala que assume a existência de comportamentos confiáveis por parte dos agentes, isto é, os agentes transmitem somente

informação em que acreditam e são crédulos e cooperativos. No capítulo 3 serão analisadas situações mais complexas com diversos tipos de modelos de agente não tão *confiáveis*.

Assim, a definição dos actos de fala que proponho é a seguinte:

Definição 26 *Inform — Informação acerca de uma proposição*

Este acto de fala é representado por:

$$\begin{array}{lll} inform(S,H,P) & causes & bel(H,bel(S,P)) \\ & if & eu(H). \end{array}$$

$$\begin{array}{lll} inform(S,H,P) & causes & bel(S,bel(H,bel(S,P))) \\ & if & eu(S), \\ & & bel(S,P) \\ & & bel(S,int(S,inform(S,H,P))). \end{array}$$

Se um emissor acredita numa dada proposição e tem a intenção de informar o receptor sobre essa proposição, após o acto de informar, o receptor acreditará que o emissor acredita na proposição referida e o emissor acredita que o receptor acredita nisso.

Esta regra modela um comportamento correcto em que só se informa o que se acredita. No próximo capítulo irão ser feitas alterações a esta regra de modo a modelar comportamentos não tão correctos e que permitem que o modelo que o receptor tem sobre a proposição informada influencie as suas crenças sobre o emissor (eventualmente considerando-o como emissor mentiroso).

É de referir, no entanto, que o receptor não passou a acreditar na proposição que lhe foi dita; de acordo com o seu modelo mental, assim será o seu comportamento: crédulo, duvidoso, etc. Estes comentários são válidos para todos os actos de fala que serão apresentados.

De modo a exemplificar o processo de tradução utilizado, apresento de seguida o resultado da tradução deste acto de fala para regras de programação em lógica. Esta tradução não será, no entanto, apresentada para os restantes actos de fala.

A tradução do acto de fala *inform* para regras de programação em lógica é, de acordo com a tradução proposta:

$$\begin{array}{rcll} enabled(E,T_i) & \leftarrow & act(E,inform(S,H,P)), & \\ & & eu(H). & \\ initiates(E,T_f,bel(H,bel(S,P))) & \leftarrow & happens(E,T_i,T_f), & (2.20) \\ & & act(E,inform(S,H,P)), & \\ & & eu(H). & \end{array}$$

$$
\begin{aligned}
enabled(E, T_i) \leftarrow\ & act(E, inform(S, H, P)), \\
& eu(S), \\
& holds\_at(bel(S, P), T_i), \\
& holds\_at(bel(S, int(S, inform(S, H, P))), T_i).
\end{aligned}
$$

$$
\begin{aligned}
initiates(E, T_f, bel(S, bel(H, bel(S, P)))) \leftarrow\ & happens(E, T_i, T_f), \\
& act(E, inform(S, H, P)), \\
& eu(S), \\
& holds\_at(bel(S, P), T_i) \\
& holds\_at(bel(S, int(S, inform(S, H, P))), T_i).
\end{aligned} \tag{2.21}
$$

De notar que, de acordo com as regras de tradução apresentadas na secção anterior, onde está $eu(X)$ deveria estar $holds\_at(eu(X), T)$. No entanto, dado que o predicado $eu(X)$ não depende de um factor temporal, assumiu-se esta simplificação.

Assume-se, nesta situação, que existe no programa em lógica que modela o agente, o seguinte facto:

$$eu(a).$$

onde a é o identificador do agente.

Caso não se efectuasse esta simplificação, deveria ser criado um evento inicial, anterior a qualquer outro evento, com as seguintes características:

$$
\begin{aligned}
& happens(e_0, t_0, t_0). \\
& act(e_0, start). \\
& initiates(e_0, t_0, eu(a)).
\end{aligned}
$$

onde $\forall t,\ t_0 < t$.

Definição 27 *Informref — Informação acerca de um termo de uma proposição*

Este acto de fala é representado por:

$$
\begin{aligned}
inform ref(S, H, T, P) \quad causes \quad & bel(H, bel(S, ref(T, P))) \\
if \quad & eu(H).
\end{aligned}
$$

$$
\begin{aligned}
inform ref(S, H, T, P) \quad causes \quad & bel(S, bel(H, bel(S, ref(T, P)))) \\
if \quad & eu(S), \\
& bel(S, ref(T, P)) \\
& bel(S, int(S, inform ref(S, H, T, P)).
\end{aligned}
$$

Se um agente acredita que T é um termo da proposição P e pretende informar o receptor sobre essa propriedade, então o receptor passará a acreditar que o emissor acredita que T é um termo da propriedade P e o emissor acreditará que o receptor acredita nisso.

Definição 28 *InformIf — Informação acerca do valor lógico de uma proposição*

Este acto de fala é representado por:

$$\begin{array}{lll} informif(S,H,P) & causes & bel(H, knowif(S,P)) \\ & if & eu(H). \end{array}$$

$$\begin{array}{lll} informif(S,H,P) & causes & bel(S, bel(H, knowif(S,P))) \\ & if & eu(S), \\ & & bel(S, knowif(S,P)) \\ & & bel(S, int(S, informif(S,H,P))). \end{array}$$

Se um emissor conhecer o valor lógico de uma proposição (verdadeiro ou falso) e tem a intenção de informar o receptor sobre esse valor lógico, então, após o acto de informar, o receptor acreditará que o emissor acredita no valor lógico referido e o emissor acreditará que o receptor acredita nisso.

Definição 29 *Request — Pedido para efectuar uma dada acção*

A definição deste acto de fala é:

$$\begin{array}{lll} request(S,H,A) & causes & bel(H, int(S,A)) \\ & if & eu(H). \end{array}$$

$$\begin{array}{lll} request(S,H,A) & causes & bel(S, bel(H, int(S,A))) \\ & if & eu(S), \\ & & bel(S, int(S,A)), \\ & & bel(S, int(S, request(S,H,A))). \end{array}$$

Se um emissor s tem como intenção que a acção a seja realizada e se tem a intenção de pedir ao receptor h para efectuar a acção, então, após o pedido, o receptor acredita que é intenção do emissor que a acção se realize e o emissor acredita que o receptor acredita nisso.

Definição 30 *AskIf — Pedido sobre o valor lógico de uma proposição*

Este acto de fala é representado por:

$$\begin{array}{lll} askif(S,H,P) & causes & bel(H, ach(S, knowif(S,P))) \\ & if & eu(H). \end{array}$$

$$\begin{array}{lll} askif(S,H,P) & causes & bel(S, bel(H, ach(S, knowif(S,P)))) \\ & if & eu(S), \\ & & \neg knowif(S,P), \\ & & bel(S, int(S, askif(S,H,P))). \end{array}$$

Se um emissor desconhece o valor lógico de uma proposição e tem a intenção de perguntar ao receptor esse valor lógico, então, após o acto de perguntar, o receptor acreditará que o emissor tem o objectivo de saber esse valor lógico e o emissor acreditará que o receptor acredita nesse facto.

Dada a utilização da negação explícita nesta regra, apresento também a tradução para regras de programação em lógica:

$$\begin{array}{rcll} enabled(E,T_i) & \leftarrow & act(E, askif(S,H,P)), & (2.22) \\ & & eu(H). & \\ initiates(E,T_f, bel(H, ach(S, knowif(S,P)))) & \leftarrow & happens(E,T_i,T_f), & (2.23) \\ & & act(E, askif(S,H,P)), & \\ & & eu(H). & \end{array}$$

$$\begin{array}{rcll} enabled(E,T_i) & \leftarrow & act(E, askif(S,H,P)), & (2.24) \\ & & eu(S), & \\ & & holds\_at(\neg knowif(S,P), T_i), & \\ & & holds\_at(bel(S, int(S, askif(S,H,P))), T_i). & \end{array}$$

$$\begin{array}{l} initiates(E,T_f, bel(S, bel(H, ach(S, knowif(S,P))))) \leftarrow \quad (2.25) \\ \quad happens(E,T_i,T_f), \\ \quad act(E, askif(S,H,P)), \\ \quad eu(S), \\ \quad holds\_at(\neg knowif(S,P), T_i), \\ \quad holds\_at(bel(S, int(S, askif(S,H,P))), T_i). \end{array}$$

Definição 31 *CheckIf — Pedido sobre a confirmação do valor lógico de uma proposição*

Este acto de fala tem grandes semelhanças com o $askif$, distinguindo-se por pretender somente confirmar o valor lógico da proposição. É representado por:

$$\begin{array}{lll} checkif(S,H,P) & causes & bel(H, ach(S, knowif(S,P))) \\ & if & eu(H). \end{array}$$

$$\begin{array}{lll} checkif(S,H,P) & causes & bel(S, bel(H, ach(S, knowif(S,P)))) \\ & if & eu(S), \\ & & bel(S, int(S, checkif(S,H,P))). \end{array}$$

Se um emissor tem a intenção de validar com o receptor o valor lógico de uma proposição, então, após o acto de perguntar, o receptor acreditará que o emissor tem o objectivo de validar esse valor lógico e o emissor acreditará que o receptor acredita nesse facto.

Definição 32 *Accept — Aceitar efectuar uma determinada acção*

Este acto de fala é representado por:

$$\begin{array}{lll} accept(S,H,A) & causes & bel(H, int(S,A)) \\ & if & eu(H). \end{array}$$

$$\begin{array}{lll} accept(S,H,A) & causes & bel(\mathrm{S}, bel(H, int(S,A))) \\ & if & eu(S), \\ & & bel(S, int(S,A)), \\ & & bel(S, int(S, accept(S,H,A))). \end{array}$$

Se um emissor tem a intenção de que uma acção se realize e comunica essa intenção, então o receptor acredita na intenção comunicada e o receptor acredita que o emissor acredita nesse facto.

Definição 33 *Reject — Rejeitar efectuar uma determinada acção*

Este acto de fala é representado por:

$$\begin{array}{lll} reject(S,H,A) & causes & bel(H,\neg int(S,A)) \\ & if & eu(H). \end{array}$$

$$\begin{array}{lll} reject(S,H,A) & causes & bel(\mathrm{S},bel(H,\neg int(S,A))) \\ & if & eu(S), \\ & & bel(S,\neg int(S,A)), \\ & & bel(S,int(S,reject(S,A))). \end{array}$$

Se um emissor não tem a intenção de que uma acção se realize e comunica esse facto, então o receptor acredita na intenção comunicada e o receptor acredita que o emissor acredita nesse facto.

2.7 Conclusões

Neste capítulo foram apresentadas as ferramentas de suporte ao processo de participação em diálogos.

Em primeiro lugar apresentou-se o paradigma utilizado como base: programação em lógica estendida com semântica bem fundada. Este ambiente permite a representação do conhecimento requerido e suporta raciocínios não monótonos. Existe, ainda, um procedimento formal de prova coerente e completo que permite obter valores lógicos para os literais e efectuar a revisão de programas contraditórios.

De seguida, apresentou-se o formalismo de representação de eventos de suporte ao raciocínio temporal e à descrição de acções. O formalismo utilizado é o resultado de alterações que proponho para o Cálculo de Eventos, de modo a suportar acções não instantâneas e concorrência.

Na secção seguinte foi apresentada uma linguagem de alto nível A_{EC} para representação de acções, a partir da linguagem A de Gelfond e Lifschitz. Foi definida a semântica dessa linguagem e foi apresentada uma tradução completa e coerente com a linguagem de programação em lógica estendida. Mencionou-se, ainda, uma tradução entre a linguagem A e a A_{EC}, que devido à sua extensão se apresenta em apêndice.

Na secção seguinte foi efectuada uma descrição das atitudes representativas dos estados mentais dos agentes, bem como o estabelecimento de relações entre elas.

Finalmente, foi apresentada uma proposta de representação de actos de fala, utilizando os mecanismos anteriormente descritos, a qual possibilita a criação de raciocínios e de inferências sobre os estados mentais subjacentes aos actos de fala (capítulo 4).

As teorias propostas neste capítulo servem de base ao processo de inferência de atitudes em diálogos descrito nos capítulos seguintes.

Capítulo 3

Modelação de agentes

Neste capítulo proponho um processo de modelação dos agentes participantes em diálogos, recorrendo a alguns dos mecanismos apresentados no capítulo anterior. Nomeadamente, um agente/sistema computacional é modelado através de um sextuplo constituído por programas em lógica estendida que representam as suas atitudes, o seu conhecimento do mundo, a descrição das acções que podem ser executadas, o processo de raciocínio temporal que o agente utiliza, as regras que definem o seu comportamento, e as regras que definem a sua racionalidade.

O processo proposto permite modelar agentes com diferentes características, desde o agente cooperativo que pretende satisfazer os objectivos dos seus interlocutores, até aos agentes mentirosos que pretendem enganá-los, passando pelos agentes reactivos/pró-activos que apresentam comportamentos distintos durante os diálogos.

As capacidades de representação do sistema proposto são apresentadas e é efectuada uma comparação com os sistemas de diálogos existentes.

Finalmente, demonstro como é possível simular ambientes multiagentes de diferentes características utilizando o sistema proposto.

3.1 Introdução

Um agente/sistema computacional, para participar em diálogos, necessita de modelar adequadamente o seu estado mental, isto é, necessita de representar as suas atitudes, o conhecimento que possui sobre o mundo que o rodeia e as regras que definem o seu comportamento e a sua racionalidade. De facto, a participação em diálogos depende directamente das características dos agentes participantes, sendo este facto um factor que condiciona o decorrer do próprio diálogo. Por exemplo, se um determinado agente tem por característica informar que determinadas propriedades são válidas, não o sendo na realidade (agente mentiroso), então o agente/sistema computacional deverá ter um comportamento adequado: deixando de acreditar na informação que aquele agente transmite. Ou seja, uma das componentes essenciais do modelo de um agente/sistema computacional é o conjunto das suas crenças em relação às atitudes dos outros agentes: o seu modelo dos outros agentes.

A abordagem proposta neste trabalho considera cada agente como uma entidade autónoma, caracterizada pelos seus estados mentais e capaz de realizar inferências sobre esses estados e de actualizá-los de acordo com nova informação veiculada através dos diversos actos de fala.

Recorrendo a uma analogia com sistemas informáticos, um agente pode ser modelado por um processo que corre num determinado computador e que interage com outros agentes (sejam eles humanos ou processos nesse ou noutros computadores). De facto, para cada agente, é indiferente a *composição interna* dos outros agentes. O fundamental é existir um meio de comunicação bem definido que possibilite a interacção entre eles. No âmbito deste trabalho esse meio de comunicação é constituído pelos diversos actos de fala. No capítulo 6 são apresentados exemplos da aplicação deste conceito a diversos diálogos. Este trabalho poderá, ainda, ser utilizado como suporte a sistemas multi-agentes para participação em diálogos ou em outro tipo de interacções.

Na próxima secção apresento uma análise dos principais sistemas de diálogos existentes e das suas limitações em relação à capacidade de modelar agentes. Estes sistemas, ao contrário do que proponho, não têm a capacidade de tornar independentes os processos de modelação dos agentes e de raciocínio. Não é possível, nesses sistemas, alterar o desenrolar de um diálogo através da modificação das regras que definem o comportamento do agente/sistema computacional: seria necessário alterar o próprio sistema de inferência.

Na secção 3.3 proponho uma estrutura que permite modelar o agente/sistema computacional em relação às seguintes características:

1. Representação das suas atitudes (crenças, intenções e objectivos).

   As atitudes do agente são representadas associadas ao factor tempo, isto é, é possível obter quais as suas atitudes num determinado instante de tempo. Estas atitudes definem o estado mental do agente nesse instante de tempo e definem o modelo que ele possui sobre os outros agentes.

2. Representação de conhecimento sobre o mundo que o rodeia.

   O agente possui conhecimento sobre o mundo que o rodeia: factos, conceitos, observações efectuadas. Este conhecimento é representado por regras de programação em lógica estendida que, conforme argumentei no capítulo anterior, é um processo adequado para a representação de conhecimento.

3. Representação das acções passíveis de serem executadas

   As acções passíveis de serem executadas devem ser descritas através de regras de programação em lógica estendida. De facto, elas são descritas em primeiro lugar na linguagem A_{EC} e, posteriormente, são traduzidas para regras de programação em lógica estendida.

4. Definição do processo de raciocínio temporal.

   Os axiomas temporais propostos no capítulo anterior permitem a identificação das propriedades válidas em cada instante de tempo.

5. Definição da sua racionalidade.

   As atitudes do agente, crenças, intenções e objectivos, estão relacionadas entre si de acordo com regras que definem a sua racionalidade. Na secção 3.4 é proposta uma teoria de atitudes que permite captar a noção de racionalidade. A incorporação explícita de uma teoria de racionalidade de agentes num sistema de diálogos é uma característica inovadora deste trabalho.

6. Definição do seu comportamento

   O comportamento dos agentes é condicionado por diversas propriedades. No âmbito deste trabalho consideraram-se essenciais as seguintes (analisadas e descritas na secção 3.5):

   - Tipo de actividade (pró-activo vs reactivo)
   - Sinceridade (verdadeiro vs mentiroso)
   - Credulidade (crédulo vs céptico)
   - Cooperatividade (cooperativo vs não cooperativo)

   Estas características permitem modelar um conjunto bastante diverso de modelos mentais e simular comportamentos bastante distintos. De facto, se considerarmos que para cada propriedade pode haver pelo menos dois comportamentos distintos, poderemos modelar dezasseis tipos de comportamento. Dentro destes comportamentos há que realçar os típicos de sistemas de pesquisa de informação (agentes reactivos, sinceros, crédulos e cooperativos) e, num extremo oposto, o comportamento em que o agente é pró-activo, mentiroso, céptico e não cooperativo. Este é o típico comportamento do vulgar *aldrabão* que participa nos diálogos com o objectivo

de activamente enganar os outros agentes e sem qualquer intenção de cooperar com eles.

A abordagem proposta permite modelar comportamentos que se alteram ao longo do tempo, dado que as características desse comportamento dependem do factor temporal. Por exemplo, um agente a é sincero num dado instante de tempo t se se verificar (ver secção 3.5):

$$holds\_at(bel(a, sincero(a)), t).$$

A capacidade de modelar comportamentos e de os alterar dinamicamente ao longo do tempo é uma característica inovadora de sistemas de participação em diálogos e permite lidar com um conjunto de situações bastante vasta. No entanto, no âmbito desta tese não são analisadas detalhadamente as situações que poderão levar a uma alteração comportamental, sendo somente descritas situações que poderão levar um agente a considerar outro como não verdadeiro (ver secção 3.5).

Na secção 3.6.1 é apresentado um exemplo em que diversos agentes (com modelos de comportamento distintos) interagem entre si.

Finalmente, na secção 3.7, são sumarizadas as principais vantagens do processo de modelação de agentes descrito neste capítulo.

3.2 Sistemas actuais

O processo de modelação do agente/sistema computacional nas suas diversas componentes, nomeadamente a representação das suas atitudes e regras de racionalidade e de comportamento, define algumas das características fundamentais de um sistema de diálogos:

1. A capacidade de modelar agentes com diferentes comportamentos;

2. A capacidade de autonomizar a representação do modelo do agente do processo de inferência de atitudes.

3. A capacidade de raciocinar sobre crenças contraditórias por parte dos agentes;

O sistema proposto neste capítulo permite modelar agentes com diferentes comportamentos (secção 3.4). Esta modelação é efectuada através de regras de programação em lógica que condicionam o processo de inferência de atitudes, mas que não fazem parte integrante desse processo. Deste modo é possível suportar uma diversidade de situações, de diálogos e de comportamentos. Note-se que está contemplada a possibilidade do agente alterar dinamicamente o seu comportamento de acordo com as situações, isto é, por exemplo, um agente crédulo poderá tornar-se céptico relativamente a outro agente ao longo do tempo.

O processo proposto para modelar agentes permite, também, raciocinar sobre crenças contraditórias por parte dos agentes, pois permite representar as crenças que o agente possui sobre os outros agentes, nomeadamente sobre as suas atitudes. Deste modo, é possível representar a crença do agente na veracidade de uma dada proposição e, simultaneamente, a sua crença de que outro agente crê na falsidade da mesma proposição.

Os principais sistemas de diálogos existentes actualmente não possuem estas três características, tendo sido desenhados de modo a suportar um só tipo de diálogo, nomeadamente os diálogos cooperativos com busca de informação num determinado domínio. São, portanto, agentes tipicamente crédulos, cooperativos, reactivos e sinceros.

Está nesta situação o sistema proposto por Litman [Lit85, LA87] em que o modelo dos agentes está incorporado no sistema de inferência. O principal objectivo é simular uma situação tipo *quiosque*, em que um utilizador interage com um sistema, informático ou não, de modo a obter informação adicional sobre uma dada área de conhecimento. O sistema de Litman não prevê a alteração do modelo subjacente aos agentes. Pelo contrário, o modelo dos agentes é uma característica do sistema e a sua alteração implicaria alterações profundas e de difícil formalização desse mesmo sistema.

Por outro lado, o sistema de Litman não permite a representação do estado mental do agente, nomeadamente do seu modelo dos outros agentes. Esta característica implica que o sistema de Litman, ao contrário do proposto nesta tese, não consegue lidar com algumas situações de erros em diálogos, como os planos incorrectos (os agentes possuem crenças contraditórias sobre como atingir determinados objectivos).

O sistema proposto por Carberry [Car85, Car88] tem características semelhantes, embora permita a existência de alguns tipos de planos incorrectos. No entanto, também não permite a existência de modelos distintos de agentes nem a alteração dos seus modelos.

A abordagem proposta por Pollack [Pol90, AP92], embora seja baseada na descrição das atitudes dos agentes, também não faz a separação explícita entre as atitudes dos agentes e as regras que definem o seu comportamento. Deste modo não permite modelar diversos comportamentos através da alteração dos *esquemas mentais* dos agentes (regras de comportamento). O esquema mental dos agentes (regras de comportamento) está incluído no processo de inferência.

A abordagem de Grosz e Sidner ([GS90]) considera, também, um ambiente de colaboração e cooperação para a obtenção de objectivos comuns. Nesta abordagem o tipo de comportamento é, tal como nos anteriores, intrínseco ao sistema de interpretação de diálogos.

3.3 Modelação dos agentes

Um agente pode ser caracterizado através de uma estrutura que contemple as suas atitudes, as suas regras de comportamento (ou esquemas mentais), a sua descrição das acções passíveis de serem realizadas, os seus axiomas temporais e o seu conhecimento do mundo.

Note-se que não se pretende com esta proposta definir modelos de agente suficientemente genéricos para permitir modelar agentes gerais, mas pretende-se, somente, definir agentes que têm como objectivo participar activa e intencionalmente em diálogos.

Assim, e de acordo com a minha proposta, um agente a dialogante é modelado através de um tuplo $< At, RC, Ac, T, Rr, CM >$ que é constituído por:

1. O conjunto das suas atitudes, At.

   - Este conjunto, que define o *estado mental* do agente a em cada instante de tempo, é:

$$x \in At \Leftrightarrow x = holds\_at(int(a, A), T) \vee$$
$$x = holds\_at(bel(a, P), T) \vee$$
$$x = holds\_at(ach(a, P), T).$$

2. As regras, RC, que definem o seu comportamento: actividade, cooperatividade, credulidade e sinceridade. Estas regras são representadas por relações das atitudes entre si e das atitudes com os actos de fala (ver secção 3.5).

   - RC é o conjunto de regras em programação em lógica estendida que define o comportamento do agente em relação ao meio que o rodeia. Estas regras permitem definir:
     (a) Actividade — Que acções pretende o agente que sejam realizadas;
     (b) Cooperatividade — Qual a relação entre as intenções de um agente e as de outro agente;
     (c) Sinceridade — Que tipo de informação é veiculada pelo agente;
     (d) Credulidade — Qual o grau de credulidade na informação que lhe é transmitida pelos outros agentes.

3. A descrição das acções, Ac, que o agente conhece e pode executar.

   - Ac é uma descrição das acções passíveis de serem efectuadas pelos agentes e é definida por um conjunto de regras em programação em lógica estendida obtidas a partir da tradução de regras do seguinte tipo, definidas na linguagem A_{EC} ,apresentada no capítulo anterior:

$$A \; causes \; F \; if \; P_1, ..., P_n$$

     Estas regras permitem caracterizar as acções em termos de pré-condições e de efeitos.

4. Um sistema, T, de axiomas temporais que permite a inferência de propriedades em cada instante de tempo.

- Este sistema é constituído pelos axiomas temporais apresentados, 2.1 a 2.15.

5. Regras de racionalidade, Rr, que relacionam as atitudes, definindo restrições de integridade e permitindo suportar a inferência de novas atitudes com base nas já existentes (secção 3.4). Como exemplo, temos uma restrição de integridade que define a impossibilidade de um agente acreditar numa proposição e na sua negação.

6. Uma descrição, CM, do mundo que o rodeia.

   - Esta descrição é composta por factos e regras em programação em lógica estendida que pretendem representar o conhecimento adquirido pelo agente sobre o mundo que o rodeia. Como exemplo, teríamos a representação de uma taxonomia de conceitos e propriedades associadas: os pinguins são aves; as aves, que não pinguins, voam, etc.

Um agente definido desta forma integra, numa estrutura de informação, os componentes essenciais para uma correcta participação em diálogos (ou interacções mais gerais) com outros agentes.

Conforme será apresentado no capítulo 7, o tuplo que representa o modelo do agente não é mais do que um programa em lógica estendida que, através do recurso à semântica bem fundada, permite inferir as propriedades que são válidas em cada instante. Para tal, são calculados os modelos referentes aos programas que descrevem o estado mental do agente num determinado instante de tempo. É com base neste modelo que é possível descrever o estado mental do agente e inferir as acções a realizar.

Nas secções seguintes serão analisadas em detalhe as regras que definem a racionalidade e o comportamento dos agentes. Das restantes componentes que integram a estrutura de cada agente, as suas atitudes e os axiomas temporais já foram apresentadas no capítulo 2. Por outro lado, as acções do domínio e o conhecimento do mundo são dependentes de cada área a representar e serão apresentadas em conjunto com os diversos exemplos.

3.4 Regras de Racionalidade

As regras de racionalidade de cada agente definem as relações existentes entre as diversas atitudes (crenças, intenções e objectivos).

O processo de representação de regras de racionalidade proposto nesta secção está directamente relacionado com a teoria de atitudes proposta por Cohen e Levesque [CL90a] que pretende definir as propriedades dessas atitudes (consistência, persistência, etc.). A minha proposta define essas propriedades em termos de regras de programação em lógica estendida. A abordagem proposta tem como vantagem fundamental a sua integração na estrutura que define o modelo do agente/sistema computacional, tornando as regras de racionalidade dependentes do agente a modelar. Além disso, esta teoria de atitudes está integrada num sistema formal que permite a obtenção da semântica do programa que

modela o agente (semântica bem fundada) e, em consequência, das atitudes válidas em cada instante.

No capítulo 2 foram já apresentadas alguns operadores epistémicos que definem relações entre atitudes (regras 2.16 a 2.19). No entanto, e dado o carácter introdutório do capítulo, foram apresentadas somente as regras necessárias a um processo básico de inferência de atitudes. Um sistema que possibilite a modelação da racionalidade dos agentes necessita de uma teoria mais aprofundada de suporte à inferência de atitudes. Nesta secção será apresentada uma proposta que irá estender as regras de coerência apresentadas anteriormente.

No âmbito deste trabalho foi assumido que os agentes a modelar teriam um comportamento racional. Isto é, as suas atitudes ao longo do tempo estariam de acordo com regras de raciocínio perfeitamente definidas. Não são analisados agentes cujo estado mental não seja suportado por nenhum processo de raciocínio, como, por exemplo, um agente que adopte uma determinada crença sem a existência de nenhuma facto que a suporte.

É, portanto, necessário definir as regras que definem as propriedades das crenças, das intenções e dos objectivos, bem como as relações entre estas atitudes.

3.4.1 Crenças

As crenças são uma componente fundamental das atitudes de um agente e suportam a criação de outras atitudes. Num dado instante de tempo, as crenças de um agente a são as propriedades que são válidas no seu modelo

$$holds\_at(bel(a, P), T)$$

e representam as propriedades em que ele acredita nesse instante de tempo. Nesta abordagem, as crenças não correspondem a informação não factual ou hipotética como, por exemplo, propriedades que sejam verdadeiras em alguns modelos do agente. De facto, as crenças de um agente, $bel(a, P)$, indexam as propriedades em que o agente acredita e, em termos de implementação, podem ser representadas, por exemplo, pela propriedade P_a. Deste modo, a construção de uma teoria de atitudes encontra-se simplificada, pois não é necessário definir regras explícitas para grande parte dos axiomas específicos para as crenças dos agentes.

Vejamos como as propriedades identificadas por Cohen e Levesque [CL90a] como necessárias para a representação das crenças, se encontram satisfeitas:

- Integridade

  $$\bot \; if \; bel(A, P), \; bel(A, \neg P). \tag{3.1}$$

  É impossível um agente acreditar simultaneamente numa propriedade e na sua negação. Note-se que esta propriedade, bem como as seguintes, está descrita recorrendo à linguagem A_{EC} proposta no capítulo anterior.

A integridade das crenças é garantida pela restrição de integridade 2.3, apresentada no capítulo 2 (tendo em conta que $bel(A, P)$ pode ser representado por P_a).

- Consistência

$$\neg bel(A, \neg P)\ if\ bel(A, P). \tag{3.2}$$

Se um agente acredita numa proposição então esse agente não acredita na sua negação.

A consistência das crenças resulta da regra de integridade:

$$holds\_at(\neg bel(A, \neg P), T) \leftarrow holds\_at(bel(A, P), T)$$

- Necessidade

Se uma dada propriedade é válida no modelo de um dado agente, então ele acredita nela.

Esta regra de inferência é modelada através da regra de programação em lógica:

$$holds\_at(bel(A, P), T) \leftarrow holds\_at(P, T). \tag{3.3}$$

Note-se que esta regra cria problemas de ciclos infinitos que sao resolvidos a nível de implementação (ver capítulo 7).

- Fecho

Esta propriedade significa que se um agente acredita numa proposição e se acredita que essa proposição implica uma outra, então ele acredita também na outra. É a regra K da lógica modal:

$$(P \rightarrow Q) \rightarrow (bel(a, P) \rightarrow bel(a, Q))$$

Esta regra é, também, representada com recurso à linguagem A_{EC} e deverá dar origem à criação de uma nova regra para cada implicação lógica $Q \leftarrow P$ que se deseja modelar:

$$holds\_at(bel(a, Q), T)\ \leftarrow holds\_at(bel(a, P), T). \tag{3.4}$$

- Introspecção positiva

$$\bot if\ bel(A, P), bel(A, \neg bel(A, P)). \tag{3.5}$$

É impossível um agente acreditar simultaneamente numa proposição e, simultaneamente, acreditar que acredita na sua negação.

A introspecção positiva é garantida do seguinte modo:

1. Pela regra 3.3:

$$bel(A, bel(A, P))\ if\ bel(A, P)$$

2. Pela consistência das crenças:

$$\neg bel(A, \neg bel(A, P))\ if\ bel(A, bel(A, P))$$

3. Ou seja,

$$\neg bel(A, \neg bel(A, P))\ if\ bel(A, P)$$

4. Por manipulação lógica:

$$\bot if\ bel(A, P), bel(A, \neg bel(A, P)). \tag{3.6}$$

- Introspecção negativa

$$\bot if\ \neg bel(A, P), bel(A, bel(A, P)). \tag{3.7}$$

É impossível um agente não acreditar numa proposição e acreditar que acredita nela.

A introspecção negativa é garantida através de um processo idêntico ao anterior.

- Persistência

Ao longo do tempo, um agente continua a acreditar nas suas crenças, caso não exista nenhum evento que altere essas crenças.

Esta regra é suportada pelos axiomas temporais já apresentados (*frame axioms*), pelo que não necessita de axiomas adicionais.

Com base nestas regras é possível inferir relações entre as crenças de um agente. É, no entanto, necessário ainda relacioná-las com as outras atitudes (intenções e objectivos, a analisar em detalhe nas secções seguintes):

- Introspecção positiva

$$\bot if\ int(A, \alpha), \neg bel(A, int(A, \alpha)). \tag{3.8}$$

$$\bot if\ ach(A, P), \neg bel(A, ach(A, P)). \tag{3.9}$$

Um agente não pode ter crenças contrárias às suas próprias atitudes.

A introspecção positiva em relação às intenções e aos objectivos é garantida de um modo análogo à introspecção em relação às crenças.

- Introspecção negativa

$$\bot if\ \neg int(A, \alpha), bel(A, int(A, \alpha)). \tag{3.10}$$

$$\bot if\ \neg ach(A, P), bel(A, ach(A, P)). \tag{3.11}$$

Novamente, um agente não pode ter crenças em atitudes que não pertencem ao seu modelo.

A introspecção negativa também é garantida de um modo análogo à introspecção negativa de crenças.

O conjunto destas regras deverá ser traduzido para regras de programação em lógica estendida, de acordo com o processo apresentado no capítulo anterior. Estas regras deverão ser adicionadas à componente *Rr* descritiva da racionalidade do agente.

3.4.2 Intenções

As intenções de um agente, $int(a, \alpha)$, são outra das componentes fundamentais das atitudes, dado serem a base do processo de interacção com os outros agentes. De facto, é a partir das intenções que é efectuada a inferência e a geração das acções a realizar pelos agentes. São, também, as intenções de cada agente que definem o seu modo de comportamento para com o mundo que o rodeia, pois representam as acções que o agente pretende que venham a ser realizadas.

Num dado instante de tempo, as intenções de um agente a são as propriedades que são válidas no seu modelo

$$holds\_at(int(a, \alpha), T)$$

As propriedades que as intenções devem possuir e as suas relações com as outras atitudes tem sido objecto de análise por diversos investigadores ([Bra90, CL90a, CL90b, Per90]). É, no entanto, relativamente consensual que um agente racional deve observar as seguintes propriedades:

- Persistência

Ao longo do tempo, um agente continua a manter as suas intenções, caso não exista nenhum evento que altere essas intenções (*frame axiom*). A noção de persistência de intenções (*commitment*) é fundamental para o processo de gestão de interacções. Um agente deve manter as suas intenções de que determinadas acções sejam realizadas, enquanto elas forem consistentes com o seu *estado mental*.

Esta propriedade é suportada pelos axiomas temporais do Cálculo de Eventos, pelo que não necessita de axiomas adicionais.

- Intencionalidade das acções

  Para cada regra de descrição de acções,

  $$\alpha \; causes \; p \; if \; Conds$$

  é criada a seguinte regra:

  $$ach(A, P) \; if \; int(A, \alpha), Conds \tag{3.12}$$

Se um agente tem como intenção realizar uma acção que tem determinados efeitos, então ele pretende que esses efeitos se venham a concretizar.

- Consistência

As intenções de um agente devem ser consistentes entre si em cada instante de tempo. Esta propriedade é descrita através da conjunção da consistência dos objectivos a atingir (ach) e da intencionalidade das acções. Isto é, não é possível um agente ter como intenção realizar duas acções distintas que causam efeitos contrários, dado que é requirida a consistência dos objectivos a atingir (sub-secção seguinte). Por outro lado, o teste da consistência do modelo do agente é garantida pela semântica do programa em lógica que representa o agente. Neste sentido, não é necessária nenhuma regra adicional para representar esta propriedade.

Note-se que seria possível um agente, num dado instante de tempo, ter intenções contrárias para momentos de tempo diferentes. Por exemplo, o agente pode, num dado instante, querer abrir uma janela a uma dada hora (por estar calor) e querer fechar fechar essa janela a uma hora posterior (por já estar frio). No entanto, o formalismo de representação de intenções proposto não permite captar esta noção, dado não associar a uma intenção o intervalo de tempo em que se pretende vir a executar a acção. Esta característica poderá vir a ser objecto de trabalho futuro, com o objectivo de permitir representar situações mais complexas.

3.4.3 Objectivos

Os objectivos são, também, uma componente fundamental das atitudes e estão relacionados directamente com as intenções. Neste trabalho, as intenções de um agente são as acções que ele pretende que sejam realizadas, enquanto que os objectivos são os estados que ele pretende que se venham a atingir. Conforme descrito no capítulo anterior, a atitude $ach(a, p)$ representa que o agente a pretende que o proposição p se venha a verificar, ou seja, representa um objectivo de a. Tal como as crenças, os objectivos de um agente podem ser representados por propriedades do tipo $p\_\_a$.

Na sub-secção anterior, já foi referida a necessidade de manter a consistência entre os diversos objectivos em cada instante de tempo. Existem, no entanto, outras propriedades necessárias a uma correcta inferência dos objectivos e das atitudes relacionadas:

- Integridade

  $$\bot \; if \; ach(A, P), ach(A, \neg P). \tag{3.13}$$

  É impossível um agente ter, simultaneamente, como objectivo uma propriedade e a sua negação.

  A integridade dos objectivos é garantida pela restrição de integridade 2.17, apresentada no capítulo 2.

- Consistência

  $$\neg ach(A, \neg P) \; if \; ach(A, P). \tag{3.14}$$

  Se um agente tem como objectivo que uma proposição seja satisfeita então esse agente não tem, simultaneamente, como objectivo a sua negação. Os agentes têm objectivos consistentes entre si.

Esta é uma regra de inferência que é representada pela regra de programação em lógica:

$$holds\_at(\neg ach(A, \neg P), T) \leftarrow holds\_at(ach(A, P), T)$$

- Persistência

Ao longo do tempo, um agente continua a manter os seus objectivos, caso não exista nenhum evento que altere essa atitude. Esta propriedade é garantida pelos axiomas temporais do Cálculo de Eventos: as propriedades persistem no tempo, se não houver nenhum evento que as termine.

- Fecho

Esta regra significa que, se um agente acredita que uma propriedade virá a verificar-se e se acredita que essa propriedade implica uma outra, então ele acredita que a outra também se virá a verificar.

Conforme foi dito para as crenças, esta propriedade deverá dar origem a uma nova regra, para cada implicação lógica $Q \leftarrow P$ que se queira modelar:

$$holds\_at(ach(a, Q), T) \leftarrow holds\_at(ach(a, P), T). \tag{3.15}$$

Após a tradução para programação em lógica estendida do conjunto das regras apresentadas, obtém-se um sistema que permite a inferência e a interligação entre as diversas atitudes dos agentes. Essa interligação é efectuada assumindo um comportamento racional de acordo com regras básicas do comportamento humano.

As regras que definem o comportamento dos agentes com base nestas regras de racionalidade são apresentadas na sub-secção seguinte.

3.5 Regras de Comportamento

Além das regras de racionalidade, é fundamental definir regras de comportamento, de modo a modelar um agente/sistema computacional racional. De facto, a participação de cada agente em interacções (diálogos ou não) depende não só da sua racionalidade, mas também do tipo de comportamento que adopta. Do conjunto das propriedades que definem o comportamento, considerei fundamental a credulidade, a sinceridade, a cooperatividade e a actividade dos agentes. Existem outras propriedades, como por exemplo a agressividade, que são necessárias a uma completa modelação do comportamento de um agente, mas, no âmbito deste trabalho não serão abordadas. Neste sentido, irei somente analisar o processo de modelação das propriedades referidas.

Através da modelação destas propriedades podemos modelar qualquer combinação de comportamentos pró-activos ou reactivos, sinceros ou mentirosos, crédulos ou cépticos, e cooperativos ou não.

A incorporação, no modelo de um agente/sistema computacional, de regras especializadas que definem o seu comportamento, com a possibilidade de modelar comportamentos distintos através da alteração destas regras, é uma característica inovadora deste trabalho. De facto, ao separar o processo de raciocínio das regras que condicionam esse raciocínio, obteve-se um sistema bastante flexível e poderoso que permite modelar um conjunto bastante vasto de agentes. Os trabalhos existentes sobre diálogos não possuem esta característica e os trabalhos que definem agentes autónomos não lhes dão a capacidade de participar activamente em diálogos.

O comportamento do agente/sistema computacional é modelado através do recurso à representação de crenças sobre as características comportamentais dos diversos agentes (incluindo as crenças que o agente possui sobre as suas características). Por exemplo, as proposições

$$holds\_at(bel(a, sincero(b)), t).$$
$$holds\_at(bel(a, ingenuo(b)), t).$$
$$holds\_at(bel(a, cooperativo(b)), t).$$
$$holds\_at(bel(a, reactivo(b)), t).$$

significam que, no instante t, o agente a considera o agente b como sendo sincero, ingénuo, cooperativo e reactivo.

O agente/sistema computacional a possui crenças iniciais (definidas a partir do evento inicial e_0) sobre as características do seu comportamento e sobre as características do comportamento dos outros agentes:

$$initiates(e_0, t_0, bel(a, sincero(a))).$$
$$...$$
$$initiates(e_0, t_0, bel(a, sincero(X))) \leftarrow voce(X).$$
$$...$$
$$happens(e_0, t_0, t_0).$$
$$act(e_0, start).$$

A definição dos predicados que modelam o comportamento do agente será efectuada ao longo das sub-secções seguintes.

Com esta representação é possível modelar comportamentos que se alteram ao longo do tempo, de acordo com determinadas situações, como, por exemplo, a detecção de que um determinado agente não está a ser sincero pois transmite informação em que não acredita.

3.5.1 Credulidade

A credulidade representa o modo como um agente é *convencido* pela informação veiculada pelos outros agentes.

A credulidade pode ser descrita em relação às crenças e em relação aos actos de fala. Em relação às crenças representa o modo como é feita a transferência destas atitudes entre os agentes; em relação aos actos de fala condiciona os efeitos desses actos no receptor desses actos.

Transferência de crenças

Existem diversos graus de credulidade, desde o ingénuo que acredita em tudo o que crê que os outros acreditam, mesmo que tenha que rever as suas crenças anteriores, passando pelo crédulo que acredita só no que não entra em contradição com as suas próprias crenças, pelo racional que acredita se for plausível (isto é, se houver uma sequência hipotética de acções que permita verificar a informação veículada), e terminando no céptico que nunca acredita no que lhe é transmitido (o tradicional ver para crer).

A credulidade pode ser modelada através de um conjunto de regras (a incorporar no termo RC do tuplo que define o modelo do agente):

$$bel(H,P)\ if \qquad \begin{array}{l} bel(H,bel(S,P)), \\ transf(S,H,P). \end{array} \tag{3.16}$$

Esta regra define que, se um agente acredita que outro agente acredita numa proposição e se a transferência de informação se pode realizar, então o agente passa a acreditar na proposição.

A propriedade que define a possibilidade de transferência de informação é dependente do grau de credulidade do agente e é definida pelo conjunto seguinte de regras:

- Agente ingénuo:

$$transf(S,H,P)\ if \qquad \begin{array}{l} eu(H), \\ bel(H,ingenuo(H)). \end{array} \tag{3.17}$$

Esta regra significa que o agente H aceita transferir para si qualquer informação, independentemente das suas crenças.

Note-se que este processo de transferência de crenças poderá vir a introduzir contradições entre as suas crenças anteriores e as novas crenças. A remoção de contradições é efectuada através de um processo de revisão descrito no capítulo 4. Este processo permite modelar comportamentos conservadores (manter as crenças já existentes) ou não (adoptar as novas crenças).

Este tipo de agente representa uma simplificação do processo de transferência de crenças e só deverá ser utilizado em situações bastante específicas, em que os agentes participantes no diálogo são todos "bem comportados", i.e. são sinceros na informação que trasmitem.

- Um agente crédulo (mas não ingénuo) aceitará transferir somente a informação que não contrarie as suas próprias crenças:

$$transf(S,H,P)\ if \quad \begin{array}{l} eu(H), \\ bel(H, credulo(H)), \\ not\ bel(H,\neg P). \end{array} \tag{3.18}$$

De referir o recurso à negação por omissão para representar que se não for possível provar por omissão a crença em não P, então P pode ser acreditado.

- Um agente racional verifica se a informação transmitida P é plausível, isto é, se existe uma sequência de acções hipotéticas que permita atingir P:

$$transf(S,H,P)\ if \quad \begin{array}{l} eu(H), \\ bel(H, racional(H)), \\ plausivel(P). \end{array} \tag{3.19}$$

O predicado $plausivel(P)$ é definido através da criação de um modelo hipotético em que é adicionada ao modelo existente um nova restrição de integridade (t_∞ representa um instante de tempo futuro)

$$IC' = IC \cup \{\Leftarrow not\ holds\_at(P, t_\infty)\}$$

Como passo seguinte, é verificada a existência de um modelo hipotético que satisfaça as restrições existentes (abduzindo as sequências de acções necessárias, de um modo idêntico ao processo de planeamento abdutivo que será analisado em detalhe no capítulo 5). Caso exista um modelo hipotético a informação é considerada plausível, caso contrário, não.

- Um agente céptico nunca aceitará transferir a informação veículada pelos outros agentes:

$$\neg transf(S,H,P)\ if \quad \begin{array}{l} eu(H), \\ bel(H, ceptico(H)). \end{array} \tag{3.20}$$

Este agente revela um comportamento pouco interessante, dado não adquirir conhecimento que não lhe seja possível verificar com a sua própria experiência: não aprende com

os outros agentes, só com a sua experiência (com os factos que são válidos no seu modelo do mundo).

É necessário definir regras de integridade que relacionam os diferentes níveis de credulidade, dado que um agente só poderá ter um dado grau de credulidade, num dado instante:

$$\Leftarrow \quad holds\_at(bel(A, ingenuo(B)), T), holds\_at(bel(A, credulo(B)), T). \quad (3.21)$$

$$\Leftarrow \quad holds\_at(bel(A, ingenuo(B)), T), holds\_at(bel(A, ceptico(B)), T). \quad (3.22)$$

$$\Leftarrow \quad holds\_at(bel(A, ceptico(B)), T), holds\_at(bel(A, credulo(B)), T). \quad (3.23)$$

$$\ldots$$

Estas regras definem a impossibilidade de um agente acreditar que outro agente tem mais do que um grau de credulidade, num mesmo instante de tempo.

Efeitos dos actos de fala

A descrição dos actos de fala conforme foi efectuada no capítulo 2 assumiu algum grau de credulidade nos agentes.

Nomeadamente, as regras apresentadas assumem que o receptor dos diversos actos de fala passa a acreditar que o emissor acredita na informação que transmitiu. Em determinadas situações, esta pode ser uma propriedade muito forte e deverá ser condicionada a não existirem indícios de emissores não sinceros (mentirosos).

Para a modelação deste tipo de situações é necessário definir primeiro as condições para um agente ser considerado como mentiroso num determinado instante de tempo.

Uma regra possível de utilizar é:

$$\begin{aligned} inform(S, H, P) \qquad & causes\ bel(H, mentiroso(S)) \\ & if\ eu(H), \\ & \quad bel(H, bel(S, \neg P)). \end{aligned}$$

Ou seja, um agente S é mentiroso, do ponto de vista do agente H, se o informa sobre uma dada propriedade e se H acredita que S não acredita nela. Note-se que é possível existirem agentes que são mentirosos de uma forma selectiva, isto é, só o são para determinados agentes e em determinados instantes de tempo.

É possível definir a propriedade de ser mentiroso através de outras regras que entrem em conta com outras propriedades. Por exemplo, poderia ser descrita da seguinte forma, significando que se o receptor não acreditar em P, então S será um mentiroso (do seu ponto de vista):

$$\begin{aligned} inform(S, H, P) \qquad & causes\ bel(H, mentiroso(S)) \\ & if\ eu(H), \\ & \quad bel(H, \neg P). \end{aligned}$$

Esta regra é, no entanto, muito drástica e não contempla situações em que um agente é sincero mas está mal informado (o seu modelo do mundo não é correcto), pelo que será adoptada a primeira solução.

Dada a sua utilização na sequência deste capítulo, apresento as regras de programação em lógica resultado da transformação das regras apresentadas acima:

$$\begin{aligned} enabled(E, T_i) \leftarrow\ & act(E, inform(S, H, P)), \\ & eu(H), \\ & holds\_at(bel(H, bel(S, \neg P)), T_i). \end{aligned} \tag{3.24}$$

$$\begin{aligned} initiates(E, T_f, bel(B, mentiroso(A))) \leftarrow\ & happens(E, T_i, T_f), \\ & act(E, inform(S, H, P)), \\ & eu(H), \\ & holds\_at(bel(H, bel(S, \neg P)), T_i). \end{aligned} \tag{3.25}$$

Tendo em conta a propriedade $mentiroso/1$ que agora foi definida, será necessário alterar as definições dos actos de fala apresentadas no capítulo 2, do ponto de vista do receptor. No âmbito desta tese não é analisado o impacto, sobre o comportamento do emissor, da crença de que o receptor acredita que ele é mentiroso ($bel(S, bel(H, mentiroso(S)))$).

É, ainda, necessário definir uma restrição de integridade e duas regras adicionais sobre as crenças de um agente, num dado instante de tempo, sobre se outro agente é mentiroso ou verdadeiro:

$$\begin{aligned} \Leftarrow\ & holds\_at(bel(A, mentiroso(B)), T), \\ & holds\_at(bel(A, verdadeiro(B)), T). \end{aligned} \tag{3.26}$$

$$holds\_at(bel(A, \neg verdadeiro(B)), T) \leftarrow holds\_at(bel(A, mentiroso(B)), T). \tag{3.27}$$

$$holds\_at(bel(A, \neg mentiroso(B)), T) \leftarrow holds\_at(bel(A, verdadeiro(B)), T) \tag{3.28}$$

A definição dos actos de fala, do ponto de vista do receptor, é:

Definição 34 *Inform — Informação acerca de uma proposição*

Este acto de fala passa a ser representado por:

$$\begin{aligned} inform(S, H, P) \quad causes \quad & bel(H, bel(S, P)) \\ if \quad & eu(H), \\ & not\ bel(H, mentiroso(S)). \end{aligned} \tag{3.29}$$

O efeito do acto de fala passa a estar condicionado a uma existência de indícios de não sinceridade do agente emissor.

A tradução para regras de programação em lógica não é apresentada para nenhum dos actos de fala referidos nesta secção devido às suas semelhanças com as regras já apresentadas.

Definição 35 *Informref — Informação acerca de uma referência de uma proposição*

Este acto de fala passa a ser representado por:

$$\begin{array}{lll} informref(S,H,T,P) & causes & bel(H,bel(S,ref(T,P))) \\ & if & eu(H), \\ & & not\ bel(H,mentiroso(S)). \end{array} \tag{3.30}$$

Definição 36 *InformIf — Informação acerca do valor lógico de uma proposição*

Este acto de fala passa a ser representado por:

$$\begin{array}{lll} informif(S,H,P) & causes & bel(H,knowif(S,P)) \\ & if & eu(H), \\ & & not\ bel(H,mentiroso(S)). \end{array} \tag{3.31}$$

Definição 37 *Request — Pedido para efectuar uma dada acção*

A definição deste acto de fala passa a ser:

$$\begin{array}{lll} request(S,H,A) & causes & bel(H,int(S,A)), \\ & if & eu(H), \\ & & not\ bel(H,mentiroso(S)). \end{array} \tag{3.32}$$

Nesta acto de fala, é de salientar que o emissor não está a transmitir uma informação, mas um pedido de realização de acção. No entanto, e dado o carácter mentiroso do agente, é também necessário impedir que o receptor acredite que o emissor tem como intenção que a acção seja realizada.

Definição 38 *AskIf — Pedido sobre o valor lógico de uma proposição*

Este acto de fala passa a ser representado por:

$$\begin{aligned} askif(S,H,P) \quad & causes\ bel(H, ach(S, knowif(S,P))) \\ if \quad & eu(H), \\ & not\ bel(H, mentiroso(S)). \end{aligned} \tag{3.33}$$

A consideração efectuada para *request* também é válida neste e nos próximos actos de fala.

Definição 39 *CheckIf — Pedido sobre a confirmação do valor lógico de uma proposição*

Este acto de fala passa a ser representado por:

$$\begin{aligned} checkif(S,H,P) \quad causes \quad & bel(H, ach(S, knowif(S,H))) \\ if \quad & eu(H), \\ & not\ bel(H, mentiroso(S)). \end{aligned} \tag{3.34}$$

Definição 40 *Accept — Aceitar que uma determinada acção seja realizada*

Este acto de fala passa a ser representado por:

$$\begin{aligned} accept(S,A) \quad causes \quad & bel(H, int(S,A)) \\ if \quad & eu(H), \\ & not\ bel(H, mentiroso(S)). \end{aligned} \tag{3.35}$$

Definição 41 *Reject — Rejeitar que uma determinada acção seja realizada*

Este acto de fala passa a ser representado por:

$$\begin{aligned} reject(S,A) \quad causes \quad & bel(H, \neg int(S,A)) \\ if \quad & eu(H), \\ & not\ bel(H, mentiroso(S)). \end{aligned} \tag{3.36}$$

Com base nesta nova definição dos actos de fala e das regras de transferência de crenças a objectivos é, agora, possível modelar o comportamento de diversos graus de credulidade.

3.5.2 Sinceridade

Outra das características analisadas no âmbito deste trabalho relaciona-se com a sinceridade dos agentes. Isto é, os agentes intervenientes num diálogo podem ter diversos graus de sinceridade, desde os que só transmitem as informações em que acreditam, até aos que transmitem o contrário do que acreditam, passando pelos que transmitem informação sem terem qualquer crença a seu respeito. No âmbito deste trabalho iremos analisar os emissores verdadeiros e os mentirosos. Note-se que a noção de emissor verdadeiro ou mentiroso está sempre dependente do tempo e poderá sofrer alterações ao longo do tempo.

A sinceridade está directamente relacionada com os actos de fala dos agentes e a sua modelação tem consequências sobre a definição desses actos de fala (do ponto de vista do emissor desses actos).

Com base nas definições apresentadas na secção anterior, serão definidas descrições alternativas para cada tipo de sinceridade e acto de fala. Por uma questão de simplicidade, essa apresentação é efectuada organizada por tipo de sinceridade: agentes verdadeiros e agentes mentirosos.

Verdadeiros

Os agentes verdadeiros só transmitem a informação em que acreditam.

Neste caso, as regras apresentadas no capítulo 2 que descrevem os diversos actos de fala do ponto do vista do emissor estão de acordo com este princípio. No entanto, devem ser alteradas de modo a validar se o agente é ou não verdadeiro:

Definição 42 *Inform — Informação acerca de uma proposição*

$$\begin{array}{lll} inform(S,H,P) & causes & bel(S,bel(H,bel(S,P))) \\ & if & eu(S), \\ & & bel(S,verdadeiro(S)), \\ & & bel(S,P), \\ & & bel(S,int(S,inform(S,H,P))). \end{array} \tag{3.37}$$

Um emissor verdadeiro informa as proposições em que acredita.

Definição 43 *Informref — Informação acerca de uma referência de uma proposição*

$$\begin{array}{lll} informref(S,H,T,P) & causes & bel(S,bel(H,bel(S,ref(T,P)))) \\ & if & eu(S), \\ & & bel(S,verdadeiro(S)), \\ & & bel(S,ref(T,P)), \\ & & bel(S,int(S,informref(S,H,T,P)). \end{array} \tag{3.38}$$

O agente informa o receptor sobre uma referência T da propriedade P.

Definição 44 *InformIf — Informação acerca do valor lógico de uma proposição*

$$\begin{array}{lll} informif(S,H,P) & causes & bel(S,bel(H,knowif(S,P))) \\ & if & eu(S), \\ & & bel(S,verdadeiro(S)), \\ & & bel(S,knowif(S,P)), \\ & & bel(S,int(S,informif(S,H,P))). \end{array} \tag{3.39}$$

O agente informa o valor lógico da proposição P.

Definição 45 *Accept — Aceitar que uma determinada acção seja realizada*

$$\begin{array}{lll} accept(S,A) & causes & bel(S,bel(H,int(S,A))) \\ & if & eu(S), \\ & & bel(S,verdadeiro(S)), \\ & & bel(S,int(S,A)), \\ & & bel(S,int(S,accept(S,A))). \end{array} \tag{3.40}$$

O agente transmite a aceitação de que a acção A seja realizada.

Definição 46 *Reject — Rejeitar que uma determinada acção seja realziada*

$$\begin{array}{lll} reject(S,A) & causes & bel(S,bel(H,\neg int(S,A))) \\ & if & eu(S), \\ & & be(S,verdadeiro(S)), \\ & & bel(S,\neg int(S,A)), \\ & & bel(S,int(S,reject(S,A))). \end{array} \tag{3.41}$$

O agente transmite a rejeição de que a acção A seja realizada.

Mentirosos

Nesta situação os agentes transmitem informação contrária às suas crenças — mentiras.

Os diversos actos de fala têm de sofrer alterações, do ponto de vista do emissor, para estar de acordo com esta nova característica e passam a ser representados por:

Definição 47 *Inform — Informação acerca de uma proposição*

$$\begin{aligned} inform(S,H,\neg P)\ causes \quad & bel(S,bel(H,bel(S,\neg P))) \\ if \quad & eu(S), \\ & bel(S,mentiroso(S)), \\ & bel(S,P), \\ & bel(S,int(S,inform(S,H,\neg P))). \end{aligned} \tag{3.42}$$

Se o emissor mentiroso acredita numa proposição, então informa da sua negação (e vice-versa). Note-se que esta é uma simplificação da modelação de um mentiroso, pois não é de esperar que o mentiroso informe sempre sobre o contrário do que crê.

De notar, ainda, que um agente pode ser mentiroso num dado instante de tempo e, posteriormente, alterar o seu comportamento. Não são analisados neste trabalho os factores que poderão fazer alterar o comportamento de um agente: passar de verdadeiro a mentiroso (e vice-versa).

Definição 48 *Informref — Informação acerca de uma referência de uma proposição*

$$\begin{aligned} informref(S,H,U,P) \quad causes \quad & bel(S,bel(H,bel(S,ref(U,P)))) \\ if \quad & eu(S), \\ & bel(S,mentiroso(S)), \\ & bel(S,ref(T,P)), \\ & compl(T,U), \\ & bel(S,int(S,informref(S,H,U,P)). \end{aligned} \tag{3.43}$$

O agente informa o receptor sobre uma referência incorrecta para a propriedade P. O predicado $compl(T,U)$ obtém argumentos complementares (por exemplo, verdade — falso).

Definição 49 *InformIf — Informação acerca do valor lógico de uma proposição*

$$\begin{array}{lll} informif(S,H,\neg P) & causes & bel(S,bel(H,knowif(S,\neg P))) \\ & if & eu(S), \\ & & bel(S,mentiroso(S)), \\ & & bel(S,knowif(S,P)), \\ & & bel(S,int(S,informif(S,H,\neg P))). \end{array} \tag{3.44}$$

O agente informa o valor lógico contrário do que acredita.

Definição 50 *Accept — Aceitar efectuar uma determinada acção*

$$\begin{array}{lll} accept(S,A) & causes & bel(S,bel(H,int(S,A))) \\ & if & eu(S), \\ & & bel(S,mentiroso(S)), \\ & & bel(S,int(S,reject(S,A))). \end{array} \tag{3.45}$$

O agente mentiroso não tem a intenção de realizar uma dada acção; no entanto, efectua um acto de fala em que transmite a aceitação dessa acção.

Definição 51 *Reject — Rejeitar efectuar uma determinada acção*

$$\begin{array}{lll} reject(S,A) & causes & bel(S,bel(H,\neg int(S,A))) \\ & if & eu(S), \\ & & bel(S,mentiroso(S)), \\ & & bel(S,int(S,accept(S,A))). \end{array} \tag{3.46}$$

Nesta situação, o agente tenta convencer o outro agente de que não irá realizar uma acção que, de facto, pretende realizar.

3.5.3 Cooperatividade

A cooperatividade de um agente é uma relação que se estabelece entre os agentes e que define o modo como as intenções e os objectivos se transferem entre eles. Esta transferência é efectuada como consequência dos diversos actos de fala.

Transferência de intenções

Uma atitude cooperativa em relação às intenções pode ser descrita por uma regra que define que se um agente cooperativo acredita que outro agente tem como intenção que uma acção seja realizada, então ele passa a ter como intenção que ela seja realizada.

$$holds\_at(int(H, A), T) \leftarrow holds\_at(bel(H, int(S, A)), T), \quad holds\_at(bel(H, cooperativo(H)), T). \tag{3.47}$$

Pelo contrário, uma atitude não cooperativa em relação às intenções pode ter dois graus de não cooperação: activo ou passivo.

Uma não cooperação activa pode ser representada pela seguinte regra:

$$holds\_at(\neg int(H, A), T) \leftarrow holds\_at(bel(H, int(S, A)), T), \quad holds\_at(bel(H, nao\_cooperativo\_activo(H)), T). \tag{3.48}$$

Esta regra significa que o agente activamente não adopta (i.e. impede) a intenção de realizar as acções que crê que os outros agentes pretendem.

Numa não cooperação passiva os agentes simplesmente não adoptam nenhuma destas regras e não acrescentam nenhuma intenção impeditiva relativamente às acções que os outros agentes desejam efectuar.

É necessário definir restrições de integridade sobre os graus de cooperatividade:

$$\Leftarrow holds\_at(bel(A, cooperativo(B)), T), \quad holds\_at(bel(A, nao\_cooperativo\_activo(B)), T). \tag{3.49}$$

$$\Leftarrow holds\_at(bel(A, cooperativo(B)), T), \quad holds\_at(bel(A, nao\_cooperativo\_passivo(B)), T). \tag{3.50}$$

$$\Leftarrow holds\_at(bel(A, nao\_cooperativo\_passivo(B)), T), \quad holds\_at(bel(A, nao\_cooperativo\_activo(B)), T). \tag{3.51}$$

Estas restrições impedem que um agente acredite que outro agente (incluindo ele próprio) tenha mais do que um nível de cooperatividade, em cada instante de tempo.

Transferência de objectivos

A cooperação pode ser levada a um grau mais elevado e modelar a transferência de objectivos, isto é, dos estados que os agentes pretendem que se venham a verificar.

Para o agente cooperativo, teremos:

$$holds\_at(ach(H, P), T) \leftarrow holds\_at(bel(H, ach(S, P)), T), \quad holds\_at(bel(H, cooperativo(H)), T). \tag{3.52}$$

Estas regras significam que o agente adopta como seus os objectivos que acredita que o outro agente possui.

Conforme referido, uma atitude não cooperativa pode ter dois graus de não cooperação.

Uma não cooperação activa pode ser representada pela seguinte regra:

$$holds\_at(ach(H, \neg P), T) \leftarrow holds\_at(bel(H, ach(S, P)), T), \quad holds\_at(bel(H, nao\_cooperativo\_activo(H)), T). \tag{3.53}$$

Esta regra significa que o agente adopta os objectivos contrários ao que crê que os outros agentes possuem.

Numa não cooperação passiva o agente limita-se a não adoptar os objectivos alheios:

$$holds\_at(\neg ach(H, P), T) \leftarrow holds\_at(bel(H, ach(S, P)), T), \quad holds\_at(bel(H, nao\_cooperativo\_passivo(H)), T). \tag{3.54}$$

Esta regra significa que o agente não adopta os objectivos que crê que os outros agentes possuem.

A não cooperação (activa ou passiva) impede os agentes de possuir objectivos comuns e, em consequência, de agir de modo a satisfazer esses objectivos.

3.5.4 Actividade

A participação dos agentes nas diversas interacções pode ser pró-activa ou reactiva. Numa participação pró-activa pura, o agente/sistema computacional possui objectivos próprios que não foram transmitidos por outros agentes e realiza acções de modo a satisfazer esses objectivos. Numa participação reactiva pura, o agente/sistema computacional não possui objectivos próprios e (re)age somente como consequência dos eventos gerados pelos outros agentes interlocutores.

Em termos dos programas em lógica que modelam os agentes, a diferença é constituída pelos objectivos que o agente partilha com os seus interlocutores:

1. Um agente B é considerado pró-activo, do ponto de vista de um agente A sse:

$$holds\_at(bel(A, proactivo(B)), T) \leftarrow holds\_at(bel(A, ach(B, P)), T), \quad holds\_at(bel(A, \neg ach(A, P)), T). \tag{3.55}$$

   Isto é, se um agente A acredita que outro agente B tem um dado objectivo e se esse objectivo não é partilhado por A, então B é pró-activo (do ponto de vista de A).

2. Um agente A é pró-activo (do seu ponto de vista) sse:

$$hold\_at(bel(A, proactivo(A)), T) \leftarrow holds\_at(bel(A, ach(A, P)), T), \quad not\ (holds\_at(bel(A, ach(B, P)), T), A \neq B). \tag{3.56}$$

Um agente considera-se pró-activo se possui um dado objectivo que não acredita que o outro agente possui (utilizando a negação por omissão).

3. Um agente é reactivo se não for possível provar a sua pró-actividade (pela negação por omissão).

$$holds\_at(bel(A, reactivo(A)), T) \leftarrow not\ holds\_at(bel(A, proactivo(A)), T). \quad (3.57)$$

3.6 Sistema de diálogos multiagente

Nesta secção ir-se-á demonstrar que o sistema proposto para modelar agentes suporta interacções entre agentes com diferentes tipos de comportamento. Para tal, apresenta-se uma situação base e, através do recurso a modelos de comportamento típicos e ao ambiente de programação em lógica proposto, efectua-se a inferência das atitudes dos agentes.

Com o objectivo de não apresentar situações demasiado complexas e, porque o objectivo desta secção é demonstrar a modelação de comportamentos distintos, optou-se por modelar uma situação em que só existem dois agentes. No entanto, e devido à característica de que um agente é uma entidade autónoma, é possível estender o processo a situações com um maior número de agentes. Do ponto de vista de cada um desses agentes, só se alteraria o conjunto de crenças que possuem uns em relação aos outros — os modelos dos outros agentes.

3.6.1 Descrição da situação

A situação é descrita pela existência de dois agentes, o Astérix e o Obélix, e pelo cão Ideiafix. Existem, ainda, dois quartos e uma cadeira num deles.

Numa situação inicial, o Astérix está no quarto que tem a cadeira e coloca o cão na cadeira. Depois, entra no outro quarto, encontra o Obélix e diz-lhe:

- O Ideiafix está na cadeira.

Numa primeira variação a esta situação vamos assumir que, anterior ao acto de fala do Astérix, o Obélix viu o Ideiafix saltar para o chão e viu que o Astérix não observou este evento. Numa segunda fase vamos assumir que o Obélix viu que o Astérix também viu o Ideiafix a saltar para o chão.

Esta situação é uma adaptação de um cenário referido por Perrault em [Per90]. Perrault, no seu trabalho, analisa uma situação idêntica e propõe quais as repercussões em termos de atitudes que esta situação origina. Nesta secção, e utilizando a metodologia de modelação de agentes descrita neste capítulo, obtenho resultados semelhantes. De notar que, ao contrário desta proposta, Perrault não propõe nenhum método procedimental para obter as atitudes válidas em cada instante de tempo.

De modo a modelar esta situação inicial, é necessário representar as acções específicas deste domínio (foram efectuadas bastantes simplificações de modo a não complexificar desnecessariamente a situação):

1. $colocar\_sobre(X, Y, Z)$ $causes$ $sobre(Y, Z)$

   que significa que a acção de um agente X colocar a entidade Y sobre Z, tem como efeito que Y fica sobre Z no modelo do agente que realizou a acção.

2. $saltar\_para(X, Y)$ $causes$ $sobre(X, Y)$

   que significa que o acto de X saltar para o local Y provoca o efeito de ficar sobre ele.

3. $ver(X, A)$ $causes$ $bel(X, P)$,

   para todo o P que seja uma consequência de A (A $causes$ P ...)

   Esta regra significa que o acto de ver uma acção provoca a crença nas consequências dessa acção. Esta é uma meta-regra que deverá ser criada para todas as acções e, nomeadamente, para a acção de $saltar\_para(x, y)$.

4. $ver(X, ver(Y, A))$ $causes$ $bel(X, bel(Y, P))$,

   para todo o P que seja uma consequência de A (A $causes$ P ...)

   Esta regra significa que o acto de ver um agente a ver uma acção provoca a crença de que o outro agente acredita nas consequências dessa acção. Esta é novamente uma meta-regra que deverá ser criada para todas as acções.

No âmbito deste exemplo as acções serão simplificadas e assumir-se-ão instantâneas e sem pré-condições. Deste modo, elimina-se o problema de verificar se a acção já se tinha completado na totalidade quando foi observada.

Estas regras são traduzidas pelos seguintes programas em lógica:

$$enabled(E, T_i) \leftarrow act(E, colocar\_sobre(X, Y, Z)). \quad (3.58)$$

$$initiates(E, T_f, sobre(X, Y)) \leftarrow happens(E, T_i, T_f), \quad (3.59)$$
$$act(E, colocar\_sobre(X, Y)).$$

$$enabled(E, T_i) \leftarrow act(E, saltar\_para(X, Y)). \quad (3.60)$$

$$initiates(E, T_f, sobre(X, Y)) \leftarrow happens(E, T_i, T_f), \quad (3.61)$$
$$act(E, saltar\_para(X, Y)).$$

$$enabled(E, T_i) \leftarrow act(E, ver(X, A)). \quad (3.62)$$

$$initiates(E, T_f, bel(X, sobre(Y, Z))) \leftarrow act(E, ver(X, \quad (3.63)$$
$$saltar\_para(Y, Z))),$$
$$happens(E, T_i, T_f).$$

$$\begin{aligned} initiates(E, T_f, bel(X, & \\ bel(Y, sobre(Z, W)))) \leftarrow & \ act(E, ver(X, \\ & \quad ver(Y, saltar\_para(Z, W)))), \\ & \ happens(E, T_i, T_f). \end{aligned} \tag{3.64}$$

De referir que para simplificar a descrição das acções, não foram apresentadas as regras dos efeitos da acção *ver* sobre a acção *colocar_sobre*:

$$ver(X, colocar\_sobre(Y, Z, W)).$$

Com base nestas regras do domínio é possível modelar a situação descrita. É necessário, previamente, definir o modelo de cada agente, de modo a simular a inferência das suas atitudes.

3.6.2 Cenário 1 — Agentes *bem comportados*

Num primeiro cenário suponhamos que ambos os agentes (o Astérix e o Obélix) são agentes bem comportados, isto é, são sinceros e ingénuos em relação à informação transmitida pelo outro.

O modelo inicial de cada agente (L_1 e L_2) é semelhante:

$$L_1 = L_2 = < At, RC, Ac, T, Rr, CM >$$

onde não existem atitudes iniciais, $At = \{\}$; RC é constituído por regras correspondentes a agentes sinceros e crédulos (apresentadas em secções anteriores); Ac é constituído pela descrição das acções do domínio apresentadas anteriormente e pelos actos de fala correspondentes a agentes sinceros e crédulos (apresentados no capítulo anterior); T são os axiomas temporais apresentados no capítulo anterior; Rr são as regras de racionalidade apresentadas. O conhecimento do mundo é dado por:

$$\begin{aligned} CM = \{ & cão(ideiafix), \\ & humano(astérix), \\ & humano(obélix), \\ & ent(cadeira), \\ & ent(chão)\}. \end{aligned}$$

Do ponto de vista do Astérix, o evento inicial e_1 pode ser representado por:

$$happens(e_1, 1, 1). \tag{3.65}$$

$$act(e_1, colocar\_sobre(astérix, ideiafix, cadeira)) \tag{3.66}$$

e tem como consequência que, no seu modelo, o cão está sobre a cadeira (a partir dos axiomas temporais e da regra 3.59):

$$holds\_at(sobre(ideiafix, cadeira), 1). \tag{3.67}$$

e que o Astérix acredita nisso (a partir da regra em programação em lógica obtida a partir da regra de necessidade das crenças 3.3):

$$holds\_at(bel(astérix, sobre(ideiafix, cadeira)), 1). \tag{3.68}$$

O segundo evento e_2 pode ser representado por:

$$happens(e_2, 3, 3). \tag{3.69}$$

$$act(e_2, inform(astérix, obélix, sobre(ideiafix, cadeira))). \tag{3.70}$$

Este evento já pertence também ao modelo do Obélix e tem como consequência que ele passa a acreditar que o Astérix acredita que o Ideiafix está sobre a cadeira (a partir das regras de $inform$ 3.29):

$$holds\_at(bel(obélix, bel(astérix, sobre(ideiafix, cadeira))), 3). \tag{3.71}$$

Como estamos a assumir um cenário de agentes crédulos, o modelo do Obélix suporta também a inferência de que acredita no facto comunicado (a partir das regras 3.16 e 3.17):

$$holds\_at(bel(obélix, sobre(ideiafix, cadeira)), 3).$$

Ou seja, ambos os agentes acreditam que o Ideiafix está efectivamente sobre a cadeira.

Se, conforme descrito nas variações apresentadas, o Obélix viu o Ideiafix saltar para o chão:

$$happens(e_3, 2, 2). \tag{3.72}$$

$$act(e_3, ver(obélix, saltar(ideiafix, chão))). \tag{3.73}$$

então o seu modelo suporta o facto de que:

$$holds\_at(bel(obélix, sobre(ideiafix, chão)), 2). \tag{3.74}$$

utilizando as regras que descrevem os efeitos da acção de ver (3.63).

Nesta situação, e como o Obélix é crédulo, o acto de fala posterior fará com que sejam revistas as suas crenças e continuará a ser válido o facto de que:

$$holds\_at(bel(obélix, sobre(ideiafix, cadeira)), 3).$$

porque o Obélix sobrepõe a informação que lhe é transmitida às suas próprias crenças (eventualmente, o Ideiafix poderá ter saltado novamente para a cadeira). Existe a necessidade de actualizar as crenças, de modo a evitar possíveis contradições. Neste caso a crença de que o Ideiafix estaria no chão deveria ser terminada após o acto de fala de Astérix (informando que o Ideiafix está na cadeira).

O processo de actualização de crenças e supressão de contradições será analisado em detalhe no capítulo 4.

3.6.3 Cenário 2 — Agentes sinceros, mas não ingénuos

Neste cenário, o parâmetro RC dos tuplos que descrevem o modelos dos agentes, deve ser substituído por um outro que defina a credulidade dos agentes, e não a sua ingenuidade (conforme apresentado previamente neste capítulo).

As inferências existentes são em tudo semelhantes ao cenário 1, excepto na inferência das consequências do acto $inform$. Neste caso, o Obélix, ao confrontar as suas crenças com a informação transmitida, não irá alterar aquilo em que acredita. Teremos, assim, suporte para os seguintes factos no modelo do Obélix (utilizando as regras 3.16 e 3.18):

$$holds\_at(bel(obélix, sobre(ideiafix, chão)), 3). \tag{3.75}$$

$$holds\_at(bel(obélix, bel(astérix, sobre(ideiafix, cadeira))), 3). \tag{3.76}$$

Representado as suas crenças de que o Ideiafix está no chão, mas que o Astérix acredita que ele está na cadeira. Esta divergência entre os modelos pode provocar um comportamento de não aceitação da informação, justificando essa não aceitação com o facto de o Obélix ter visto o Ideiafix saltar para o chão.

3.6.4 Cenário 3 — Um agente mentiroso e um crédulo

Neste cenário, o Astérix é mentiroso (segunda variação da situação proposta) e o Obélix é crédulo. Os modelos dos agentes são distintos no início da situação: o modelo do Obélix é semelhante ao do cenário 2, mas o modelo do Astérix contém as regras que definem a não sinceridade dos agentes.

Neste cenário existe um evento adicional no modelo do Astérix, dado que ele viu o Ideiafix saltar para o chão:

$$happens(e_4, 2, 2). \tag{3.77}$$

$$act(e_4, ver(astérix, saltar(ideiafix, chão))). \tag{3.78}$$

e então o seu modelo suporta o facto de que:

$$holds\_at(bel(astérix, sobre(ideiafix, chão)), 2). \tag{3.79}$$

utilizando as regras que descrevem os efeitos da acção de ver (3.63).

O acto e_2 $inform$ continua a ser possível, dado que as suas pré-condições são válidas (a crença no contrário do que se pretende informar — regra 3.42).

Do ponto de vista do Obélix, e na ausência de mais informação, ele passa a acreditar que o Astérix acredita que o Ideiafix está sobre a cadeira (3.29):

$$holds\_at(bel(obélix, bel(astérix, sobre(ideiafix, cadeira))), 3). \tag{3.80}$$

No entanto, devido ao evento e_3 (ele viu o Ideiafix saltar para o chão), ele não tem em consideração a informação transmitida — o Obélix não é ingénuo (3.18):

$$holds\_at(bel(obélix, sobre(ideiafix, chão)), 3). \tag{3.81}$$

Na segunda variação a esta situação, existe um evento adicional no modelo do Obélix, dado que ele viu que o Astérix viu o Ideiafix saltar para o chão:

$$happens(e_5, 2, 2). \tag{3.82}$$

$$act(e_5, ver(obélix, ver(astérix, saltar(ideiafix, chão)))). \tag{3.83}$$

então, como consequência desta acção, o seu modelo suporta o facto de que (a partir da regra 3.64):

$$holds\_at(bel(obélix, bel(astérix, sobre(ideiafix, chão))), 2). \tag{3.84}$$

Ou seja, o Obélix acredita que o Astérix acredita que o Ideiafix está no chão.

No entanto, após o acto de informar e_2, o Obélix deduz que o Astérix é mentiroso e não considera a informação veiculada.

Pelo contrário, cataloga o Astérix como um agente mentiroso (regra 3.25):

$$holds\_at(bel(obélix, mentiroso(astérix)), 3). \tag{3.85}$$

Conforme foi analisado nestes diferentes cenários, o modelo proposto suporta o tipo de inferências que se julgaria normal, a partir dos diversos comportamentos observados.

No capítulo 6 será apresentado um exemplo de um ambiente multiagente, em que é explorada em maior profundidade a capacidade de adaptação do comportamento dos agentes, em face a diversas situações.

3.7 Conclusões

Neste capítulo apresentou-se uma proposta para modelação de agentes racionais que participem em interacções com outros agentes. Estas interacções podem ser através de actos de fala (diálogos) ou por recurso a outras acções (visualização, por exemplo).

Nesta proposta, os agentes são descritos através de um tuplo de informação que contém o que poderá designar-se pelo estado mental do agente. De facto, contém a informação sobre as suas atitudes, o seu conhecimento do mundo exterior e das acções que pode praticar, bem como regras que definem a sua racionalidade e o seu comportamento.

Foram apresentadas as regras que possibilitam a modelação da racionalidade e de diversos tipos de comportamento. As propriedades do comportamento analisadas foram a cooperatividade, a sinceridade, a credulidade e a actividade. Apresentaram-se, ainda, as regras que permitem modelar a conjugação destas quatro propriedades. Note-se que não se pretende com este trabalho analisar em detalhe o processo de modelação de comportamento dos agentes em todas as suas componentes; pretendeu-se somente demonstrar a adequabilidade do sistema proposto para a representação dessas componentes.

A partir do estado mental do agente, e recorrendo ao processo de inferência apresentado no capítulo anterior, obtém-se um agente com capacidade de inferir atitudes e de alterá-las ao longo do tempo.

O processo de modelação foi ilustrado recorrendo a um exemplo em que agentes com diferentes tipos de comportamento interagem entre si. Através deste exemplo, demonstrou-se que o sistema proposto permite modelar diversos tipos de situações e que, alterando as regras que definem o comportamento dos agentes, o sistema suporta interacções distintas.

Em suma, o processo proposto permite simular e formalizar sob um ambiente de programação em lógica com semântica bem fundada diversos tipos de situações e obter ambientes multiagentes de diferentes características.

As inferências efectuadas recorreram somente a métodos dedutivos sobre os eventos observados. No capítulo seguinte será proposto um sistema que recorre à abdução de modo a inferir acções e estados não conhecidos inicialmente — o problema do conhecimento incompleto nos diálogos.

Capítulo 4

Inferência de atitudes

Neste capítulo apresento o modo como o sistema proposto consegue realizar a inferência das atitudes (crenças, objectivos, intenções) do agente/sistema computacional. Para tal, é descrito o processo de actualização das atitudes, com base nos eventos ocorridos (secção 4.2). Este processo poderá introduzir inconsistências no modelo do agente, que deverão ser eliminadas (4.3).

A actualização e a revisão do estado mental do agente serve de suporte ao processo de inferência de atitudes e de reconhecimento dos planos dos agentes interlocutores no diálogo (4.4). O encadeamento destes processos é demonstrado, sendo realçada a capacidade de lidar com diversos tipos de diálogos, em que surgem com frequência situações de erro e de ambiguidade (secção 4.5).

A solução proposta permite resolver problemas detectados em trabalho anterior de outros autores, conforme é demonstrado através dos exemplos apresentados no domínio de mensagens de correio electrónico ([Pol86]).

Finalmente, na última secção são sumarizadas as principais contribuições deste capítulo, o qual, ao permitir a inferência de atitudes e de planos, serve de base a um processo de planeamento de acções necessário para uma participação *inteligente* em diálogos.

4.1 Introdução

Conforme foi apresentado nos capítulos anteriores, qualquer agente, para poder participar de um modo *inteligente* em diálogos, necessita de representar o conhecimento e de modelar os outros agentes participantes nesse diálogo de uma forma o mais correcta possível. É com base neste conhecimento, e tendo em conta os diversos actos de fala do diálogo, que o agente efectua um conjunto de inferências que lhe permitem rever as suas atitudes em relação a si próprio, em relação ao meio que o rodeia e, em especial, em relação aos agentes que interagem com ele.

Como exemplo, vejamos um pequeno diálogo trocado em mensagens de correio electrónico (adaptado de [Pol86]):

Ana: Quero falar com a Catarina. Qual o número de telefone do hospital?

Susana: Ela já está em casa. O número de casa dela é o 999999.

Neste exemplo, a intervenção do primeiro interlocutor deverá dar origem aos seguintes actos de fala (por uma questão de simplificação, considerei que os actos de fala ocorrem num dado instante de tempo):

$$happens(e_1, 1, 1). \tag{4.1}$$

$$act(e_1, inform(ana, susana, int(ana, falar(ana, catarina)))). \tag{4.2}$$

$$happens(e_2, 1, 1). \tag{4.3}$$

$$act(e_2, request(ana, susana, \tag{4.4}$$
$$\quad informref(susana, ana, N, telefone(N, hospital)))).$$

Estes actos de fala são obtidos a partir de uma análise às frases em Língua Natural.

De acordo com o exemplo apresentado, é esperado que, com base no seu *estado mental*, a Susana tenha a capacidade para inferir que o *request* efectuado era apenas um passo no plano mais global explícito da Ana de falar com a Catarina. Esta interpretação é possível porque telefonar a alguém é uma acção que permite falar com essa pessoa e pedir um número de telefone pode ser visto como um passo no sentido de telefonar. A Susana deverá ainda, caso seja cooperativa, tentar ajudar a Ana a satisfazer as suas intenções — falar com a Catarina.

Assim, é necessário suportar a inferência das seguintes crenças (supondo que a Susana acredita que a Catarina já teve alta do hospital e já está em casa):

$$bel(susana, int(ana, falar(ana, catarina))). \tag{4.5}$$

$$bel(susana, int(ana, telefonar(ana, catarina, hospital))). \tag{4.6}$$

$$bel(susana, bel(ana, em(hospital, catarina))). \tag{4.7}$$

$$bel(susana, em(casa(catarina), catarina)). \tag{4.8}$$

$$int(susana, falar(ana, catarina)). \tag{4.9}$$

$$int(susana, telefonar(ana, catarina, casa(catarina))). \tag{4.10}$$

Ou seja, a Susana acredita que a Ana tem a intenção de falar com a Catarina, tem a intenção de telefonar para o hospital para falar com ela e acredita que a Catarina está no hospital. Por outro lado, a Susana acredita que a Catarina está em casa, assume a intenção de que a Ana fale com a Catarina (é cooperativa) e a intenção de que a Ana telefone para casa da Catarina.

A Susana, como agente inteligente e autónomo que é, deveria, ainda, ter a capacidade de efectuar a geração dos seguintes actos de fala:

$$inform(susana, ana, bel(susana, em(casa(catarina), catarina))). \quad (4.11)$$

$$informref(susana, ana, 999999, telefone(999999, casa(catarina))). \quad (4.12)$$

Ou seja, a Susana informa a Ana de que acredita que a Catarina já está em casa e informa-a ainda do número de telefone de casa da Catarina.

O processo de geração dos actos de fala será abordado em detalhe no capítulo 5.

Este exemplo é elucidativo de um tipo de problemas com os quais é necessário lidar em diálogos — planos incorrectos (i.e. não adequados aos dados efectivos) e informação incompleta — e que será elaborado neste capítulo. No caso deste diálogo, existe um plano incorrecto inferível a partir da informação transmitida pela Ana. Existe, ainda, uma relação de causalidade que não está explicitada no diálogo e que é necessário inferir (a relação entre *telefonar* visando *falar* com alguém).

Estes problemas têm vindo a ser objecto de análise nos últimos anos, tendo sido propostas algumas soluções parciais. Nomeadamente, Pollack ([Pol86]) no seu trabalho suporta situações em que existem planos incorrectos e Litman ([LA87]) permite a resolução de situações com actos de fala indirectos (por exemplo, a frase: "Está demasiado calor!", como um pedido indirecto para que se abra uma janela). No entanto, e conforme foi referido no capítulo 1, estas abordagens não possuem um sistema formal de suporte com uma semântica bem definida, que permita o cálculo das propriedades válidas em cada instante de tempo, bem como a sua actualização e revisão.

A solução proposta neste capítulo permite obter um ambiente formal de programação em lógica com semântica bem fundada que resolve um leque bastante vasto dos problemas enfrentados por aquelas abordagens. De facto, o processo proposto para a inferência de atitudes e de reconhecimento de planos permite suportar situações em que existem planos incorrectos e omissões, sem haver a necessidade de efectuar alterações ao sistema proposto. Isto é, a solução proposta é suficientemente abrangente e flexível para suportar um leque vasto de situações em diálogos.

Na próxima secção será apresentado o processo de actualização dos estados mentais do agente/sistema computacional, sendo descrito na secção 4.3 o processo de revisão do estado mental que permite eliminar possíveis contradições. Na secção 4.4 é descrito o processo de inferência de atitudes e o processo de reconhecimento de planos. Nas secções 4.4.1 a 4.4.3 serão apresentados exemplos de aplicação da teoria proposta.

4.2 Actualização de estados mentais

A actualização do estado mental do agente/sistema computacional é uma componente fundamental no processo de participação em diálogos. De facto, tendo como base a representação do estado mental do agente e uma acção que foi executada (por si ou por outro agente), é necessário desencadear um processo de raciocínio que dê origem a uma actualização das atitudes do próprio agente, visando criar um novo *estado mental*. O processo de actualização de atitudes pode levar a que, em cada instante, sejam iniciadas novas atitudes ou sejam terminadas atitudes anteriores.

O processo de actualização de atitudes, com base num conjunto de eventos que ocorreram num dado intervalo de tempo, pode ser descrito por uma função *actualiza*, tal que:

Definição 52 *Seja M o conjunto de todos os modelos possíveis de um agente; seja A o conjunto de todas as acções passíveis de serem executadas; seja AT o conjunto de todas as atitudes; seja E o conjunto de todos os eventos. Sejam $a_1, ..., a_n$ acções e sejam $e_1, ..., e_n$ eventos tais que $a_i \in A$, $e_i \in E, 1 \leq i \leq n$, e se verifica que $act(e_i, a_i)$ e $happens(e_i, t_i, t'_i), 1 \leq i \leq n$.*

Seja m um modelo de um agente, tal que: $m = < At, RC, Ac, T, Rr, CM > \in M$, onde At são as atitudes do agente, RC são as regras de conhecimento, Ac as regras que definem as acções, T os axiomas temporais, Rr as regras de racionalidade e CM as regras que representam o conhecimento do mundo.

A actualização do modelo do agente é uma função actualiza : $M \times E^n \longrightarrow M$, tal que:

1. *$actualiza(m, e_1 \times ... \times e_n) = < At_1, RC_1, Ac_1, T_1, Rr_1, CM_1 >$*
2. *$RC_1 = RC$, as regras de comportamento mantêm-se;*
3. *$Ac_1 = Ac$, a descrição de acções mantém-se;*
4. *$T1 = T$, os axiomas temporais mantêm-se;*
5. *$Rr_1 = Rr$, os axiomas de racionalidade mantêm-se;*
6. *$CM_1 = CM \cup \{act(e_1, a_1), happens(e_1, t_1, t'_1), ..., act(e_n, a_n), happens(e_n, t_n, t'_n)\}$, a descrição do mundo é actualizada com os novos eventos $e_1, ..., e_n$ (descritos pelas acções e pelo intervalo de tempo durante o qual essas acções decorreram);*
7. *At_1 são as novas atitudes suportadas pelo modelo do agente, i.e., $at \in At_1$ sse $at \in AT$ e*

   $at \in PCFXSM(At \cup RC \cup Ac \cup T \cup Rr \cup CM_1)$, isto é at pertence ao modelo revisto preferido do programa em lógica que define o estado mental do agente acrescido com os novos eventos. Recorde-se, que de acordo com o apresentado no capítulo 2,

PCRXM representa o modelo preferido da revisão a 2 valores, conforme proposto no sistema REVISE.

Esta definição significa que, no processo de actualização de atitudes de um agente, somente são alteradas as suas atitudes e o conhecimento do mundo. As novas atitudes são as atitudes suportadas pelo modelo revisto do programa em lógica estendida obtido pela conjunção das regras de programação em lógica que definiam o modelo anterior com as regras que definem os novos eventos. O conhecimento do mundo é aumentado com a descrição dos novos eventos. O processo de revisão será descrito na secção 4.3, sendo o processo de selecção da solução preferida abordado no capítulo 5.

De notar que esta definição contempla a possibilidade de existência de eventos simultâneos $(e_1, ..., e_n)$.

Dado que as atitudes de cada agente estão sempre associadas aos instantes temporais em que são válidas, o processo de actualização traduz-se num processo em que não é alterada a maior parte das atitudes anteriores ao evento que causou a actualização. São acrescentadas novas atitudes — atitudes válidas a partir do instante de tempo final do evento — e são terminadas atitudes já existentes que deixaram de ser suportadas pelo modelo do agente.

Note-se que o processo de actualização de atitudes pode iniciar atitudes que sejam inconsistentes com atitudes anteriores. Nesta situação é necessário efectuar uma revisão do modelo, terminando as atitudes que contribuem para a inconsistência. Este processo de revisão é efectuado recorrendo a um processo de remoção de contradições (secção 4.3) e seleccionando a solução preferida (capítulo 5).

Utilizando este processo de actualização de atitudes é possível suportar a crença de um agente numa dada proposição num dado intervalo de tempo e, ao mesmo tempo, suportar uma crença contrária noutro intervalo de tempo.

Como exemplo, suponhamos um agente que, num instante de tempo t_1, acredita que a Catarina está no hospital:

$$holds\_at(bel(a, em(hospital, catarina)), t_1).$$

Num instante $t_2 > t_1$, outro agente informa-o que afinal a Catarina está em casa:

$$happens(e_1, t_2, t_2).$$
$$act(e_1, inform(b, a, em(casa, catarina))).$$

Supondo que o modelo de a é descrito por regras de programação em lógica que definem um comportamento ingénuo (acredita em tudo o que lhe dizem), o novo modelo de a suportaria as seguintes atitudes:

$$holds\_at(bel(a, em(hospital, catarina)), t_1).$$
$$holds\_at(bel(a, em(casa, catarina)), t_2).$$

Para tal, utilizou-se a definição de *inform* apresentada anteriormente e a função de actualização de atitudes descrita nesta secção.

No entanto, assumindo que existe uma restrição de integridade que impede um agente de acreditar que um outro agente pode estar em dois locais no mesmo instante de tempo, é necessário rever o estado mental do agente terminando uma das duas crenças:

1.

$$holds\_at(bel(a, em(hospital, catarina)), t_2).$$

2.

$$holds\_at(bel(a, em(casa, catarina)), t_2).$$

A revisão do estado mental e selecção da solução preferida permite adoptar qualquer das opções.

Resumindo, o processo de actualização de atitudes tem como base a semântica bem fundada dos programas em lógica (com eventual remoção de contradições) que definem o modelo de cada agente e os diversos actos que vão sendo executados.

Nas sub-secções seguintes demonstrar-se-á que o processo de actualização e revisão dos estados mentais permite modelar, não só raciocínios monótonos dedutivos, mas também raciocínios não-monótonos abdutivos, fundamentais a um ambiente em que existe informação inicial incompleta. Esta característica é fundamental para a participação dos agentes em diálogos, dado o carácter não-monótono da evolução do conhecimento durante os diálogos.

4.2.1 Inferência dedutiva

O processo básico de inferência em diálogos é baseado no raciocínio dedutivo. Neste tipo de raciocínio, a partir de $\{b \leftarrow a, a\}$ é possível inferir b.

No ambiente de programação em lógica estendida proposto, o raciocínio dedutivo é garantido pela semântica bem fundada para programas em lógica estendida, apresentada no capítulo 2.

Este tipo de raciocínio permite que, dado um acto de fala, o modelo bem fundado obtido para a actualização do modelo dos agentes, contenha os efeitos desses actos de fala.

Como exemplo, suponhamos o seguinte acto de fala *inform,* ocorrido no instante de tempo 1 entre o agente a — Susana — e o agente b — Ana:

Susana: A Catarina está em casa.

$$happens(e_1, 1, 1). \tag{4.13}$$

$$act(e_1, inform(a, b, em(casa, catarina))). \tag{4.14}$$

Considerando o operador *inform* definido para um agente receptor (ver capítulo 2 e 3):

$$\begin{aligned} inform(S,H,P) \quad & causes\ bel(H,bel(S,P)), \\ & if\ eu(H), \\ & \quad not\ bel(H,mentiroso(S)). \end{aligned}$$

com as correspondentes regras de programação em lógica para o receptor H:

$$\begin{aligned} enabled(E,T_i) \leftarrow\ & act(E,inform(S,H,P)), \\ & eu(H), \\ & not\ holds\_at(bel(H,mentiroso(S)),T_i). \end{aligned} \tag{4.15}$$

$$\begin{aligned} initiates(E,T_f,bel(H,bel(S,P))) \leftarrow\ & happens(E,T_i,T_f), \\ & act(E,inform(S,H,P)), \\ & eu(H), \\ & not\ holds\_at(bel(H,mentiroso(S)),T_i). \end{aligned} \tag{4.16}$$

Seja M o programa em lógica correspondente ao modelo do utilizador b antes do acto de fala. Tendo em conta o acto de fala descrito (regras 4.13 e 4.14), a descrição da acção *inform* (regras 4.15 e 4.16) e o modelo do receptor b (incluindo os seus axiomas temporais) é válida a seguinte proposição:

$$holds\_at(bel(b,bel(a,em(casa,catarina))),1) \in PCFXSM(actualiza(M,e_1))$$

Ou seja, o efeito da acção *inform* é suportado pelo modelo preferido revisto do programa em lógica estendida que representa o modelo do agente b.

Deste modo é assegurado que os efeitos das acções praticadas são suportadas pelos modelos actualizados de cada agente.

4.2.2 Raciocínio abdutivo

Conforme analisado no capítulo 1, um processo de participação em diálogos necessita de possuir capacidade para efectuar raciocínios não-monótonos abdutivos. De facto, a existência de informação incompleta acerca do mundo que rodeia os agentes, cria a necessidade de efectuar inferências sobre eventos que não são conhecidos *a priori*, e que poderão ter influenciado eventos posteriores.

No capítulo 2 definiu-se o raciocínio abdutivo e o processo de o modelar utilizando a programação em lógica estendida com a negação explícita e a semântica bem fundada.

Em termos gerais, dado um literal abdutível a, uma regra $b \leftarrow a$, e uma observação b, o modelo bem fundado do programa em lógica estendida que modela o raciocínio abdutivo

permite obter o literal a como uma possível explicação para a observação b (quando acrescentada como restrição de integridade $b \Leftarrow$).

Como exemplo de aplicação de raciocínio abdutivo, vejamos a seguinte frase entre dois agentes:

a: Queria ficar magro!

que dá origem aos seguintes factos que transmitem o objectivo de a de ser magro:

$$happens(e_1, 1, 1). \tag{4.17}$$
$$act(e_1, inform(a, b, ach(a, magro(a)))). \tag{4.18}$$

Suponhamos que o modelo dos agentes contém a descrição das consequências da acção *fazer_dieta* (versão bastante simplificada que elimina o factor temporal e as pré-condições da acção de fazer uma dieta):

$$fazer\_dieta(X)\ causes\ magro(X)$$

e a respectiva descrição em regras de programação em lógica:

$$enabled(E, T_i) \leftarrow act(E, fazer\_dieta(X)). \tag{4.19}$$
$$initiates(E, T_f, magro(X)) \leftarrow act(E, fazer\_dieta(X)), \tag{4.20}$$
$$happens(E, T_i, T_f).$$

A partir do acto de fala *inform* descrito inicialmente (regras 4.17 e 4.18), das regras do operador *inform* (4.15 e 4.16) e da transferência de crenças é suportada a seguinte atitude:

$$holds\_at(bel(b, ach(a, magro(a))), 1).$$

Esta atitude representa a crença de b em que o agente a tem como objectivo ficar magro.

Assumindo um modelo de agente cooperativo, temos que:

$$holds\_at(ach(b, magro(a)), 1).$$

Ou seja, o agente b adopta o objectivo do agente a.

O planeamento das acções pode ser efectuado utilizando os axiomas temporais descritos no capítulo 2 que estão contidos no modelo de cada agente. Neste sentido, o agente acrescenta ao seu modelo, como restrições de integridade, os objectivos que deseja que se venham a verificar no futuro (este processo será analisado em datalhe no capítulo 5):

$$holds\_at(magro(a), t) \Leftarrow$$
$$t \geq 1 \Leftarrow$$

O modelo bem fundado do programa lógica que modela o agente b, com estas novas restrições de integridade, e devido à possibilidade de abduzir predicados act e $happens$, irá abduzir as acções:

$$happens(e_2, t_1, t_2).$$
$$act(e_2, fazer\_dieta(a)).$$
$$1 \leq t_1 \leq t_2.$$

De modo a suportar o estado $magro(a)$ o agente necessita de abduzir as acções e os eventos necessários a provocar esse estado: $fazer\_dieta(a)$.

A acção abduzida deverá ser comunicada ao agente a, dado que, neste caso, só ele a poderá executar.

No capítulo 5 será apresentado o processo necessário a um correcto planeamento e geração de acções.

4.3 Revisão de estados mentais

O processo de actualização de estados mentais descrito na secção anterior pode introduzir inconsistências no modelo mental do agente. Na verdade, a introdução dos novos factos que descrevem os eventos que ocorreram ($happens(e_i, t, t'), act(e_i, a_i)$, com $1 \leq i \leq n$), poderá tornar contraditório o programa em lógica que modela o estado mental do agente/sistema computacional.

A contradição pode ser originada por duas causas distintas:

1. Contradição provocada pelos factos em si;

2. Contradição provocada pelas consequências desses factos.

A primeira causa resulta da violação de restrições de integridade pelos factos que foram acrescentados ao modelo do agente e será analisada na sub-secção 4.3.1. A segunda causa resulta de violações de restrições de integridade associadas aos efeitos das acções executadas, e será analisada na sub-secção 4.3.2. Note-se que as contradições lógicas são expressas pela violação de uma restrição de integridade do tipo:

$$\Leftarrow L, \neg L$$

4.3.1 Factos contraditórios

Os factos que descrevem os eventos que ocorreram podem introduzir uma contradição com o programa em lógica existente, através da violação de alguma restrição de integridade.

Como exemplo, suponhamos a seguinte descrição de acção de abrir uma janela (versão bastante simplificada):

$$abrir\_janela\ causes\ janela\_aberta$$
$$if\ janela\_fechada$$

Esta descrição tem a seguinte tradução em termos de regras de programação em lógica:

$$enabled(E, T_i) \leftarrow act(E, abrir\_janela), \quad (4.21)$$
$$holds\_at(janela\_fechada, T_i).$$
$$initiates(E, T_f, janela\_aberta) \leftarrow happens(E, T_i, T_f), \quad (4.22)$$
$$act(E, abrir\_janela),$$
$$holds\_at(janela\_fechada, T_i).$$

Suponhamos que, no modelo do agente/sistema computacional, a janela está aberta:

$$happens(e_0, t_0, t_0). \quad (4.23)$$
$$act(e_0, start). \quad (4.24)$$
$$initiates(e_0, t_0, janela\_aberta). \quad (4.25)$$

O agente reconheceu a seguinte evento:

$$happens(e, t, t'). \quad (4.26)$$
$$act(e, abrir\_janela), \quad (4.27)$$
$$t_0 \leq t \leq t'. \quad (4.28)$$

Nestas condições, o programa em lógica estendida que modela o agente é contraditório devido à violação da restrição de integridade 2.15 (impossibilidade de ocorrência de um evento sem que as suas pré-condições estejam satisfeitas):

$$\Leftarrow happens(E, T_i, T_f), not\ enabled(E, T_i).$$

De facto, não é possível provar $enabled(e, t)$, pois não é possível provar:

$$holds\_at(janela\_fechada, t).$$

A contradição pode ser eliminada através pelos seguintes processos:

1. Abduzindo uma acção que permita satisfazer a pré-condição desejada (por exemplo, um evento anterior que tivesse fechado a janela);

2. Assumindo que houve um reconhecimento incorrecto dos eventos ocorridos e, consequentemente, considerá-los inexistentes.

O primeiro processo (abdução de acções) é suportado pelo sistema que proponho através do recurso às regras que permitem a abdução de eventos e acções associadas ($happens/3$ e $act/2$). Estas regras permitem, também, ultrapassar situações de possível contradição devido à existência de conhecimento incompleto sobre o mundo. Ou seja, a contradição é evitada sempre que for possível abduzir acções que permitam satisfazer as condições necessárias para evitar a contradição.

No entanto, este processo não garante que é possível evitar todas as contradições. No exemplo apresentado, poderia não ser possível abduzir uma acção que tivesse como efeito que a janela ficasse fechada.

Nestas situações, é necessário recorrer ao processo de remoção de contradições proposto por [AP96] e apresentado no capítulo 2.

Para tal, é necessário definir os predicados revisíveis:

$$rev = \{not\ happens(E, T_i, T_f), not\ act(E, A)\}$$

Este conjunto de revisíveis significa que é possível rever a existência de eventos e das suas acções associadas, ou seja, é possível que o reconhecimento dos eventos tenha sido incorrecto e os eventos de facto não tenham ocorrido conforme foram reconhecidos. De facto, se a ocorrência de eventos e das acções associadas não puder ser incorporada no modelo do agente sem introduzir contradições, então deve ser revista a existência desses eventos.

No exemplo apresentado, a revisão céptica a 2 valores faria a revisão do estado lógico dos eventos e_0 e e, passando-os de verdadeiros a falsos.

Em determinadas situações, esta opção poderá ser muito drástica e ser preferível rever somente um dos eventos: o mais antigo ou o mais recente. Neste contexto, deve ser possível optar por um dos modelos estáveis estendidos dos submodelos minimais não-contraditórios. O processo de definição e selecção das soluções preferidas é efectuado recorrendo à metodologia proposta no sistema REVISE [SDP96] e descrito no capítulo 5.

4.3.2 Estado mental contraditório

Uma origem possível para as contradições provocadas pela descrição de eventos que ocorreram é a violação de restrições de integridade associadas aos efeitos das diversas acções.

Estas contradições podem ser distinguidas das contradições referidas na sub-secção anterior através da análise do conjunto de remoção de contradições do programa em lógica ([AP96]) e das restrições violadas.

As contradições provocadas pelos efeitos de acções estão associadas a regras de integridade do seguinte tipo (é contraditório suportar duas propriedades específicas e distintas P_1 e P_2 no mesmo instante de tempo):

$$\Leftarrow holds\_at(P_1, T), holds\_at(P_2, T).$$

Para este tipo de contradições, o processo de revisão deverá permitir definir preferências na ordem das soluções da revisão, nomeadamente permitindo rever a assumpção de que uma das propriedades não terminou com os novos eventos.

Supondo que um evento iniciou a propriedade P_2, e que a propriedade P_1 também é válida, então uma abordagem possível será assumir que o evento também termina a propriedade P_1 (revendo a assumpção de que não teria terminado). Se a existência de contradição se mantiver, dever-se-ão aplicar os procedimentos descritos para a revisão de factos contraditórios (abduzindo acções necessárias ou anulando a ocorrência de alguns eventos).

Como exemplo, vejamos a situação descrita na secção 4.2 em que, num instante de tempo t_1, o agente a acredita que a Catarina está no hospital. Num instante $t_2 > t_1$, outro agente informa-o de que afinal a Catarina está em casa, e ele, passa a acreditar nessa propriedade (consequência do acto de fala *inform* para agentes verdadeiros e da transferência de crenças para agentes ingénuos).

$$holds\_at(bel(a, em(hospital, catarina)), t_1).$$
$$holds\_at(bel(a, em(casa, catarina)), t_2).$$

O problema surge se existir a seguinte restrição de integridade

$$\begin{aligned} \Leftarrow \quad & holds\_at(bel(X, em(L_1, Y)), T), \\ & holds\_at(bel(X, em(L_2, Y), T), \\ & not(L_1 = L_2). \end{aligned} \tag{4.29}$$

Ou seja, um agente não pode acreditar que outro agente possa estar em dois locais distintos simultaneamente.

Nesta situação existe uma contradição: o agente a acredita quem em t_2, a Catarina está em dois locais distintos (em casa e no hospital).

O processo de revisão do estado mental deverá permitir obter e ordenar as soluções da revisão:

1. Terminar a crença de a Catarina está no hospital;

2. Terminar a crença de a Catarina está em casa;

Em termos gerais, a eliminação de contradições referentes ao estado mental do agente é efectuada do seguinte modo, após a actualização do estado mental devido ao evento e — $happens(e, t_i, t_f)$, $act(e, a)$:

1. Detecção da restrição de integridade violada. Este processo pode ser feito recorrendo ao cálculo dos conjuntos de remoção de contradições e do conjunto de suporte a literais [AP96].

2. Detecção da propriedade mais antiga, isto é, calcular os instantes de tempo que iniciaram as propriedades ($initiates(E, T, P)$) e relacioná-los. Se existir uma propriedade p anterior às outras, rever a assumpção de que o evento não termina a propriedade p. Se não existir uma propriedade anterior às outras, ou se o modelo continuar contraditório, aplica-se o processo descrito para a remoção de contradições de factos (abdução de acções e anulação de eventos).

Ou seja, o processo de revisão ordena as soluções de acordo com a seguinte ordem:

1. Rever término de propriedades;

2. Rever existência de eventos e acções associadas.

No capítulo 7 apresento uma descrição detalhada da construção de um protótipo que permite efectuar a revisão dos estados mentais ordenando as soluções obtidas, de acordo com o descrito nesta secção.

4.4 Inferência de atitudes

Um processo de participação em diálogos necessita de ter como suporte a inferência das atitudes dos agentes ao longo do tempo. Nas secções anteriores foi apresentado o modo como o sistema proposto suporta a actualização de novos eventos e a revisão do modelo do agente/sistema computacional. Nesta secção, pretende-se demonstrar qual a relação entre as atitudes (crenças, objectivos e intenções) e os diversos actos de fala.

4.4.1 Crenças

As crenças ($bel/2$) são atitudes criadas directamente através dos diversos actos de fala (um $inform$, por exemplo) ou como resultado de um processo de introspecção (uma crença pode ser consequência de um conjunto de outras crenças). De facto, qualquer acto de fala tem como efeito a criação de novas crenças, tanto no emissor como no receptor, do acto de fala.

A tabela seguinte apresenta as crenças que são criadas directamente nos emissores e nos receptores para cada acto de fala (assumindo emissores não mentirosos e receptores crédulos). Estas crenças são a consequência directa dos actos de fala e são suportadas pelos programas em lógica que modelam os agentes através de inferências dedutivas sobre as regras que definem os actos de fala. No caso dos emissores e dos receptores terem um comportamento distinto, os efeitos dos actos de fala estarão de acordo com o que foi apresentado no capítulo 3. Note-se que, no âmbito de diálogos, não é necessário descrever um nível de recursividade de crenças superior a 2 (ver [TCM96] para uma análise detalhada).

	emissor (s)	receptor (h)
inform(s,h,p)	bel(s,bel(h,bel(s,p)))	bel(h,bel(s,p))
informref(s,h,t,p)	bel(s,bel(h,bel(s,ref(t,p))))	bel(h,bel(s,ref(t,p)))
informif(s,h,p)	bel(s,bel(h,knowif(s,p)))	bel(h,knowif(s,p))
request(s,h,a)	bel(s,bel(h,int(s,a)))	bel(h,int(s,a))
askif(s,h,p)	bel(s,bel(h,ach(s,knowif(s,p))))	bel(h,ach(s,knowif(s,p)))
checkif(s,h,p)	bel(s,bel(h,ach(s,knowif(s,p))))	bel(h,ach(s,knowif(s,p)))
accept(s,a)	bel(s,bel(h,int(s,a)))	bel(h,int(s,a))
reject(s,h,a)	bel(s,bel(h,¬int(s,a)))	bel(h,¬int(s,a))

Tabela 4.1: Efeitos dos actos de fala

A partir das crenças que são causadas pelos diversos actos de fala, são criadas novas crenças que estão relacionadas directamente com as anteriores através de regras de racionalidade e de comportamento existentes no modelo do agente.

Assim, pode-se caracterizar a inferência das crenças de um agente como um processo dedutivo com base nos actos de fala e nos modelos de cada agente:

Definição 53 *Seja* $M = < At, Rc, Ac, T, Rr, CM >$ *o modelo de um agente a e* $a_1, ..., a_n$ *acções associadas aos eventos* $e_1, ..., e_n$ *que decorrem no intervalo de tempo* $]t, t']$. *A crença*

$$holds\_at(bel(a, X), t') \in actualiza(M, e_1 \times ... \times e_n)$$

pertence ao modelo resultante da actualização do estado mental do agente a sse :

$$holds\_at(bel(a, X), t') \in PCFXSM(At \cup RC \cup Ac \cup T \cup Rr \cup CM \cup \{act(e_1, a_1), happens(e_1, t, t'), ..., act(e_n, a_n), happens(e_n, t, t')\})$$

Esta definição significa que as únicas crenças de um agente são as crenças suportadas pelo modelo bem fundado preferido do programa em lógica que o define, depois de ter sido aumentado com os factos relativos ao novo acto de fala.

Note-se que o sistema proposto não efectua abduções sobre qualquer uma das atitudes: as abduções são efectuadas apenas sobre as acções e eventos respectivos (predicados $act/2$ e $happens/3$).

Como exemplo, vejamos uma variação sobre a primeira frase do exemplo apresentado no início do capítulo:

Ana: Quero falar com a Catarina.

Esta frase tem a seguinte correspondência em termos de actos de fala:

$$happens(e, t_1, t_2). \tag{4.30}$$

$$act(e, inform(ana, susana, int(ana, falar(ana, catarina)))). \tag{4.31}$$

Independentemente do *modelo mental* da Susana (ingénuo ou não), a semântica do programa em lógica que representa o modelo da Susana suportará o seguinte facto (utilizando a definição do acto *inform* apresentada anteriormente — regras 4.15 e 4.16):

$$holds\_at(bel(susana, bel(ana, int(ana, falar(ana, catarina)))), t_2).$$

Esta regra representa uma visão que o modelo da Susana possui sobre a Ana.

Deste modo foi efectuada a inferência dedutiva de uma crença de um agente a partir de um acto de fala. Devido à existência de regras de comportamento que relacionam as crenças dos diversos agentes, também seria possível efectuar a transferência de crenças e obter a seguinte inferência:

$$holds\_at(bel(susana, int(ana, falar(ana, catarina))), t_2).$$

Na sub-secção seguinte será focado o processo de inferência de objectivos.

4.4.2 Objectivos

Os objectivos são os estados que cada agente pretende que se venham a verificar e são modelados recorrendo ao predicado $ach/2$. Os objectivos não são originados directamente através dos diversos actos de fala. De facto, qualquer acto de fala nunca tem como efeito a criação directa de novos objectivos, mas sim a criação de crenças sobre objectivos.

A tabela 4.2 apresenta os actos de fala que têm como consequência a criação, nos emissores e nos receptores, de crenças sobre objectivos.

	emissor	receptor
askif(s,h,p)	bel(s,bel(h,ach(s,knowif(s,p)))))	bel(h,ach(s,knowif(s,p))))
checkif(s,h,p)	bel(s,bel(h,ach(s,knowif(s,p)))))	bel(h,ach(s,knowif(s,p))))

Tabela 4.2: Objectivos consequência dos actos de fala

Estas crenças sobre objectivos, consequência directa dos actos de fala, podem transformar-se em objectivos dos próprios agentes através das regras que definem a transferência de objectivos a partir das crenças existentes (apresentadas no capítulo 3).

A regra para agentes cooperativos é (3.52):

$$holds\_at(ach(A, P), T) \leftarrow \quad holds\_at(bel(A, ach(B, P)), T), \quad holds\_at(bel(A, cooperativo(A)), T). \tag{4.32}$$

para agentes não cooperativos passivos (3.54):

$$holds\_at(\neg ach(A, P), T) \leftarrow \quad holds\_at(bel(A, ach(B, P)), T), \quad holds\_at(bel(A, nao\_cooperativo\_passivo(A)), T). \tag{4.33}$$

e para agentes não cooperativos activos:

$$holds\_at(ach(A, \neg P), T) \leftarrow holds\_at(bel(A, ach(B, P)), T), \quad (4.34)$$
$$holds\_at(bel(A, nao\_cooperativo\_activo(A)), T).$$

Os objectivos de cada agente são atitudes fundamentais para o processo de planeamento, quer na componente de reconhecimento (a analisar neste capítulo), quer na componente de geração (a analisar no próximo capítulo). De facto, o processo de planeamento não é mais do que tentar reconhecer (ou gerar) o plano de acções do interlocutor (ou do próprio sistema) que poderá levar a um determinado estado.

A inferência dos objectivos de um agente é, também, um processo dedutivo com base nos actos de fala e nos modelos de cada agente:

Definição 54 *Seja* $M = < At, Rc, Ac, T, Rr, CM >$ *o modelo de um agente* a *e* $a_1, ..., a_n$ *acções associadas aos eventos* $e_1, ..., e_n$ *que decorrem no intervalo de tempo* $]t, t']$. *O objectivo*

$$holds\_at(ach(a, X), t') \in actualiza(M, e_1 \times ... \times e_n)$$

pertence ao modelo resultante da actualização do estado mental do agente a sse

$$holds\_at(ach(a, X), t') \in PCFXSM(At \cup RC \cup Ac \cup T \cup Rr \cup CM \cup \{act(e_1, a_1), happens(e_1, t, t'), ..., act(e_n, a_n), happens(e_n, t, t')\})$$

Os objectivos de um agente são os objectivos suportados pelo modelo bem fundado preferido do programa em lógica que representa esse agente.

Como exemplo, vejamos as consequências de uma frase que transmite objectivos:

A: Quero ficar rico!

Esta frase tem a seguinte correspondência em termos de actos de fala:

$$happens(e, t_1, t_2). \quad (4.35)$$
$$act(e, inform(a, b, ach(a, rico(a)))). \quad (4.36)$$

A definição de *inform* (regras 4.15 e 4.16) permite suportar o seguinte facto:

$$holds\_at(bel(b, bel(a, ach(a, rico(a)))), t_2).$$

Este facto representa a crença de b em que a acredita que tem como objectivo ser rico.

Utilizando as regras de transferência de crenças (regra 3.16 e 3.17) obtém-se:

$$holds\_at(bel(b, ach(a, rico(a))), t_2).$$

Este facto representa a crença de b em que a tem como objectivo ficar rico.

Supondo que b é um agente cooperativo (regra 4.32), temos:

$$holds\_at(ach(b, rico(a)), t_2).$$

Este facto representa o objectivo de b em que a seja rico.

Deste modo foram inferidas crenças sobre objectivos a partir do acto de fala e, com base nessas crenças, deduziram-se novos objectivos.

Na sub-secção seguinte será focado o processo de inferência de intenções.

4.4.3 Intenções

As intenções ($int/2$) são as atitudes que representam as acções que cada agente pretende que venham a ser realizadas por ele ou por outrém. Do mesmo modo que os objectivos, as intenções não são originadas directamente através dos diversos actos de fala. Os actos de fala podem ter como efeito somente a criação de crenças sobre intenções.

A tabela seguinte apresenta os actos de fala que têm como consequência directa a criação, nos emissores e nos receptores, de crenças sobre intenções.

	emissor	receptor
inform(s,h,int(s,a))	bel(s,bel(h,bel(s,int(s,a))))	bel(h,bel(s,int(s,a)))
accept(s,a)	bel(s,bel(h,int(s,a)))	bel(h,int(s,a))
reject(s,h,a)	bel(s,bel(h,¬int(s,a)))	bel(h,¬int(s,a))

Tabela 4.3: Intenções consequência dos actos de fala

As crenças sobre intenções podem transformar-se em intenções dos próprios agentes através das regras de comportamento que definem a cooperatividade entre os agentes. Para um agente cooperativo a um nível máximo, conforme apresentado no capítulo anterior, teremos que o agente toma como suas as intenções que crê que os outros possuem:

$$holds\_at(int(H, A), T) \leftarrow holds\_at(bel(H, int(S, A)), T), \quad (4.37)$$
$$holds\_at(bel(H, cooperativo(H)), T).$$

Neste contexto, a inferência das intenções de um agente é um misto de processo dedutivo e abdutivo, com base nos acções realizadas, nos modelos de cada agente e no planeamento das acções futuras:

Definição 55 *Sejam $M = < At, Rc, Ac, T, Rr, CM >$ o modelo de um agente a e $a_1, ..., a_n$ acções associada aos eventos $e_1, ..., e_n$ que decorrem no intervalo de tempo $]t, t'[$. A intenção*

$$holds\_at(int(a, X), t') \in actualiza(M, e_1 \times ... \times e_n)$$

pertence ao modelo resultante da actualização do estado mental do agente a sse :

$$holds\_at(int(a, X), t') \in PCFXSM(At \cup RC \cup Ac \cup T \cup Rr \cup CM \cup \{act(e_1, a_1), happens(e_1, t, t'), ..., act(e_n, a_n), happens(e_n, t, t')\})$$

Esta definição significa que as intenções de um agente são as intenções suportados pelo modelo bem fundado preferido do programa em lógica que representa o estado mental do agente.

As intenções são, também, criadas durante o processo de planeamento de acções para atingir objectivos, que será objecto de análise mais detalhado no próximo capítulo. Neste processo, para cada acção abduzida como necessária para atingir os objectivos do agente, é criada uma intenção de a realizar. Posteriormente, as intenções de realizar acções são transformadas em acções propriamente ditas. Este outro processo de criar intenções é definido do seguinte modo (uma descrição mais detalhada do planeamento abdutivo é apresentada no capítulo 5):

Definição 56 *Seja M o modelo do agente a. No instante de tempo t, o processo de planeamento abdutivo das acções a realizar para satisfazer os objectivos do agente inicia as intenções:*

$$holds\_at(int(a, X_i), t).$$

se:

$$\{happens(e_i, t_i, t_i'), act(e_i, X_i)\} \subset PCFXSM(M \cup IC(\{holds\_at(p_1, t_\infty) \Leftarrow, ..., holds\_at(p_n, t_\infty) \Leftarrow, \}))$$

onde t_∞ representa um tempo futuro posterior e IC representa o conjunto de novas restrições de integridade associadas aos objectivos $p_1, ..., p_n$:

$$holds\_at(ach(a, p_i), t) \in PCFXSM(M),\ 1 \leq i \leq n.$$

e

$$happens(e_i, t_i, t_i') \notin PCFXSM(M)$$

Esta definição significa que são criadas as intenções de realizar as acções que tenham como consequência as propriedades que o agente deseja que se venham a verificar (os seus objectivos).

Conforme foi referido, este processo é um processo abdutivo e será analisado em maior detalhe no próximo capítulo.

Como exemplo, vejamos, novamente, as consequências de uma variação à primeira frase do diálogo apresentado no início do capítulo:

Ana: Quero telefonar à Catarina para o hospital.

Conforme apresentado na secção sobre a inferência de crenças, é suportado o seguinte facto:

$$holds\_at(bel(susana, int(ana, telefonar(ana, catarina, hospital))), t_2).$$

Assumindo que a Susana tem um comportamento cooperativo, então seria válida a transferência de intenções entre os dois agentes (regra 4.37):

$$holds\_at(int(susana, telefonar(ana, catarina, hospital))), t_2).$$

Este processo permitiu a criação de novas intenções a partir de actos de fala. Conforme referido, o outro método que permite a criação de intenções é baseado no planeamento de acções e no raciocínio abdutivo (capítulo 5).

4.5 Reconhecimento de planos

O processo de inferência de atitudes apresentado na secção anterior permite estabelecer o *estado mental* do agente em relação ao mundo que o rodeia e, em especial, em relação aos outros agentes, após a ocorrência de eventos. O passo seguinte no processo de participação em diálogos é o processo de reconhecimento dos planos dos outros agentes, como componente fundamental para o planeamento e geração das próprias acções do agente.

Tradicionalmente, um plano de um agente é representado pela sequência de acções que esse agente pretende que venham a ser realizadas. Este conceito pode ser representado considerando que um plano de um agente, num dado instante de tempo, é dado pelo conjunto de intenções sobre acções que ele deseja que se realizem.

Definição 57 *Do ponto de vista do agente a, $P_a(b,t)$ é um plano do agente b no instante de tempo t e é definido por:*

$$P_a(b,t) = \{int(b,X) : holds\_at(bel(a, int(b,X)), t) \in PCFXSM(M_a)\}$$

onde M_a é o modelo de a.

Ou seja, em cada instante, um agente tenta identificar as intenções dos outros agentes interlocutores. Após a realização de um evento, o modelo do agente é revisto e os planos são novamente actualizados.

Recorrendo à definição apresentada e aos modelos de agentes descritos anteriormente, conclui-se que é possível um agente ter um plano próprio $P_a(a,t)$, que representa as acções que ele pretende que sejam realizadas:

$$P_a(a,t) = \{int(a,X) : holds\_at(bel(a, int(a,X)), t) \in PCFXSM(M_a)\}$$

e ter, também, uma visão sobre os planos dos outros agentes:

$$P_a(b,t) = \{int(b,X) : holds\_at(bel(a, int(b,X)), t) \in PCFXSM(M_a)\}$$

Note-se que, no caso de plano próprio, as intenções do agente também fazem parte de $P_a(a,t)$, já que a regra de necessidade das crenças (3.3) garante que:

$$holds\_at(bel(A,P),T) \leftarrow holds\_at(P,T)$$

e, nomeadamente:

$$holds\_at(bel(A,int(A,X)),T) \leftarrow holds\_at(int(A,X),T)$$

Na sequência deste trabalho iremos utilizar o predicado $P_A(B,T)$ para representar os planos dos agentes com o significado agora apresentado.

O reconhecimento dos planos dos agentes interlocutores de um dado agente resume-se, assim, à inferência das crenças sobre intenções que esse agente possui num dado instante de tempo.

Este processo de reconhecimento de planos pode ser visto como um misto entre as abordagens clássicas tipo-STRIPS ([FN71, Lit85]) e a abordagem de planos como estados mentais de Pollack ([Pol90]). De facto, as acções são representadas em termos das suas pré-condições e efeitos, como na abordagem clássica; no entanto, o processo de reconhecimento de planos baseia-se numa teoria de atitudes que define os estados mentais dos agentes e na semântica bem fundada de programas em lógica estendida, de modo a inferir os planos que são suportados pelo modelo do agente.

Nas sub-secções seguintes iremos apresentar variações sobre o exemplo apresentado no início do capítulo, de modo a ilustrar situações típicas de reconhecimento de planos em diálogos, realçando os problemas existentes nos trabalhos actuais e o modo como o sistema proposto os resolve. Assim, na sub-secção 4.5.1 apresento uma situação de diálogo "correcto", na sub-secção 4.5.2 ilustro uma situação de diálogo em que há planos incorrectos por parte de um agente e na sub-secção 4.5.3 apresento um diálogo em que há situações de ambiguidade.

4.5.1 Planos correctos

Nesta sub-secção ir-se-á analisar uma variação sobre o diálogo cooperativo apresentado no início do capítulo em que os planos subjacentes aos actos de fala dos agentes não apresentam qualquer incorrecção ou ambiguidade. Assume-se, também, que os agentes são cooperativos, sinceros e crédulos.

Ana: Quero falar com a Catarina. Qual o número de telefone do hospital?

Susana: É o 111111.

Esta situação de diálogo é a ideal, no sentido em que não existem incorrecções ou ambiguidades. Tanto do ponto de vista da Ana como da Susana a Catarina ainda se encontra no hospital. Os planos são correctos, dado permitirem atingir e satisfazer os objectivos

desejados. Este é o tipo de situação que é suportada pelos sistemas de participação em diálogos já existentes. No entanto, é de realçar que existe uma relação de causalidade entre as duas frases do primeiro agente: telefonar é uma acção que permite realizar a acção de falar com alguém. Esta relação é captada pelo sistema proposto através do recurso a regras de programação em lógica.

Conforme foi referido no início deste capítulo, a intervenção do primeiro interlocutor deverá dar origem aos seguintes actos de fala:

$$happens(e_1, 1, 1). \tag{4.38}$$

$$act(e_1, request(ana, susana, \tag{4.39}$$
$$\quad informref(susana, ana, N, telefone(N, hospital)))).$$

$$happens(e_2, 1, 1). \tag{4.40}$$

$$act(e_2, inform(ana, susana, int(ana, falar(ana, catarina)))), \tag{4.41}$$

Estes factos representam os actos de fala efectuados, assumindo-os simultâneos e com uma duração nula. Esta simplificação não retira expressividade ao exemplo, sendo que a demonstração a efectuar também seria válida para outros pressupostos (eventos não simultâneos e com uma duração não nula).

Neste sistema, é necessário a descrição de quatro regras específicas do domínio para relacionar as diversas acções (a adicionar ao termo CM do modelo dos agentes).

As regras do domínio são:

$$bel(X, int(Y, telefonar(Y, Z, L)) \tag{4.42}$$
$$if$$
$$bel(X, int(Y, informref(X, Y, N, telefone(N, L))),$$
$$bel(X, int(Y, falar(Y, Z))).$$

$$bel(X, bel(Y, em(L, Z))) \tag{4.43}$$

Esta regra significa que, se um agente X acredita que outro agente Y pretende ser informado sobre um dado número de telefone de um local L e se acredita que Y tem como intenção falar com Z, então X acredita que Y pretende telefonar para L (ou que se telefone para L) para falar com Z.

A segunda regra é:

$$bel(X, bel(Y, em(L, Z))) \tag{4.44}$$
$$if$$
$$bel(X, int(Y, telefonar(Y, Z, L))).$$

Esta regra significa que, se um agente X acredita que outro agente Y pretende telefonar para um local L para falar com um agente Z, então X acredita que Y acredita que Z está em L.

A terceira regra é:

$$bel(X, int(Y, falar(Y, Z))) \quad (4.45)$$
$$if$$
$$bel(X, int(Y, telefonar(Y, Z, L))).$$

Esta regra significa que, se um agente X acredita que outro agente Y pretende telefonar para um local L para falar com um agente Z, então X acredita que Y tem como objectivo falar com Z.

A quarta regra é:

$$int(X, telefonar(Y, Z, L_2)) \quad (4.46)$$
$$if$$
$$bel(X, em(L_2, Z)),$$
$$bel(X, int(Y, telefonar(Y, Z, L_1))).$$

Esta regra significa que, se um agente X acredita que outro agente Z está no local L_2 e se acredita que Y tem como intenção telefonar-lhe para um outro local L_1, então X passa a pretender que Y telefone para L_2 para falar com Z (o agente é cooperativo).

Estas regras, que relacionam crenças sobre o domínio do exemplo, dão origem às seguintes regras de programação em lógica, de acordo com o processo descrito no capítulo 2:

$$holds\_at(bel(X, int(Y, telefonar(Y, Z, L))), T) \leftarrow \quad (4.47)$$
$$\quad holds\_at(bel(X, int(Y, informref(X, Y, N, telefone(N, L)))), T),$$
$$\quad holds\_at(bel(X, int(Y, falar(Y, Z))), T).$$
$$holds\_at(bel(X, bel(Y, em(L, Z))), T) \leftarrow \quad (4.48)$$
$$\quad holds\_at(bel(X, int(Y, telefonar(Y, Z, L)))).$$
$$holds\_at(bel(X, int(Y, falar(Y, Z))), T) \leftarrow \quad (4.49)$$
$$\quad holds\_at(bel(X, int(Y, telefonar(Y, Z, L)))).$$
$$holds\_at(int(X, telefonar(Y, Z, L_2)), T) \leftarrow \quad (4.50)$$
$$\quad holds\_at(bel(X, em(L_2, Z)), T),$$
$$\quad holds\_at(bel(X, int(Y, telefonar(Y, Z, L_1))), T).$$

Em adição, existem duas restrições de integridade que definem a impossibilidade de um agente acreditar que alguém pode estar em dois lugares ao mesmo tempo e a impossibilidade de um agente acreditar que se pode telefonar a alguém para um dado lugar

e não acreditar que esse alguém está nesse lugar:

$$\begin{aligned} \Leftarrow \quad & holds\_at(bel(X, em(L_1, Y)), T), \\ & holds\_at(bel(X, em(L_2, Y)), T), \\ & not(L_1 = L_2). \end{aligned} \tag{4.51}$$

$$\begin{aligned} \Leftarrow \quad & holds\_at(bel(X, int(X, telefonar(Y, Z, L_1))), T), \\ & holds\_at(bel(X, em(L_2, Z)), T), \\ & not(L_1 = L_2). \end{aligned} \tag{4.52}$$

Do ponto de vista da Susana, o plano de Ana após o instante $t = 1$, é representado por $P_{susana}(ana, 1)$ e contém as seguintes intenções:

$$int(ana, falar(ana, catarina)) \tag{4.53}$$

$$int(ana, telefonar(ana, catarina, hospital)). \tag{4.54}$$

Este plano indica que a Ana pretende pretende falar com a Catarina e pretende telefonar para o hospital para falar com a Catarina.

Vejamos como o sistema suporta a inferência deste plano:

- A partir da definição do acto de fala *inform* para agentes não mentirosos (regra 3.29), da transferência de informação para agentes crédulos (3.16 e 3.17) e do acto de fala e_2 realizado (factos 4.40 e 4.41) é obtido:

$$holds\_at(bel(susana, int(ana, falar(ana, catarina))), 1) \tag{4.55}$$

- Pela definição de *request* (regra 3.32) e pelos factos (4.38 e 4.39):

$$\begin{aligned} & holds\_at(bel(susana, int(ana, \\ & informref(susana, ana, N, telefone(N, hospital)))), 1) \end{aligned} \tag{4.56}$$

- Pela primeira regra do domínio apresentada (4.47) e pelas propriedades anteriores (4.55 e 4.56):

$$holds\_at(bel(susana, int(ana, telefonar(ana, catarina, hospital))), 1). \tag{4.57}$$

Além disso, pela segunda regra do domínio (4.48) e pela intenção (4.57) obtém-se:

$$holds\_at(bel(susana, bel(ana, em(hospital, Catarina))), 1). \tag{4.58}$$

O plano da Ana, conforme foi reconhecido pela Susana como estando subjacente aos actos de fala e_1 e e_2 da Ana, contém a intenção que lhe permite servir de base ao planeamento dos seus próprios actos de fala (capítulo 5).

Por outro lado, as intenções da Susana após a intervenção da Ana, são:

$$int(susana, telefonar(ana, catarina, hospital)). \tag{4.59}$$

$$int(susana, informref(susana, ana, N, telefone(N, hospital))). \tag{4.60}$$

Vejamos, novamente, como estas inferências são efectuadas:

- A primeira representa a intenção da Susana de que a Ana telefone à Catarina para o hospital e é obtida a partir da definição de agente cooperativo (apresentada no capítulo anterior), que define que um agente adopta como suas as intenções que acredita que os outros agentes têm. Neste caso, a intenção é uma consequência da intenção da Ana derivada anteriormente (4.57), da regra de cooperatividade (3.47) e da regra de necessidade de crenças (3.3).

- A segunda intenção representa a intenção da Susana em satisfazer o pedido da Ana e é suportada a partir da definição de *request* e da cooperatividade da Susana. De facto,

$$holds\_at(bel(susana, int(ana, \\ informref(susana, ana, N, telefone(N, hospital)))), 1)$$

  é obtido a partir da definição de *request* (3.32) e do evento e_1.

  A crença:

$$holds\_at(bel(susana, int(susana, \\ informref(susana, ana, N, telefone(N, hospital)))), 1).$$

  é obtido a partir das regras de cooperatividade dos agentes (3.47 e 3.3).

Com base neste plano, e consultando uma base de conhecimentos com números de telefones, é possível suportar a geração de actos de fala equivalentes aos apresentados no exemplo. Este processo será apresentado em detalhe no próximo capítulo.

4.5.2 Planos incorrectos

Nesta sub-secção ir-se-á analisar o diálogo cooperativo apresentado no início do capítulo em que o plano subjacente ao acto de fala do primeiro agente é incorrecto do ponto de vista do segundo agente:

Ana: Quero falar com a Catarina. Qual o número de telefone do hospital?

Susana: Ela já está em casa. O número de telefone da casa dela é o 999999.

Este é um tipo de situações que surge com alguma frequência em diálogos devido à existência de modelos do mundo distintos por parte dos agentes. Assim, o plano da Ana é correcto do seu ponto de vista, sendo, no entanto, incorrecto do ponto de vista da Susana — existem crenças distintas sobre o local onde está a Catarina. Neste exemplo iremos assumir que a Susana é crédula, mas não ingénua; isto é, acredita no que lhe é transmitido, desde que não vá contra as suas próprias crenças.

Conforme foi referido na secção anterior, a intervenção do primeiro interlocutor dá origem aos seguintes actos de fala:

$$happens(e_1, 1, 1). \tag{4.61}$$

$$act(e_1, request(ana, susana, \tag{4.62}$$
$$\quad informref(susana, ana, N, telefone(N, hospital)))).$$

$$happens(e_2, 1, 1). \tag{4.63}$$

$$act(e_2, inform(ana, susana, int(ana, falar(ana, catarina)))), \tag{4.64}$$

Assume-se a existência das regras específicas do domínio definidas anteriormente (4.47 e 4.49) e das restrições de integridade 4.51 e 4.52.

Do mesmo modo que na secção anterior, no modelo da Susana, o plano de Ana após o instante $t = 1$, $P_{susana}(ana, 1)$ contém a intenção:

$$int(ana, telefonar(ana, catarina, hospital)). \tag{4.65}$$

A justificação da obtenção deste plano é idêntica à dada na secção anterior.

Este exemplo diverge do anterior nos resultados da inferência do plano da Susana após a intervenção da Ana: $P_{susana}(susana, 1)$.

A base das diferenças observadas está na crença existente no modelo da Susana de que a Catarina está em casa.

Suponhamos, pois, que existiu uma acção anterior ao diálogo que iniciou esta crença. Nestas condições, o facto seguinte é suportado pelo modelo da Susana:

$$holds\_at(bel(susana, em(casa(catarina), catarina)), 0). \tag{4.66}$$

O plano das acções a realizar pela Susana, após a frase da Ana, $P_{susana}(susana, 1)$, será:

$$int(susana, telefonar(ana, catarina, casa(catarina))). \tag{4.67}$$

Esta intenção representa o plano da Ana e é obtida do seguinte modo:

- O plano inferido pela Susana no exemplo anterior

  $$int(susana, telefonar(ana, catarina, hospital)).$$

  não pode ser gerado porque viola a segunda restrição de integridade apresentada 4.52 (não é possível ter como intenção telefonar a alguém para um dado local e acreditar que esse alguém está noutro local — neste caso a Ana acredita que a Catarina está em casa).

- A quarta regra do domínio (4.50) permite corrigir a acção de telefonar e obter:

  $$int(susana, telefonar(ana, catarina, casa(catarina))) \tag{4.68}$$

De modo a suportar uma participação *inteligente* em diálogos, um agente cooperativo deve tentar remover os possíveis obstáculos à satisfação dos objectivos dos outros agentes. Nesta situação, o plano da Susana deveria, também, conter:

$$int(susana, informref(susana, ana, N, telefone(N, casa(catarina)))).$$

No entanto, este processo está directamente relacionado com a geração de planos e será somente abordado no próximo capítulo.

Este exemplo realça uma característica importante neste sistema: a capacidade de, devido a conhecimento adquirido anteriormente, o sistema considerar planos incorrectos e sugerir alternativas.

4.5.3 Planos omissos

Nesta sub-secção ir-se-á analisar uma variação ao diálogo cooperativo apresentado no início do capítulo em que existe uma omissão subjacente ao acto de fala do primeiro agente:

Ana: Quero falar com a Catarina.

Susana: Podes-lhe telefonar, o número de casa dela é o 999999.

Neste exemplo, o primeiro agente só transmite uma intenção de realizar um acção, e não o processo como pretende vir a realizá-la. Supondo que existem diversos modos de realizar a acção *falar com alguém* (telefonar ou ir ao local onde o agente se encontra), existe uma ambiguidade em relação ao método que o primeiro agente pretende escolher.

Este é um tipo de situações que também surge com frequência em diálogos, nomeadamente em diálogos cooperativos em que um agente tem determinados objectivos e intenções, mas não sabe como os satisfazer. Cabe ao segundo agente transmitir-lhe o plano correcto para a satisfação dos seus objectivos e intenções.

Os sistemas propostos por Pollack ([Pol86, Pol90]) e por Litman e Allen ([Lit85, LA87]) permitem gerir situações semelhantes a estas, embora não situações de remoção de possíveis contradições.

Neste exemplo, uma segunda frase da Ana poderia ser:

Ana: Não! Eu quero falar pessoalmente com ela.

Susana: Ela mora na Rua 25, nº 2.

Esta frase obrigaria a um novo planeamento que não contemplasse a acção de telefonar.

Uma das vantagens do sistema proposto nesta tese é que permite esta revisão sem perder o planeamento anterior; tem memória sobre os estados mentais anteriores. Nas outras abordagens referidas este histórico perde-se, e não está previsto poder refazer os processos de inferência anteriores.

Recorrendo a este exemplo, é possível identificar e seleccionar um plano como correcto num dado instante de tempo (*telefonar* como acção que permite *falar com alguém*) e seleccionar um outro plano, também correcto, num instante de tempo posterior (*encontrar-se com alguém* como acção que também permite *falar com esse alguém*). Em cada instante de tempo, o processo de planeamento abdutivo (capítulo 5) obtém a sequência de acções que poderá permitir atingir osobjectivos existentes.

De modo a analisarmos este exemplo, temos que a intervenção do primeiro interlocutor dá origem ao seguinte acto de fala:

$$happens(e_1, 1, 1). \tag{4.69}$$

$$act(e_1, inform(ana, susana, int(ana, falar(ana, catarina)))). \tag{4.70}$$

Assume-se a existência das regras específicas do domínio apresentadas anteriormente. Em adição, existem regras que definem as acções falar, telefonar e ir a um local:

$$\begin{aligned} &falar(Y, Z)\ if \\ &em\_contacto(Y, Z). \end{aligned} \tag{4.71}$$

$$\begin{aligned} &telefonar(Y, Z, L)\ causes\ em\_contacto(Y, Z) \\ &if \\ &em(L, Z). \end{aligned} \tag{4.72}$$

$$\begin{aligned} &ir\_a(Y, L)\ causes\ em\_contacto(Y, Z) \\ &if \\ &em(L, Z). \end{aligned} \tag{4.73}$$

Estas regras definem que dois agentes podem falar se estiverem em contacto e que a acção de telefonar (ou ir) para um local causa esse contacto, se o outro agente se encontrar nesse local.

No modelo da Susana, o plano de Ana após o instante $t = 1$, $P_{susana}(ana, 1)$ contém a seguinte intenção:

$$int(ana, falar(ana, catarina))$$

Utilizando para esta inferência as regras do acto de fala *inform* e de cooperatividade, de um modo análogo ao dos exemplos anteriores.

O modelo da Susana contém a seguinte intenção, devido à cooperatividade:

$$int(susana, falar(ana, catarina))$$

O processo de planeamento e geração proposto, e descrito em detalhe no próximo capítulo, tenta satisfazer as pré-condições necessárias à acção pretendida ($em\_contacto/2$). Para tal irá abduzir as acções que permitem satisfazer essas pré-condições. Este processo permite a obtenção de dois planos possíveis para a Susana após a frase da Ana $P(Susana, 1)$:

$$int(susana, telefonar(ana, catarina, casa(catarina))). \tag{4.74}$$

$$int(susana, ir\_a(ana, casa(catarina))). \tag{4.75}$$

A opção entre os dois planos possíveis é efectuada pelo processo de ordenação de soluções abdutivas que, conforme foi referido anteriormente, será descrito no próximo capítulo e que permite hierarquizar os planos possíveis de serem realizados (telefonar é preferível a deslocar-se, ou vice versa).

4.6 Conclusões

Neste capítulo foram descritos mecanismos que permitem a inferência de atitudes de um agente/sistema computacional e o reconhecimento dos planos dos agentes participantes em diálogos.

O processo proposto permite efectuar a actualização e a revisão das atitudes do agente em relação a si próprio e aos seus interlocutores, com base nas acções que são realizadas, nomeadamente nos diversos actos de fala. A função de actualização e de revisão foi apresentada, sendo as novas atitudes suportadas pela revisão do modelo do programa em lógica estendida que representa o agente.

A ordenação das soluções da revisão é efectuada com base na definição de preferência entre os predicados revisíveis: terminar propriedades; rever eventos e acções associadas. Este mecanismo permite lidar com um conjunto de situações em que existe informação incompleta, seja sobre as consequências das acções, seja sobre os eventos que de facto ocorreram.

Com base no processo de actualização e revisão de atitudes é possível inferir, dedutiva ou abdutivamente, as novas atitudes e os planos dos agentes. Os planos dos agentes são representados pelo conjunto de intenções que um agente pretende que venham a ser executados. Este processo permite efectuar, também, a geração dos planos do próprio agente/sistema computacional, através da abdução das acções necessárias para satisfazer os seus objectivos e as suas intenções. O processo de abdução poderá, também, criar novos objectivos e intenções. Este processo será, no entanto, analisado no próximo capítulo.

A abordagem que proponho permite ultrapassar um conjunto de problemas existentes nos sistemas de diálogos:

1. Integra as componentes de:

   - representação dos estados mentais do agente;

- planeamento de acções a realizar.

Um agente reconhece as intenções e os planos parciais dos outros agentes através da informação veículada pelos diversos actos de fala.

Este processo é inovador e permite integrar as diferentes perspectivas existentes de outros autores:

- Planeamento de acções sem uma correspondente representação em termos de estado mental (Allen [AKPT91, AF94], Litman [Lit85, LA87], Carberry [Car85, Car88]);
- Planeamento como um estado mental (Pollack [Pol86, Pol90], Cohen e Levesque [CL90a], Perrault [Per90]).

2. O agente/sistema computacional possui memória sobre as suas atitudes e, consequentemente, sobre os seus planos. Esta característica permite-lhe que, num dado instante, tenha uma determinada atitude e, posteriormente, a altere, actualizando o seu estado mental. No entanto, é sempre possível obter as atitudes que eram válidas num dado intervalo de tempo e os eventos que se realizaram.

   A utilização de uma representação temporal integrada com o processo de representação do estado mental do agente e das suas atitudes permite ultrapassar um conjunto de problemas existentes em sistema anteriores. De facto, os sistemas de gestão de diálogos de Litman, Carberry e Pollack permitem a detecção de algum tipo de situações em que existem planos incorrectos por parte dos agentes interlocutores. A solução que propõem não permite, no entanto, suportar a co-existência de visões distintas sobre a mesma realidade. Um plano incorrecto deverá ser alterado e corrigido. Na solução proposta neste trabalho, um plano inferido para um agente A pode ser considerado incorrecto por outro agente B (dado não permitir atingir os seus objectivos), por representar a visão que o agente B tem sobre o plano do agente A. No entanto, do ponto de vista de A, o seu plano pode ser correcto (os agentes possuem *estados mentais* distintos).

3. Permite ultrapassar situações de inconsistência no estado mental do agente/sistema computacional.

   Os processos de actualização do estado mental e de planeamento de acções a realizar podem introduzir situações de inconsistência no estado mental do agente. De facto, um determinado evento pode iniciar uma crença contraditória com crenças anteriores. A revisão do estado mental permite terminar atitudes (em especial, crenças) e, deste modo, eliminar situações de inconsistência.

   O processo de revisão do estado mental do agente, eliminando inconsistências e terminando determinadas atitudes, é uma característica inovadora deste trabalho e permite suportar um conjunto mais vasto de situações do que os sistemas anteriores.

De facto, nenhum dos trabalhos referidos (Litman, Allen, Carberry, Pollack, Cohen, Levesque e Perrault) permite a definição de restrições de integridade sobre o modelo dos agentes e a definição de procedimentos que permitam a eliminação de eventuais inconsistências com essas restrições.

Em suma, o processo proposto para a inferência de atitudes, actualização e revisão de estados mentais, e para o reconhecimento de planos é um mecanismo fundamental para uma participação mais inteligente de um agente/sistema computacional em diálogos.

Capítulo 5

Participação em diálogos

Neste capítulo apresento o processo completo de participação em diálogos. Na secção 5.2 é descrito o processo de identificação (parcial) dos planos dos diversos agentes interlocutores. Na secção 5.3 é descrito o processo de geração dos planos do próprio agente, através da inferência dos seus próprios objectivos e da abdução das acções necessárias para que esses objectivos sejam atingidos. A seguir, na secção 5.4, apresento o processo de geração dos actos de fala abstractos correspondentes, não traduzidos em Língua Natural. Finalmente, na secção 5.5, é descrito o processo de gestão dos diversos componentes apresentados.

De modo a permitir uma melhor compreensão de todo este processo apresento, na secção 5.6, um exemplo típico de participação em diálogos.

Finalmente, e em conclusão, na secção 5.7 sumarizo as características mais importantes do processo proposto neste capítulo, realçando os aspectos inovadores.

5.1 Introdução

A participação *inteligente* em diálogos por parte de um agente requer diversas actividades:

1. Inferência das atitudes dos agentes interlocutores, com base nos actos de fala e no modelo desses agentes;
2. Actualização e revisão das suas próprias atitudes, com base nas atitudes dos seus interlocutores e no seu modelo de comportamento;
3. Geração de planos, com base nas suas atitudes, nomeadamente nos seus objectivos e nas acções necessárias para os atingir;
4. Selecção do plano mais adequado, caso existam vários planos possíveis;
5. Execução do plano seleccionado.

Os dois primeiros pontos foram abordados no capítulo anterior, onde se apresentou o processo que permite a um agente inferir as atitudes de outros agentes com base nos actos de fala realizados por um interlocutor e no modelo que o agente tem desse interlocutor. Nomeadamente, foi apresentado um sistema de actualização e de revisão de atitudes do agente que permite obter em cada instante as atitudes suportadas pelo seu *estado mental*. É com base nas atitudes inferidas que é efectuada a identificação parcial dos planos dos interlocutores, sendo estes não mais do que o conjunto de intenções de realizar acções que o agente acredita que o interlocutor possui num dado instante.

O passo seguinte no processo de participação em diálogos, ou em qualquer processo de interacção com outros agentes, é a geração de planos próprios adequados a atingir os objectivos que o agente possui num dado instante. O processo proposto efectua esta geração através de um processo abdutivo sobre acções passíveis de serem realizadas. Ao contrário dos planos do interlocutor, aqui os planos não se limitam a conjuntos de intenções de realizar acções, mas abduzem posteriormente as próprias acções através de um processo de geração. No entanto, não é tratada a problemática da interacção ou temporização das acções, que constitui matéria substancial da investigação em planeamento.

Como exemplo, temos a situação descrita no capítulo anterior em que um agente pretende ficar magro (não o sendo no instante actual t_1), e em que uma acção possível de executar é fazer dieta:

$$holds\_at(ach(a, magro(a)), t_1)$$
$$fazer\_dieta(A)\ causes\ magro(A)$$

Nesta situação, o sistema proposto irá seleccionar a acção de *fazer_dieta* como acção que leva ao estado desejado e irá criar o plano que consiste na intenção de executar essa acção.

No entanto, o processo abdutivo de geração de planos pode levar à criação de mais do que um plano que permita a satisfação dos objectivos existentes (diversas soluções abdutivas). Assim, é necessária a existência de um processo de selecção de planos (soluções abdutivas) que os hierarquize e permita a escolha de um deles. O processo de selecção é efectuado com base na definição de uma ordenação entre os predicados revisíveis. O processo de planeamento abdutivo será descrito em detalhe na secção 5.3.

Recorrendo a um exemplo apresentado no capítulo anterior, existe uma situação em que há dois planos passíveis de serem executados de modo a que um agente fale com outro: telefonar-lhe ou encontrar-se com ele.

$$telefonar(ana, catarina, casa(catarina)).$$
$$ir\_a(ana, casa(catarina)).$$

Supondo que a acção de telefonar tem um custo associado menor do que a acção de ir a um determinado local, então a solução preferencial será a de telefonar.

Finalmente, é necessário transformar os planos gerados, em actos propriamente ditos. No âmbito deste trabalho ir-se-ão apresentar essencialmente situações em que os actos gerados são actos de fala, embora o sistema proposto permita a geração de outro tipo de acções, que não somente os actos de fala, como é o caso do exemplo acima. De facto, o processo proposto para a geração de planos como conjunto de acções é independente do domínio dos diálogos e pode ser utilizado em qualquer outro domínio, desde que o processo de descrição das acções passíveis de serem executadas tenha sido correctamente efectuado. O processo de geração das acções baseia-se na transformação das intenções de realizar as acções abduzidas na concretização dessas acções, dando origem a eventos.

Como exemplo, temos que o plano de telefonar para um local l:

$$holds\_at(int(a, telefonar(a, b, l)), t_1).$$

será transformado na acção propriamente dita e o modelo do agente será actualizado com os seguintes factos (efectuando a actualização e a, eventual, revisão do seu estado mental):

$$happens(e, t_i, t_f).$$
$$act(e, telefonar(a, b, l)).$$

Note-se que a execução *(física)* da acção de *telefonar*, como de qualquer outra acção, requer uma outra abordagem, envolvendo o micro-planeamento da acção em si e não será objecto de estudo desta tese.

Na secção 5.4 será apresentado o processo de transformação de planos em acções (geração das acções).

Em conclusão, poder-se-á afirmar que, com base neste conjunto de processos e recorrendo à metodologia apresentada nos capítulos anteriores, é possível modelar um ambiente

de diálogos, em que os agentes têm capacidade de identificar parcialmente os planos subjacentes aos diversos actos de fala, inferem as diversas atitudes, geram os seus próprios planos e agem de modo a atingir os seus objectivos. Estas diversas componentes são integradas através do recurso a um módulo de gestão de diálogos, que será descrito na secção 5.5.

5.2 Identificação de planos

No âmbito deste trabalho, a identificação, por parte de um agente, dos planos dos seus interlocutores é efectuada através do reconhecimento do conjunto das intenções desses agentes de que determinadas acções venham a ser realizadas, de modo a que determinados objectivos sejam atingidos. Esta identificação é parcial, dado ser efectuada apenas com base nos actos de fala que foram produzidos no diálogo. Note-se que os actos de fala podem ter variadas interpretações. Aqui assume-se que o agente adopta uma das interpretaçõe e que, se o interlocutor estiver em desacordo, poderá manifestar essa discordância, desambiguando o acto de fala.

Em concreto, após a ocorrência de um evento, é iniciado o processo de cálculo do novo modelo bem fundado do programa em lógica que modela o estado mental do agente. Este processo levará à actualização e, eventual revisão das suas atitudes e a um processo de identificação dos planos dos interlocutores.

Definição 58 *Do ponto de vista do agente a, o plano do agente x num instante de tempo t pode ser representado por:*

$$P_a(x,t) = \{int(x,A) : holds\_at(bel(a, int(x,A)), t) \in PCFXSM(M_a(t))\}$$

onde $M_a(t)$ é o programa em lógica que representa o estado mental de a no instante de tempo t (após actualização e revisão) e A é uma acção.

De acordo com esta definição de plano, um agente efectua a suposição parcial de planos dos outros agentes a partir da inferência das suas crenças sobre as intenções atribuídas aos outros agentes (e cuja manifestação em contrário não foi subsequentemente realizada). Note-se que, de acordo com a definição de actos de fala apresentada no capítulo 3, as intenções dos agentes não são directamente manifestadas por eles, mas sim, uma consequência indirecta dos seus actos de fala (obtidas a partir do modelo do agente).

Como exemplo de uma aplicação deste conceito, temos a situação descrita no capítulo anterior em que um agente comunica que quer falar com outro agente:

$$happens(e_1, t_1, t_2).$$
$$act(e_1, inform(ana, susana, int(ana, falar(ana, catarina)))).$$

Conforme foi apresentado, o acto de informar leva à inferência de que o agente ao qual o acto é dirigido acredita que o emissor tem como intenção realizar a acção:

$$holds\_at(bel(susana, int(ana, falar(ana, catarina))), t_2).$$

Deste modo, um agente efectuou a identificação parcial do plano do outro agente, a partir dos seus actos de fala.

Note-se que a inferência de intenções também poderá ser efectuada a partir de actos de fala que não referem explicitamente intenções, mas sim acções, crenças ou informações. Para tal bastará que existam regras no modelo do agente que relacionem a informação transmitida com determinadas intenções.

Como exemplo, suponhamos a continuação do exemplo anterior, em que a Ana, além de informar que pretende falar com a Catarina, informa que acredita que ela está no hospital e que o número de telefone do hospital é o 999999:

$$happens(e_2, t_3, t_4).$$
$$act(e_2, inform(ana, susana, bel(ana, em(hospital, catarina)))).$$
$$act(e_2, inform(ana, susana, bel(ana, telefone(hospital, 999999)))).$$

Caso exista uma regra que relaciona as crenças,

$$\begin{aligned} holds\_at(bel(X, int(Y, telefonar(Y, L))), T) \leftarrow\ & holds\_at(bel(X, int(Y, falar(Y, Z))), T), \\ & holds\_at(bel(X, bel(Y, em(L, Z))), T), \\ & holds\_at(bel(X, bel(Y, telefone(L, N))), T_f). \end{aligned}$$

A intenção de telefonar para o hospital é inferida:

$$holds\_at(bel(susana, int(ana, telefonar(ana, hospital))), t_2).$$

5.3 Geração de Planos

O processo de geração dos planos de um agente necessita de uma metodologia distinta da utilizada para a identificação dos planos dos seus interlocutores, dado necessitar de abduzir as acções necessárias à satisfação dos objectivos do agente.

Assim, o planeamento das acções a realizar por um agente é efectuado a partir da conjugação de duas estratégias:

1. Inferência das intenções de realizar acções que pertencem ao modelo mental do agente ($M_a(t)$);
2. Abdução das acções que permitem satisfazer os objectivos do agente.

A primeira estratégia foi descrita na secção anterior e equivale a efectuar a identificação dos planos do agente ($Pl_a(a, t)$) no seu modelo ($M_a(t)$). Esta estratégia não basta para obter as acções necessárias à satisfação dos seus objectivos.

A segunda estratégia, tem como objectivo gerar as acções que possibilitem a satisfação dos objectivos do agente. Esta estratégia pode ser decomposta em quatro passos:

1. Inferência de objectivos;
2. Abdução de acções, para permitir a satisfação desses objectivos;
3. Selecção da solução abdutiva preferida;
4. Criação de intenções de realizar apenas as acções abduzidas preferidas.

No primeiro passo, o agente efectua a inferência dos seus objectivos, tendo em conta o seu estado mental e os actos de fala realizados. Na segunda e terceira fases, é efectuada a inferência abdutiva das acções que poderão levar à satisfação desses objectivos e é seleccionada a solução preferida. Finalmente, numa última fase, com base nas acções abduzidas preferidas são criadas intenções novas, visando a realização dessas acções.

Numa fase posterior, as acções planeadas deverão ser geradas. Este processo será objecto de análise na secção 5.4. Note-se que a geração das acções planeadas não garante que os objectivos sejam sempre satisfeitos pois poderão existir situações de interferência entre acções e pode haver interferência com eventos externos (gerados pelos outros agentes) não previstos no processo de planeamento. No âmbito deste trabalho não serão abordadas estes problemas, assumindo-se a sua possível resolução através do recurso a um módulo especializado de planeamento.

A gestão deste processo de planeamento é efectuada através de uma arquitectura de gestão de diálogos, definida por regras de programação em lógica, e que será apresentada na secção 5.5.

A título ilustrativo, suponhamos o exemplo apresentado anteriormente, em que existe uma situação onde o agente a tem o objectivo de atingir um estado de *magro(a)* e em que uma acção possível para provocar esse estado é fazer dieta,

$$holds\_at(ach(a, magro(a)), t_1).$$
$$fazer\_dieta(A)\ causes\ magro(A).$$

O processo de planeamento de acções contemplará a abdução das acções que poderão levar a esse objectivo (processo analisado na sub-secção 5.3.2):

$$happens(e, t_i, t_f).$$
$$t_1 < t_i < t_f.$$
$$act(e, fazer\_dieta(a)).$$

O passo seguinte corresponderá à criação da intenção de que a referida acção venha a ser executada:

$$holds\_at(int(a, fazer\_dieta(a)), t_1).$$

Ou seja, o plano de a no instante de tempo t_1 será:

$$P_a(a, t_1) = \{int(a, fazer\_dieta(a))\}.$$

representando a intenção de que a acção de fazer dieta seja realizada. Caso esta intenção fosse contraditória com intenções anteriores (situação a analisar em 5.3.2), ter-se-ia de preferir uma outra solução abdutiva de acções.

Finalmente, a acção será executada de acordo com a abdução efectuada, visando satisfazer os objectivos existentes.

As quatro componentes que foram identificadas como constituintes do processo de geração de planos (inferência de objectivos, abdução de acções, selecção de soluções e criação de intenções) serão objecto de análise detalhada nas sub-secções seguintes.

5.3.1 Inferência de Objectivos

A inferência de objectivos é a primeira fase do processo global de geração de planos. De facto, é com base nos objectivos de um agente, nas situações que ele pretende que venham a verificar-se, que é efectuado o processo de planeamento das acções que poderão levar à satisfação desses objectivos.

Do ponto de vista de um agente a, os seus objectivos, num dado instante de tempo t, são representados assim:

Definição 59 *Seja $O_a(t)$ o conjunto dos objectivos do agente a, no instante de tempo t.*

$$O_a(t) = \{P : holds\_at(bel(a, ach(a, P)), t) \in PCFXSM(M_a(t))\}$$

em que P é uma propriedade que o agente a deseja que venha a ser verdadeira e $M_a(t)$ é o programa em lógica que modela o agente a.

As crenças sobre os objectivos de um agente são adoptadas através do recurso a dois métodos distintos:

1. Directamente a partir das acções, nomeadamente, dos diversos actos de fala;
2. Através de transferência de objectivos, utilizando as regras que definem a cooperatividade dos agentes.

Em relação ao primeiro método, a realização de uma acção pode provocar o início de um determinado objectivo (por exemplo, o objectivo de ser magro pode ser iniciado pela leitura de um livro que diz que os magros têm uma esperança de vida maior). Analisando a definição dos diversos actos de fala verifica-se que somente os actos $checkif(s, h, p)$ e $askif(s, h, p)$ têm como consequência a criação de crenças no receptor sobre os objectivos do emissor (que através da transferência de objectivos poderão passar a ser objectivos do receptor).

Note-se que, embora o efeito nos receptores de actos de fala *checkif* e *askif* seja idêntico (em ambas as situações adoptam a crença de que o emissor deseja saber se uma dada propriedade é válida), as suas pré-condições são diferentes: em *checkif* o emissor acredita

	receptor
askif(s,h,p)	bel(h,ach(s,knowif(s,p)))
checkif(s,h,p)	bel(h,ach(s,knowif(s,p)))

Tabela 5.1: Objectivos consequência dos actos de fala no receptor

que sabe o valor da proposição e pretende confirmá-lo, em *askif* o emissor pretende saber o valor da proposição.

As outras acções não têm efeito directo sobre a criação de objectivos nos diversos agentes e podem, somente, criar crenças sobre as crenças dos emissores. Por exemplo, um acto de informar sobre um determinado objectivo — $inform(a, b, ach(a, p))$ — cria no receptor um crença sobre a crença do emissor: $bel(b, bel(a, ach(a, p)))$. Utilizando as regras de transferência de crenças poder-se-ão criar, indirectamente, objectivos no receptor: $bel(b, ach(a, p))$.

De acordo com o segundo método, os objectivos dos agentes também são inferidos através da transferência de crenças sobre objectivos descrita pelas regras de cooperatividade. Assim, um agente assume como seus os objectivos dos outros agentes consoante o seu grau de cooperatividade: um agente totalmente cooperativo assume todos os objectivos que acredita que os outros agentes possuem; um agente menos cooperativo assume somente os objectivos que não contrariem os seus; um agente racionalmente cooperativo assume, apenas, os objectivos que sejam plausíveis no seu modelo, i.e. existe um conjunto de acções que permitem satisfazer esses objectivos. Note-se que a transferência de objectivos implica que o agente que aceita esses objectivos os adopte como seus e efectue as acções que considera necessários à sua satisfação.

Como exemplo, suponhamos o exemplo em que existe uma situação de obesidade:

A: Queria ficar magro!

Esta frase tem como correspondente acto de fala:

$$happens(e, t_1, t_1).$$
$$act(e, inform(a, b, ach(a, magro(a)))).$$

Recorrendo à definição de $inform$ (regra 3.29):

$$holds\_at(bel(b, bel(a, ach(a, magro(a)))), t_1).$$

Se suposermos que o agente b é ingénuo e totalmente cooperativo, então ele acredita no que lhe é transmitido e assume como seus os objectivos dos outros:

Pela regra de transferência de crenças (3.17):

$$holds\_at(bel(b, ach(a, magro(a))), t_1).$$

Pela regra de transferência de objectivos (3.52):

$$holds\_at(bel(b, ach(b, magro(a)), t_1).$$

Ou seja,

$$magro(a) \in O_b(t_1)$$

isto é, houve transferência de objectivos entre os dois agentes.

Em conclusão, pode-se afirmar que, utilizando estes dois métodos, é possível efectuar a inferência de objectivos de um agente, passo fundamental para o processo de abdução das acções que permitirão a sua satisfação.

5.3.2 Abdução de Acções

O segundo passo na metodologia de geração de planos é a abdução das acções necessárias à satisfação dos objectivos dos agentes.

Este processo pode ser decomposto em quatro fases:

1. Identificação dos objectivos;
2. Transformação desses objectivos em restrições de integridade hipoteticamente consideradas;
3. Abdução das acções que suportam as novas restrições;
4. Escolha da solução abdutiva preferida.

O primeiro passo foi descrito na sub-secção anterior e consiste na identificação dos objectivos do próprio agente num dado instante de tempo sendo representado por $O_a(t_1)$. Este conjunto contém os objectivos suportados pelo modelo do agente a no instante de tempo t_1, isto é, são as propriedades

$$holds\_at(bel(a, ach(a, p)), t_1)$$

que pertencem ao $PCFXSM(M_a(t_1))$, onde $M_a(t_1)$ é o programa em lógica que modela o agente a, no instante de tempo t_1 após o processo de actualização e revisão descrito no capítulo anterior. Note-se que a existência de objectivos contraditórios é evitada devido à definição de restrições de integridade adicionais que obrigam à revisão do modelo do agente e à eliminação dos objectivos contraditórios. Nomeadamente, para qualquer restrição de integridade

$$\Leftarrow holds\_at(P, T), holds\_at(Q, T)$$

deverá existir a restrição:

$$\Leftarrow holds\_at(bel(X, ach(X, P)), T), holds\_at(bel(X, ach(X, G)), T).$$

Em concreto, para cada proposição P deverá existir a restrição:

$$\Leftarrow holds\_at(bel(X, ach(X, P)), T), holds\_at(bel(X, ach(X, \neg P)), T).$$

O segundo passo consiste na transformação destes objectivos em restrições de integridade sobre as propriedades que se deseja que se verifiquem num tempo posterior ao presente. Esta fase pretende modelar o seguinte tipo de raciocínio hipotético: *para os objectivos serem satisfeitos, quais as acções necessárias?*

Para a concretização desta fase é necessário efectuar o seguinte procedimento:

Definição 60 *Seja $M_a(t)$ o modelo do agente a no instante t e $IC_a(t)$ as restrições de integridade desse modelo. O novo modelo hipotético $M'_a(t)$ é obtido do anterior, sendo as novas restrições de integridade $IC'_a(t)$:*

$$IC'_a(t) = IC_a(t) \cup \{holds\_at(p, t_\infty) \Leftarrow : p \in O_a(t) \wedge t < t_\infty\}$$

em que t_∞ representa um posterior ao instante de tempo actual, t (i.e. é uma constante de Skolem resultante de uma quantificação existencial).

Este procedimento indica que, para cada objectivo de um agente num dado instante de tempo, é efectuada a asserção de uma restrição de integridade que força a que o objectivo se verifique, num instante de tempo futuro.

Será com base nestas restrições que o sistema proposto irá efectuar a abdução das acções necessárias à sua concretização e a obtenção do valor concreto de t_∞, que deverá ser o menor futuro possível (o controlo deste processo abdutivo é descrito na secção 5.5). Este processo abdutivo poderá demonstrar a não exequibilidade dos objectivos existentes, caso o modelo seja contraditório e não exista uma revisão que elimine essa contradição.

De facto, se $M'_a(t)$ for contraditório e se, além disso, não for possível hipoteticamente remover essa contradição, então não é possível efectuar o planeamento de acções que permitam satisfazer os objectivos existentes. Nesta situação, os objectivos manter-se-ão activos, embora não sejam geradas de imediato acções que os venham a satisfazer. No entanto, eventos futuros não dependentes da intervenção do agente, poderão vir a alterar esta situação, criando as condições necessárias à abdução de acções que satisfaçam os objectivos existentes. Estes eventos, ao dependerem da intervenção de outros agentes, não poderiam ser abduzidos durante o processo de planeamento (só é possível abduzir acções pelas quais o agente seja o responsável).

Recorrendo ao exemplo apresentado na subsecção anterior, temos que:

$$magro(a) \in O_b(t_1)$$

Pelo que a restrição de integridade seguinte é criada no modelo hipotético:

$$holds\_at(magro(a), t_\infty) \Leftarrow .$$

O passo seguinte desta fase é a abdução das acções que permitam a concretização dos objectivos do agente. Para a concretização desta fase é necessário obter o $PCFXSM$ correspondente ao modelo do agente acrescido das restrições de integridade deduzidas no passo anterior, i.e.:

$$PCFXSM(M_b'(t)), onde IC_b'(t) = IC_b(t) \cup \{holds\_at(p, t_\infty) \Leftarrow : p \in O_b(t)\}$$

De notar que as novas restrições de integridade, tal como as anteriores, são vistas como teoremas que têm de ser demonstrados, i.e., a restrição

$$IC \Leftarrow$$

é codificada como:

$$\bot \leftarrow not\ IC$$

Deste modo, o processo de remoção de contradições utilizado permite rever o programa de modo a que IC seja válido. Na remoção a 2 valores não é possível garantir uma única solução e, em geral, existem várias soluções que deverão ser sujeitas a um processo adicional de escolha (ver secção seguinte).

É de realçar ainda que, de acordo com o definido no capítulo 2, os predicados *happens* e *act* são abdutíveis, pelo que as restrições de integridade poderão ser satisfeitas através da abdução de novos eventos e de acções associadas. As restrições de integridade apresentadas no capítulo 2 garantem que os eventos abduzidos têm sempre acções associadas, que não existem acções sem eventos e que um mesmo evento não pode ocorrer em instantes de tempo distintos.

As acções abduzidas ($Ab_a(t)$) correspondem ao planeamento efectuado, de modo a atingir os objectivos requeridos, e podem ser identificadas pela seguinte regra:

Definição 61

$$\begin{aligned} Ab_a(t) = \{&happens(e, t_{Ai}, t_{Af}), act(e, A) : \\ &happens(e, t_{Ai}, t_{Af}), act(e, A) \in PCFXSM(M_a'(t)) \wedge \\ &happens(e, t_{Ai}, t_{Af}) \notin PCFXSM(M_a(t)) \wedge \\ &act(e, A) \notin PCFXSM(M_a(t))\} \end{aligned}$$

Isto é, as acções e seus acontecimentos pertencem ao modelo obtido para o planeamento e não pertencem ao modelo da conversação em que o planeamento é iniciado: são acções hipotéticas. Estas acções estão identificadas pelo seu tempo de início e fim.

No exemplo apresentado, a propriedade a satisfazer:

$$holds\_at(magro(a), t_\infty) \Leftarrow .$$

permite a abdução da acção para a sua concretização:

$$\begin{aligned} &happens(e_1, t_i, t_f). \\ &act(e_1, fazer\_dieta(a)). \\ &t_1 < t_i < t_f \leq t_\infty. \end{aligned}$$

(assumindo a acção fazer dieta definida como: $fazer\_dieta(X)\ causes\ magro(X)$).

Nesta situação temos que:

$$Ab_b(t_1) = \{happens(e_1, t_i, t_f), act(e_1, fazer\_dieta(a))\}.$$

No capítulo 6 serão apresentados exemplos de situações mais complexas, em que é necessário encadear no tempo a realização de algumas acções.

No entanto, é possível que exista mais do que uma solução abdutiva. Nesta situação é necessário escolher uma solução preferencial. Esta é a quarta fase do processo de abdução de acções e será analisada na subsecção seguinte.

5.3.3 Selecção de soluções

O processo de selecção de soluções pode ser efectuado através de duas estratégias distintas:

1. Hierarquização das acções, estipulada na definição do modelo do agente;
2. Hierarquização das soluções abdutivas.

A primeira estratégia, descrita na sub-secção seguinte, actua antes do processo de planeamento, impedindo que sejam geradas várias soluções abdutivas. Esta estratégia não selecciona soluções, apenas impede que elas sejam geradas.

A segunda estratégia pretende hierarquizar as várias soluções abdutivas, com base numa definição de prefereências entre as diersas soluções abdutivas.

Hierarquização de Acções

O processo de hierarquização de acções pretende impedir que certas soluções abdutivas sejam geradas, quando são possíveis outras. Para tal, são definidas regras que impedem determinadas soluções de serem abduzidas. O objectivo é definir uma hierarquia das acções que elimine algumas soluções possíveis.

Como exemplo, suponhamos a situação apresentada anteriormente em que um agente tem como objectivo ficar magro. No contexto deste exemplo, existem duas acções possíveis de executar para satisfazer esse objectivo: *fazer dieta* e *fazer exercício*.

$$holds\_at(bel(a, ach(a, magro(a))), t_1)$$
$$fazer\_dieta(A)\ causes\ magro(A)$$
$$fazer\_exercício(A)\ causes\ magro(A)$$

De acordo com o processo de planeamento abdutivo descrito na secção anterior, existiriam duas soluções possíveis para satisfazer o objectivo:

1. $happens(e_1, t_i, t_f), act(e_1, fazer\_dieta(a))$

2. $happens(e_1, t_i, t_f), act(e_1, fazer\_exercício(a))$

Suponhamos que se pretende definir que a acção de *fazer exercício* deve ser prioritária em relação à acção de *fazer dieta*. Utilizando a estratégia de hierarquização de acções, a descrição de acções deveria ser alterada para:

$$fazer\_dieta(A)\ causes\ magro(A)\ if$$
$$not\ possible\_action(fazer\_exercicio(A))$$
$$fazer\_exercício(A)\ causes\ magro(A)$$

O predicado $possible\_action(Action)$ verifica se a acção $Action$ pode ser executada, ou seja, se as suas pré-condições estão ou podem ser satisfeitas abdutivamente.

A ideia do processo de hierarquização é definir que uma acção menos prioritária só poderá ser executada se a acção mais prioritária não tiver os seus pré-requisitos satisfeitos.

O processo pode ser definido de um modo geral para qualquer par de acções que tenham efeitos semelhantes, do seguinte modo:

Definição 62 *Sejam A_1 e A_2 duas acções do domínio, tais que se pretende que A_2 seja prioritária em relação a A_1. Isto é, a intenção de realizar A_2 deverá sobrepôr-se à intenção de realizar A_1. Sejam r_1 e r_2 as regras que inicialmente descrevem as acções A_1 e A_2:*

$$r_1 : A_1\ causes\ F\ if\ P_1, ..., P_n$$
$$r_2 : A_2\ causes\ F\ if\ P_1', ..., P_m'$$

As regras devem ser modificadas do seguinte modo:

$$r_1 : A_1\ causes\ F\ if\ P_1, ..., P_n, not\ possible\_action(A_2)$$
$$r_2 : A_2\ causes\ F\ if\ P_1', ..., P_m'$$

O predicado possible_action/1 é definido do seguinte modo:

$$possible\_action(A_2)\ if\ P_1', ..., P_m'$$

Assume-se o processo de tradução para regras de programação em lógica apresentado no capítulo 2.

Este processo garante que a acção A_1 só será escolhida se a acção A_2 não o puder ser.

Hierarquização de soluções abdutivas

Ao contrário da estratégia anterior, que actua eliminando possíveis soluções abdutivas, esta estratégia hierarquiza as soluções existentes, de modo a optar por um modelo abdutivo preferencial.

A hierarquização de soluções é obtida através da definição de preferências globais , i. e. preferências sobre a ordem das revisões, sendo o processo de definição de preferências globais efectuado de um modo idêntico ao proposto no sistema REVISE ([DNP94]). Neste sistema as preferência globais são definidas através de um grafo directo acíclico e/ou[1] por regras do tipo:

$$Level_0 << Level_1 \wedge ... \wedge Level_n \ (n \geq 1)$$

onde os nós $Level_i$ do grafo são identificadores de nível de preferência. A cada nó nível de preferência está associado um conjunto de revisíveis — $\mathcal{R}(Level_i)$. A raiz do grafo de preferências é o nó denominado *bottom*.

Definição 63 *Seja P um programa em lógica estendida e Π um grafo de preferências contendo o nível Lev. A revisão R é a preferida sse R é uma revisão minimal usando os revisíveis $\mathcal{R}(Lev)$ e existe uma árvore-e $\mathcal{T}$ em Π, com raiz Lev, tal que todas as folhas de $\mathcal{T}$ são bottom e nenhum outro nível de preferência em $\mathcal{T}$ tem revisões.*

Schroeder et al. [SDP96], no seu trabalho, propõem um algoritmo para a computação da revisão preferida de um programa em lógica estendida.

Voltando ao exemplo apresentado na secção anterior, supondo que existe uma definição de preferências pré-definida entre as soluções que possuem acções de *fazer exercício* e de *fazer dieta*:

$$\begin{aligned} 1 &< < bottom \\ \mathcal{R}(bottom) &= \{act(E, fazer\_dieta(A))\} \\ \mathcal{R}(1) &= \{act(E, fazer\_exercicio(A))\} \end{aligned}$$

Nestas condições o modelo preferencial será o que contém a acção de fazer dieta.

5.3.4 Criação de Intenções

A última fase do processo de geração de planos de um agente é a da criação no modelo não hipotético do agente das intenções de realizar as acções que foram abduzidas $Ab_a(t)$ (no modelo hipotético preferido). Esta fase permite a representação no modelo do agente do resultado do planeamento efectuado: as intenções de realizar as acções abduzidas preferidas.

A criação de intenções pode ser descrita através do seguinte procedimento que actualiza o modelo do agente com o evento de planear e inicia um novo conjunto de intenções, no instante de tempo t:

Definição 64 *Seja e_t um novo evento e $planear(a)$ a acção que representa o acto do agente A planear as acções que deseja que sejam realizadas.*

[1] labeled directed acyclic and/or graph

Seja $Pl_a(t)$ o conjunto de regras de programação em lógica que descreve o evento de planear acções no instante de tempo t, bem como os seus efeitos (início de determinadas intenções). Este conjunto é definido por:

$$\begin{aligned} Pl_a(t) &= \{happens(e_t, t, t),\ act(e_t, planear(a)\} \cup \\ \{initiates(e_t, t, int(a, A)) &: act(E, A) \in Ab_a(t)\} \end{aligned}$$

Note-se que o predicado initiates/3 define que a intenção foi iniciada no instante de tempo t e é válida em instantes de tempo superiores ($holds\_at(int(a, A), t'), t \leq t'$).

Definição 65 *O novo modelo mental do agente a, $M''_a(t)$, que inclui as intenções de que as acções necessárias à satisfação dos seus objectivos sejam realizadas é dado por:*

$$M''_a(t) = M_a(t) \cup Pl_a(t).$$

Note-se que a reunião é efectuada com o modelo anterior ($M_a(t)$) e não com o modelo hipotético ($M'_a(t)$), devido ao facto de que o modelo hipotético conter como restrições de integridade os objectivos do agente, conduzindo às várias soluções abdutivas de acções e não apenas à escolhida.

No entanto, a introdução de novas atitudes poderá introduzir também contradições no modelo do agente. De facto, embora o conjunto de acções abduzidas resulte do $PCFXSM$ do programa em lógica, não sendo, portanto, contraditório, as intenções de realizar essas acções poderão ser contraditórias com intenções anteriores.

Como exemplo, e recorrendo ao exemplo anterior em que a acção abduzida é:

$$Ab_b(t_1) = \{happens(e_1, t_i, t_f), act(e_1, fazer\_dieta(a))\}.$$

Suponhamos que existe uma intenção anterior de que o agente a não faça dieta, devido a um outro motivo:

$$holds\_at(int(b, \neg fazer\_dieta(a)), t_1) \in PCFXSM(M_b(t_1))$$

Existe, ainda, uma restrição de integridade que relaciona duas quaisquer intenções contrárias:

$$\Leftarrow holds\_at(int(X, L), T), holds\_at(int(X, \neg L), T).$$

significando que é contraditório, num dado instante de tempo, um agente ter uma intenção e a sua contrária. O agente deverá optar por uma das intenções.

Voltando ao exemplo apresentado, teríamos que:

$$\begin{aligned} &holds\_at(int(b, fazer\_dieta(a)), t_1) \\ &holds\_at(int(b, \neg fazer\_dieta(a)), t_1) \end{aligned}$$

e o estado mental do agente b seria contraditório (de acordo com a restrição de integridade apresentada).

Nestas condições, o processo de revisão permite a definição de preferências de modo a que a opção seja pelo modelo que revê as atitudes mais antigas (ou as mais recentes). Assumindo que se dá preferência aos modelos que mantêm as intenções mais recentes, a intenção

$$int(b, \neg fazer\_dieta(a))$$

é terminada, e mantém-se a intenção de

$$holds\_at(int(b, fazer\_dieta(a)), t_1)$$

Se assumíssemos a opção contrária, a intenção que prevaleceria seria a de não fazer dieta. De notar que, neste caso, a eliminação da intenção de vir a fazer dieta, elimina também a possibilidade de atingir o objectivo desejado: $ach(b, magro(a))$. No entanto, o objectivo mantém-se activo e poderá vir a ser satisfeito como resultado de um processo posterior de planeamento (por exemplo, quando a acção de fazer dieta tiver terminado)

Em suma, após qualquer acto de fala que termine no instante de tempo t, é efectuado um processo de inferência que contempla três etapas:

1. Cálculo das novas atitudes do agente;
2. Abdução das acções preferidas que permitem satisfazer os objectivos existentes: $Ab_a(t)$
3. Cálculo do novo modelo do agente que inclua a intenção de realizar o plano abduzido: $M''_a(t)$.

É de salientar que este processo implica o cálculo de três modelos : o correspondente ao programa em lógica após os eventos, o correspondente ao modelo hipotético (para abdução e selecção das acções preferidas) e o correspondente ao programa em lógica acrescido das novas intenções inferidas e dos novos eventos de planeamento (revendo e preferindo determinadas intenções, se necessário). Estas questões serão abordadas em maior detalhe no capítulo 7.

O processo de geração, a partir do conjunto de acções abduzidas, será abordado na próxima secção.

5.4 Geração de Actos de Fala

No âmbito deste trabalho será analisado, fundamentalmente, o processo de geração de actos de fala, e não o processo mais geral de geração de acções. Note-se que, quando se refere geração de actos de fala não se está a considerar a geração das frases em Língua Natural correspondentes, mas somente a problemática da geração do acto de fala em si.

De facto, a geração de actos de fala tem algumas características que permitem simplificar o processo de geração:

1. Os pré-requisitos de um acto de fala são conjuntos de atitudes do agente (crenças e intenções). Deste modo, a geração de actos de fala pode ser efectuada de um modo independente de factores externos aos modelo mental do agente. Note-se que, no entanto, os factores externos podem influenciar o processo de geração, através da criação de atitudes no modelo do agente.

2. Os actos de fala têm como consequência a criação de novas atitudes no modelo dos agentes. Estas novas atitudes poderão vir a desencadear processos de planeamento e geração de novas acções.

3. Podem ser considerados como acções não decomponíveis. Para efeitos de geração, os actos de fala poderão ser considerados como não decomponíveis, atómicos. De facto, o processo de geração dos actos de fala a partir das intenções não necessita de analisar o conteúdo desses actos; é suficiente a identificação do acto de fala em si. É de realçar que, no entanto, o processo de geração das frases a partir dos actos de fala deverá ter em conta os diversos constituintes da acção: o receptor, a acção em si, a informação veículada pela acção, etc. Além disso, dever-se-á ter em conta a possibilidade de gerar diversos actos de fala associados à mesma frase. Este processo de geração de frases em linguagem natural, não é objecto deste trabalho.

4. A geração pode ser considerada instantânea. Os actos de fala podem ser simplificados e considerados de execução num dado instante de tempo. De facto, embora a execução dos actos de fala possa não ser instantânea em termos *físicos*, dado que o processo de reconhecimento e geração de atitudes não necessita de decompor esses actos em sub-acções, não existe necessidade de considerar tempos não instantâneos para os referidos actos de fala. É de salientar que esta é uma característica que não é imposta pela metodologia proposta, é somente um factor de simplificação. A metodologia proposta ao longo desta tese não restringe as acções a serem executados num intervalo de tempo pontual.

O processo de geração de acções, sejam ou não actos de fala, pode ser dividido em duas fases:

1. Geração das acções seleccionadas abduzidas pelo processo de planeamento;

2. Geração de acções não planeadas, mas para as quais existe uma intenção de as realizar (por exemplo, resultado de regras de transferência de intenções)

A primeira fase corresponde à execução do plano abduzido e é efectuada através da actualização do modelo do agente com as acções preferidas:

$$M_a'''(t) = M_a''(t) \cup Ab_a(t).$$

Note-se que o modelo obtido contém as acções abduzidas durante o processo de planeamento ($happens/3$ e $act/2$).

A segunda fase, geração de acções não abduzidas, mas que possuem uma intenção explícita de serem realizadas, pode ser decomposta em duas fases:

1. Verificação dos pré-requisitos;

2. Execução da acção.

A primeira fase pretende verificar se a acção tem os seus pré-requisitos satisfeitos, isto é, se ela pode ser executada:

Definição 66 *Seja $a_t = int(a, x)$ a intenção do agente a em realizar uma acção no instante de tempo t, e o evento de executar essa acção $act(e, x)$ e $M_a(t)$ o modelo do agente a nesse instante de tempo. A acção x pode ser executada no instante de tempo t sse*

$$PCFXSM(M_a(t)) \models enabled(e, t)$$

Esta definição significa que uma acção pode ser executada se o $PCFXSM$ do programa em lógica, correspondente ao estado mental do agente, acrescido da instância dessa acção, suporta a derivação das suas pré-condições de execução. Relembre-se que o predicado $enabled(e, t)$ é verdadeiro quando o evento e pode ocorrer (ou iniciar-se) no instante de tempo indicado.

O passo seguinte, caso a acção possa ser executada, é a sua transformação em acção propriamente dita, o que nos casos dos actos de fala significa uma frase em Língua Natural. Conforme foi referido anteriormente, esta é uma área não abordada no âmbito deste trabalho e será identificada somente pela execução do acto em si:

Definição 67 *Seja $A_a(t) = \{a_{1,\ldots}, a_n\}$ o conjunto de intenções do agente a no instante de tempo t. Sejam $a_i = int(a, \alpha_i)$ as intenções tal que:*

1. $a_i \in A_a(t)$

2. *α_i pode ser executada.*

A execução das acções α_i, representada por $do(\alpha_i, t)$, tem como efeito a actualização do modelo do agente a com os seguintes factos:

(a) $happens(e, t, t)$.

(b) $act(e, \alpha_i)$.

(c) $terminates(e, t, int(a, \alpha_i))$.

A acção deverá estar em condições de ser executada, tendo os seus pré-requisitos satisfeitos e, a sua execução, implicará a actualização correspondente do estado mental dos agentes (incluindo o término da intenção de que a acção seja realizada).

De notar que a definição apresentada permite a execução simultânea de acções.

Como exemplo, suponhamos que, no instante de tempo t, um agente possui intenção de informar outro sobre uma propriedade p e de pedir que uma acção α seja realizada:

$$holds\_at(int(a, inform(a, b, p)), t)$$
$$holds\_at(int(a, request(a, b, \alpha)), t)$$

Suponhamos, ainda, que estas intenções foram resultado do processo de cálculo do modelo do agente e não do processo de planeamento. Admitindo que as duas acções têm as suas pré-condições satisfeitas, temos que:

$$happens(e, t, t).$$
$$act(e, inform(a, b, p)).$$
$$act(e, request(a, b, \alpha)).$$
$$terminates(e, t, int(a, inform(a, b, p)).$$
$$terminates(e, t, int(a, request(a, b, \alpha)).$$

Ou seja, as acções foram executadas simultaneamente e as intenções canceladas.

5.5 Gestão de diálogos

A metodologia apresentada ao longo deste capítulo permite não só efectuar a identificação parcial das atitudes dos agentes interlocutores mas, também, efectuar a geração de planos próprios. Conforme foi referido anteriormente, este processo de participação em diálogos necessita, no entanto, de ser gerido por uma arquitectura de gestão de interacções que integre as diversas componentes apresentadas (reconhecimento de eventos, actualização e revisão, planeamento e geração).

Nesta secção será proposto um conjunto de procedimentos (por questões de simplicidade de representação, serão descritos através da linguagem de programação em lógica Prolog), que têm como objectivo gerir o processo de participação em diálogos, de acordo com a metodologia que foi apresentada anteriormente. Não é objectivo deste trabalho apresentar uma arquitectura geral de gestão de diálogos que tenha em consideração outro tipo de questões como, por exemplo, a representação da estrutura do discurso e do tópico e foco.

O processo de participação em diálogos é gerido por um procedimento *gestao* que dado o modelo M_i de um agente antes do início do diálogo, obtém o modelo após o final desse diálogo M_f:

$$gestao(M_i, M_f) \quad :- \quad repeat,$$

$$
\begin{aligned}
&obter\_evento(E), !,\\
&act\_rever(M_i, E, M),\\
&planear(M, M'', Abd),\\
&gerar(M'', Abd, M'''),\\
&gestao(M''', M_f).
\end{aligned}
$$

Este procedimento de gestão de diálogos pode ser descrito da seguinte forma:

1. Obter um novo evento ($obter\_evento(E)$);
2. Fazer a actualização e, eventual revisão do modelo inicial com o novo evento;
3. Planear as acções a realizar, abduzindo as acções necessárias e obtendo o novo modelo;
4. Gerar as acções abduzidas planeadas (e as não abduzidas, mas que possuam intenções);
5. Esperar novos eventos.

O predicado $obter\_evento(E)$ deverá interagir com módulos autónomos de reconhecimento de eventos, quer de frases em Língua Natural, quer de outro tipo de eventos. Esses módulos deverão reconhecer as acções envolvidas e os seus tempos de execução. Conforme foi referido, no âmbito deste trabalho, não se analisou a construção destes módulos, assumindo-se que o gestor de diálogos obteria o seu *output*.

O predicado $act\_rever(M_i, E, M)$ obtém o modelo M após a actualização do modelo inicial M_i com os novos eventos E:

$$
\begin{aligned}
act\_rever(M_i, E, M) \ :- \ & actualizar(M_i, E, M_1),\\
& rever(M_1, M).
\end{aligned}
$$

Onde $actualizar/3$ é o procedimento de actualizar modelos, descrito no capítulo 4, e $rever/2$ é o procedimento de remoção de contradições definido (ver capítulo 2 e 4).

O predicado $planear(M, M'', Abd)$ obtém a solução preferida para o planeamento, Abd, e o novo modelo (incluindo as intenções de executar as acções abduzidas):

$$
\begin{aligned}
planear(M, M'', Abd) \ :- \ & obter\_objectivos(M, Obj),\\
& juntar\_restricoes(M, Obj, M'),\\
& setof(S_i, solution(M', S_i), L_S),\\
& obter\_preferencia(L_S, Abd),\\
& criar\_intencoes(M, Abd, M'').
\end{aligned}
$$

O predicado $obter\_objectivos(M, Obj)$ obtém os objectivos Obj do agente (equivale a utilizar os procedimentos de prova definidos para a WFSX); $juntar\_restricoes/3$ adiciona ao

modelo M as restrições de integridade associadas aos objectivos do agente. O predicado $solution(M', S_i)$ obtém as soluções abduzidas para o modelo M' e L_S representa a lista de soluções abduzidas. $obter\_preferencia(L, Abd)$ obtém a solução abduzida preferida (de acordo com a definição apresentada na secção 5.3.3). Finalmente, em $criar\_intencoes/3$, M'' é o modelo actualizado com as intenções de realizar as acções abduzidas Abd.

O predicado $gerar(M'', Abd, M''')$ obtém o modelo M''' após o processo de geração definido sobre o modelo M'' com a solução abduzida Abd:

$$\begin{aligned} gerar(M'', Abd, M''') \ :- \ & act\_rever(M'', Abd, M_1), \\ & obter\_intencoes(M_1, Int), \\ & gerar\_act(M_1, Int, M'''). \end{aligned}$$

Após serem geradas as acções abduzidas e ser obtido o novo modelo M_1, são calculadas as intenções ainda existentes e são geradas as acções que têm as pré-condições satisfeitas (ver secção anterior).

Os procedimentos apresentados permitem efectuar a gestão dos diversos componentes necessários a uma participação em diálogos: desde o reconhecimento de eventos até à sua geração. No capítulo 7 serão abordadas em maior detalhe as questões relacionadas com a construção de um protótipo e com a obtenção de resultados experimentais.

5.6 Exemplo: Sistemas Operativos

Neste capítulo será apresentado um exemplo típico de diálogo em que é necessário efectuar a geração e a selecção de planos para satisfazer os objectivos dos agentes.

O diálogo passa-se num ambiente de interacção relacionado com sistemas operativos (ver [WC88] para um sistema semelhante).

Para efeitos deste exemplo, vamos supor a existência de apenas duas operações:

1. rm <f>; que apaga o ficheiro <f>;
2. mv <f> <f.bak>; que apaga o ficheiro <f> mantendo uma cópia.

Um ficheiro poderá somente ter duas propriedades:

1. apagado(f); significa que o ficheiro foi apagado;
2. recuperavel(f); significa que o ficheiro pode ser recuperado.

A descrição das acções, bastante simplificada, será:

$$\begin{aligned} rm \ &< \ f > causes \ apagado(f), \neg recuperavel(f) \\ mv \ &< \ f >< f.bak > causes \ apagado(f), \ recuperavel(f) \end{aligned}$$

Estas regras significam que uma acção (rm) apaga um ficheiro de um modo irrecuperável, enquanto a outra acção (mv) o apaga, podendo, no entanto, ser recuperado.

Suponhamos, agora, a seguinte frase do agente a para o agente b:

A: Quero que o ficheiro <f> seja apagado.

Esta frase poderá ter como correspondente acto de fala:

$$happens(e, t, t).$$
$$act(e, inform(a, b, ach(a, apagado(f)))).$$

Com base nestas regras, e assumindo um modelo de agente b ingénuo e cooperativo e de a verdadeiro, temos que:

$$holds\_at(ach(b, apagado(f)), t) \in PCFXSM(M_b(t))$$

a partir das regras do acto de *inform* (3.29) e das regras de transferência de crenças e objectivos (3.16, 3.17 e 3.52);

O processo de abdução de acções irá criar a nova restrição de integridade:

$$holds\_at(apagado(f), t_\infty) \Leftarrow .$$

Com base neste facto é efectuada a abdução da acção que possibilita a satisfação destes objectivos. Como existem duas acções possíveis, existirão dois modelos abdutivos. Se assumirmos que o custo da acção rm é inferior à da acção mv então o modelo do agente será actualizado com a seguinte solução preferida:

$$happens(e_1, t_1, t_1).$$
$$act(e_1, rm < f >).$$
$$t < t_1.$$

Neste exemplo, está-se a assumir que o agente iria executar a acção. No entanto, seria também possível considerar que a acção $rm < f >$ não significa apagar o ficheiro mas informar o outro agente sobre este comando (imprimindo-o no ecrã do computador, por exemplo).

Suponhamos que, no entanto, existe uma segunda frase na sequência da primeira:

A: Quero que o ficheiro <f> seja apagado. Mas quero que seja recuperável!

Neste caso, teríamos mais um acto de fala:

$$happens(e_2, t, t).$$
$$act(e_2, inform(a, b, ach(a, recuperavel(f)))).$$

Com base nestes novos factos, teríamos que,

$$\begin{aligned} holds\_at(ach(b, apagado(f)), t) &\in PCFXSM(M_b(t)) \\ holds\_at(ach(b, recuperavel(f)), t) &\in PCFXSM(M_b(t)) \end{aligned}$$

de um modo semelhante ao referido anteriormente.

O processo de abdução de acções criará duas novas restrições de integridade:

$$holds\_at(apagado(f), t_\infty) \Leftarrow .$$
$$holds\_at(recuperavel(f), t_\infty) \Leftarrow .$$

O processo abdutivo que possibilita a satisfação destes objectivos já só obterá um modelo que incluirá a acção de mover o ficheiro:

$$happens(e_1, t_1, t_1).$$
$$act(e_1, mv < f > < f.bak >).$$
$$t < t_1$$

Conforme foi possível demonstrar pela apresentação deste exemplo simples, a metodologia proposta permite a inferência de acções a realizar em diversos tipos de situações, dependendo da informação que é veículada através dos actos de fala.

5.7 Conclusões

Neste capítulo foi proposta uma metodologia para a participação de agentes em diálogos. Esta metodologia baseia-se na inferência de atitudes descrita no capítulo anterior como suporte para a geração de planos e de acções.

Esta metodologia permite a um agente:

1. Efectuar a inferência das atitudes dos agentes;
2. Efectuar a inferência abdutiva dos planos que permitem satisfazer os seus objectivos;
3. Seleccionar o plano mais adequado;
4. Gerar as acções correspondentes e pssá-las para um módulo de geração de LN e de outras acções.

A inferência das atitudes dos agentes baseia-se na descrição dos diversos actos de fala através de regras de programação em lógica e da utilização de regras que definem o comportamento dos agentes, nomeadamente do processo de transferência de atitudes entre eles.

O processo de detecção e supressão de contradições proposto ([AP96]) permite que o sistema suporte situações de revisão de atitudes durante o decurso dos diálogos. O processo de revisão de intenções em diálogos, incorporado num ambiente formal de programação em lógica, é algo que é inovador e que permite suportar classes de diálogos que não são suportados normalmente pelas abordagens tradicionais, nomeadamente, ao permitir a definição de preferências entre os diversos modelos. Estão nestas situações os

diálogos em que existem situações de planeamento criadoras de intenções contraditórias com o modelo do agente.

De facto, quer os trabalhos de Pollack ([Pol86, Pol90]), quer os trabalhos de Litman e Allen ([Lit85, LA87]) não suportam a resolução de contradições nas intenções de realizar acções. Estas abordagens consideram que são situações de erro anómalas.

Ferguson ([Fer95]) permite a revisão dos modelos existentes, só que não representa a relação entre as acções e os estados mentais, não detectando contradições resultantes de inconsistências em termos de estado mental (intenções contraditórias, por exemplo).

A inferência abdutiva dos planos é efectuada dentro do ambiente de programação em lógica proposto, forçando os objectivos existentes a serem satisfeitos num modelo hipotético do futuro.

O processo de geração de actos de fala proposto permite executar os planos abduzidos e efectuar a transformação das intenções não planeadas de realizar eventos em eventos propriamente ditos, e incorporá-los nos modelos dos agentes, de modo a condicionar comportamentos futuros. Este processo, embora não contemple aspectos linguísticos da geração de discurso, permite suportar a geração dos eventos e, deste modo, possibilitar o estabelecimento e a continuação dos diálogos.

Foi, ainda, apresentado um módulo que integra e gere as diversas componentes de participação em diálogos. Este é um dos aspectos fundamentais da metodologia proposta: a sua capacidade de incorporar num ambiente formal de programação em lógica as diversas componentes de um sistema de participação em diálogos (desde o reconhecimento de atitudes à geração de actos de fala).

Este sistema deverá, no entanto, ser inserido dentro de uma arquitectura global de participação em diálogos que permita a existência de um meta-nível de gestão de diálogos com controlo de tópicos e focos de conversação, ciclos de conversação e acções do discurso de um nível superior (introduzir, mudar e alterar tópicos).

Lopes e Quaresma [Lop86, Lop91, QL92]), em trabalho anterior, propuseram uma arquitectura para a participação em diálogos que resolvia alguns destes problemas. No entanto, esta arquitectura não é baseada num ambiente de programação em lógica, pelo que, como trabalho futuro, poderá ser efectuada a integração do sistema proposto nesta tese e uma adaptação das arquitecturas propostas (ver capítulo 8).

Capítulo 6

Exemplos

Neste capítulo pretendo ilustrar as potencialidades do sistema proposto, demonstrando a sua aplicabilidade a um conjunto de diálogos.

Na secção 6.2 é apresentado um exemplo de diálogos cooperativos que permitem demonstrar o processo de resolução de alguns dos problemas existentes num processo de participação em diálogos, nomeadamente o processo de actualização e revisão do estado mental do agente.

De seguida, na secção 6.3, apresento um exemplo de um ambiente em que existem agentes com modelos mentais distintos. É realçado o processo de planeamento abdutivo das acções que permitem atingir os objectivos desejados.

Na secção 6.4, em conclusão, são enumerados alguns dos aspectos mais positivos dos sistemas de participação em diálogos apresentados neste capítulo.

6.1 Introdução

O sistema de participação em diálogos proposto deste trabalho permite a sua aplicação a diversos domínios. Neste capítulo ir-se-ão apresentar alguns exemplos que, por um lado, permitem ilustrar a teoria apresentada anteriormente e, por outro, permitem demonstrar a aplicabilidade da teoria proposta a um conjunto variado de domínios.

O primeiro exemplo de aplicação pertence ao domínio de diálogos cooperativos de busca de informação sobre comboios. Este domínio tem sido utilizado em diversos sistemas de diálogos ([Lit85, GAT93, HA95]) e permite aferir da capacidade do sistema proposto em solucionar problemas típicos de sistemas de gestão de diálogos (reconhecimento e geração de planos, actualização e revisão do modelo do agente).

Como segundo exemplo, é apresentado um ambiente em que o agente/sistema computacional interage com outro agente que possui um modelo comportamental distinto. Este exemplo permite ilustrar a capacidade do sistema proposto para modelar situações em que os agentes interlocutores não são *bem comportados*, em que as acções não são exclusivamente compostas por actos de fala, e em que os agentes necessitam de planear abdutivamente as acções necessárias à satisfação dos seus objectivos.

6.2 Diálogos sobre comboios

Nesta secção serão apresentados em detalhe dois exemplos típicos de diálogo sobre o domínio de comboios. Nomeadamente serão apresentados diálogos em que um passageiro recorre a um balcão de informações sobre comboios para obter informações de que necessita.

Pretende-se um sistema com capacidade para ultrapassar situações de informação incompleta por parte do passageiro, com capacidade para inferir as intenções dos passageiros a partir dos respectivos actos de fala e com a capacidade para planear e gerar os seus próprios actos de fala.

O primeiro exemplo pretende demonstrar o processo de planeamento de acções a realizar com vista à satisfação dos objectivos do empregado/sistema computacional. Estes objectivos são obtidos a partir do reconhecimento dos objectivos dos passageiros interlocutores. O processo de planeamento implica a abdução de acções e a actualização do modelo do empregado.

Será utilizado o seguinte diálogo:

- P: Para ir para o Porto?
- E: Pode embarcar no comboio às 8:30 para Campanhã. Tem de comprar bilhete e ir para a linha 7.
- P: Queria que a hora de partida fosse próximo das 15:00.

- E: Não há nenhum comboio com essas características.
- P: Então quero um bilhete para o comboio das 8:30.

O segundo exemplo analisado já foi introduzido em capítulos anteriores para salientar alguns aspectos específicos desses capítulos, sendo nesta secção apresentado na sua globalidade. Este diálogo, sendo já clássico na área de gestão de diálogos, permite, também, a comparação do sistema proposto com outros sistemas já existentes.

O exemplo analisado será o seguinte (adaptado de [Lit85]):

- P: O comboio das 8:30?
- E: Para Campanhã? Linha 7.
- P: Não, o que chega de Leiria.
- E: Chega na linha 5!
- P: Obrigado.

O primeiro passo no processo de definição de um agente computacional que suporte este tipo de diálogos, é o da definição exacta do modelo de utilizador a utilizar, nomeadamente, o conhecimento específico do domínio e do mundo exterior e as regras que definem o seu comportamento

Na sub-secção seguinte será apresentado em detalhe o modelo do empregado/sistema computacional que foi utilizado nos dois exemplos.

6.2.1 Modelo do empregado

O empregado do balcão de informações deverá apresentar um comportamento cooperativo, verdadeiro, crédulo e reactivo. Isto é:

1. O agente coopera com os outros agentes de modo a tentar satisfazer os seus objectivos;
2. O agente é sincero, só transmitindo informação que considera correcta;
3. O agente é crédulo, acreditando na informação que lhe é veículada pelos outros agentes e que não seja incompatível com as suas próprias crenças;
4. O agente é reactivo, na medida em que age respondendo às solicitações dos outros agentes.

Em suma, o agente modela um sistema ideal de busca de informação.

De acordo com o apresentado no capítulo 3, este agente será modelado através da seguinte estrutura:

$$M = < At, Rc, Ac, T, Rr, CM >$$

Nesta estrutura:

1. At representa as atitudes do agente (crenças, intenções e objectivos);
2. Rc representa as regras que definem o comportamento do agente;
3. Ac representa a descrição das acções passíveis de serem realizadas;
4. T representa os axiomas temporais;
5. Rr representa as regras que definem a racionalidade do agente;
6. CM representa as regras que definem o conhecimento que o agente possui sobre o mundo que o rodeia.

Estas componentes serão analisadas nas sub-secções seguintes.

Axiomas Temporais

Os axiomas temporais (T) são os definidos no capítulo 2 e permitem, nomeadamente:

1. Inferir quais as propriedades válidas em cada instante de tempo;
2. Definir restrições de integridade sobre a modelação dos eventos;
3. Abduzir eventos associados a acções

Estes axiomas estão representados pelas regras 2.1 a 2.15.

Regras de Racionalidade

As regras de racionalidade (Rr) utilizadas foram definidas no capítulo 3 e permitem modelar um comportamento racional do agente, definindo as relações existentes entre as diversas atitudes (crenças, objectivos e intenções).

A racionalidade está definida pelas regras 3.1 a 3.15 (as regras que se encontram descritas na linguagem A_{EC} deverão ser traduzidas para regras de programação em lógica estendida, de acordo com o processo apresentado no capítulo 2).

Regras de Comportamento

As regras de comportamento foram definidas no capítulo 3, sendo utilizadas as regras que definem um comportamento:

1. Crédulo (regras 3.16 e 3.18);
2. Verdadeiro (analisado na sub-secção seguinte);
3. Cooperativo (regras 3.47 e 3.52);
4. Reactivo (o agente não possui objectivos iniciais próprios).

Descrição de acções

Os actos de fala são os descritos no capítulo 3 para emissores verdadeiros (regras 3.37 a 3.41) e para receptores crédulos (regras 3.29 a 3.36). Esta descrição deverá ser traduzida para regras de programação em lógica.

Como regras específicas do domínio foram identificadas as seguintes acções:

1. ir_para_linha(Agente, Linha, Estação)
2. esperar(Agente, Comboio, Estação)
3. embarcar(Agente, Comboio, Estação, Hora)
4. ir_de_comboio_a(Agente, Comboio, Destino).
5. comprar_bilhete(Agente, Destino)

A primeira acção pode ser representada, de um modo bastante simplificado, por:

$$ir\_para\_linha(Agente, Linha, Estacao) \quad causes\ na\_linha(Agente, Linha)\ if\ em(Agente, Estacao),\ tem\_linha(Estacao, Linha).$$

Esta expressão significa que se um agente estiver na estação de comboio e se executar a acção de ir para uma linha de comboio, então ele passa a estar nessa linha. O predicado *tem_linha* será definido na sub-secção seguinte.

Assume-se nesta representação que não existem outros pré-requisitos à acção de ir a uma determinada linha, isto é, o agente consegue sempre executar a acção.

A tradução em regras de programação em lógica será:

$$\begin{aligned} enabled(E,T_i) \leftarrow\ & act(E, ir\_para\_linha(A,L,Es)), \\ & holds\_at(em(A,Es),T_i), \\ & holds\_at(tem\_linha(Es,L),T_i). \end{aligned} \tag{6.1}$$

$$\begin{aligned} initiates(E,T_f,na\_linha(A,L)) \leftarrow\ & happens(E,T_i,T_f), \\ & act(E, ir\_para\_linha(A,L,Es)), \\ & holds\_at(em(A,Es),T_i) \\ & holds\_at(tem\_linha(Es,L),T_i). \end{aligned} \tag{6.2}$$

A segunda acção é definida por (nesta representação assumiu-se que esperar um comboio é uma acção e não um estado):

$$\begin{aligned} & esperar(Agente, Comboio, Estacao) \\ & if\ na\_linha(Agente, Linha), \\ & \quad ha\_comboio(Comboio, Linha, Estacao). \end{aligned}$$

Esta expressão significa que um agente espera um comboio numa estação se estiver numa linha de comboio onde passa o comboio. O predicado *há_comboio* será definido na próxima secção.

A sua tradução será:

$$\begin{aligned} enabled(E,T_i) \leftarrow\ & act(E, esperar(A,C,Es)), \\ & holds\_at(na\_linha(A,L),T_i), \\ & holds\_at(ha\_comboio(C,L,Es),T_i). \end{aligned} \tag{6.3}$$

A terceira acção é definida por:

$$\begin{aligned} & embarcar(Agente, Comboio, Estacao)\ causes\ em(Agente, Comboio) \\ & if \\ & em(Agente, Estacao), \\ & na\_linha(Agente, Linha), \\ & ha\_comboio(Comboio, Linha, Estacao). \end{aligned}$$

Esta regra significa que, num dado instante de tempo, um agente embarca num comboio numa dada estação, se o agente estiver na estação e na linha onde passa o comboio. Como consequência, o agente passa a estar no comboio.

A tradução em regras de programação em lógica é:

$$\begin{aligned} enabled(E,T_i) \leftarrow\ & act(E, embarcar(A,C,Es)), \\ & holds\_at(em(A,Es),T_i), \end{aligned} \tag{6.4}$$

$$\begin{aligned} & & & holds\_at(na\_linha(A,L),T_i), \\ & & & holds\_at(ha\_comboio(C,L,Es),T_i). \\ initiates(E,T_f,em(A,C)) & \leftarrow & & happens(E,T_i,T_f), \\ & & & act(E,embarcar(A,C,Es)), \\ & & & holds\_at(em(A,Es),T_i), \\ & & & holds\_at(na\_linha(A,L),T_i), \\ & & & holds\_at(ha\_comboio(C,L,Es),T_i). \end{aligned} \tag{6.5}$$

A quarta acção é definida por:

$$\begin{aligned} ir\_de\_comboio\_a(Agente,Comboio,Local) & \quad causes\ em(Agente,Local) \\ & \quad if \quad tem\_bilhete(Agente,Local), \\ & \quad em(Agente,Comboio), \\ & \quad destino(Comboio,Local). \end{aligned} \tag{6.6}$$

Esta regra significa que um agente apanha um comboio para um dado local se tiver um bilhete para esse local e se estiver no comboio que tem esse destino. Além disso, a acção terá como consequência que o agente passará a estar no local desejado. Como se pode observar, esta é uma versão simplificada desta acção (não tem em linha de conta possíveis problemas no decurso da acção de viajar de comboio que poderão impedir a sua concretização).

A sua tradução será:

$$\begin{aligned} enabled(E,T_i) & \leftarrow & & act(E,ir\_de\_comboio\_a(A,C,L)), \\ & & & holds\_at(tem\_bilhete(A,L),T_i), \\ & & & holds\_at(em(A,C),T_i), \\ & & & holds\_at(destino(C,L),T_i). \end{aligned} \tag{6.7}$$

$$\begin{aligned} initiates(E,T_f,em(A,L)) & \leftarrow & & happens(E,T_i,T_f), \\ & & & act(E,ir\_de\_comboio\_a(A,C,L)), \\ & & & holds\_at(tem\_bilhete(A,L),T_i), \\ & & & holds\_at(em(A,C),T_i), \\ & & & holds\_at(destino(C,L),T_i). \end{aligned} \tag{6.8}$$

A última acção é definida por:

$$comprar\_bilhete(Agente,Destino)\ causes\ tem\_bilhete(Agente,Destino)$$

Esta regra representa a acção de comprar bilhete para um dado destino e tem como efeito a posse do bilhete. Por questões de simplificação, não se consideraram pré-requsitos à acção (ter dinheiro, ainda haver bilhetes, etc.), assumindo-se a sua concretização sempre que necessário.

A sua tradução será:

$$enabled(E, T_i) \leftarrow act(E, comprar\_bilhete(A, D)), \quad (6.9)$$

$$initiates(E, T_f, tem\_bilhete(A, D)) \leftarrow happens(E, T_i, T_f), \quad (6.10)$$
$$act(E, comprar\_bilhete(A, D)).$$

Conhecimento do Mundo

Este parâmetro do modelo dos agentes representa o conhecimento do mundo (CM) que não é capturado pelos outros parâmetros. Estão nestas condições os eventos que vão sucedendo no tempo, a representação de conhecimento sobre as diversas entidades e as regras de relação entre conceitos.

Em primeiro lugar é necessário definir o agente/sistema computacional e os seus interlocutores (assume-se a existência de só um interlocutor — o passageiro):

$$eu(empregado). \quad (6.11)$$

$$voce(passageiro). \quad (6.12)$$

Por questões de simplificação, nas regras seguintes *empregado* será representado por e e o *passageiro* será representado por p.

As estações de caminho-de-ferro existentes são as seguintes (simplificando o número de estações existentes):

$$estacao(staApolonia). \quad (6.13)$$

$$estacao(campanha). \quad (6.14)$$

$$estacao(leiria). \quad (6.15)$$

Um agente que se encontre na estação de Sta. Apolónia está, também, em Lisboa e, de um modo análogo, se estiver na estação de Campanhã, está no Porto:

$$holds\_at(em(A, lisboa), T) \leftarrow holds\_at(em(A, staApolonia), T). \quad (6.16)$$

$$holds\_at(em(A, porto), T) \leftarrow holds\_at(em(A, campanha), T). \quad (6.17)$$

No instante inicial t_0, o empregado e o passageiro encontram-se na estação de comboios de Sta. Apolónia em Lisboa:

$$happens(e_0, t_0, t_0). \quad (6.18)$$

$$act(e_0, start). \quad (6.19)$$

$$initiates(e_0, t_0, em(e, staApolonia)). \quad (6.20)$$

$$initiates(e_0, t_0, em(p, staApolonia)). \quad (6.21)$$

Note-se a definição de um evento inicial e_0 que inicia as propriedades desejadas.

A estação de Sta. Apolónia tem 7 linhas, a de Campanhã tem 5 linhas e a de Leiria tem 3 linhas:

$$initiates(e_0,t_0,tem\_linha(staApolonia,1)). \tag{6.22}$$
...
$$initiates(e_0,t_0,tem\_linha(staApolonia,7)). \tag{6.23}$$
$$initiates(e_0,t_0,tem\_linha(campanha,1)). \tag{6.24}$$
...
$$initiates(e_0,t_0,tem\_linha(campanha,5)). \tag{6.25}$$
$$initiates(e_0,t_0,tem\_linha(leiria,1)). \tag{6.26}$$
...
$$initiates(e_0,t_0,tem\_linha(leiria,3)). \tag{6.27}$$

Por outro lado, neste exemplo, assume-se a existência de um conjunto de factos que definem os horários de cada comboio:

$$horario(Comboio, Estacao, HoraChegada, HoraPartida, Linha).$$

Este predicado representa, para cada comboio $Comboio$ e para cada estação $Estacao$, o tempo de chegada $HoraChegada$ a essa estação, o tempo de partida $HoraPartida$, e a linha $Linha$ onde chega/parte.

Note-se que um tempo de partida/chegada indefinido ($-$) significa que a estação é o local de chegada/partida do comboio.

Como exemplo, suponhamos que a base de conhecimentos existente é a seguinte (com bastantes simplificações de horários):

$$horario(a, staApolonia, -, 8:30, 7). \tag{6.28}$$
$$horario(a, campanha, 10:30, -, 3). \tag{6.29}$$
$$horario(b, leiria, -, 6:50, 2). \tag{6.30}$$
$$horario(b, lisboa, 8:30, -, 5). \tag{6.31}$$

Existe a necessidade de definir regras para os predicados $destino/2$ e $há\_comboio/3$:

$$holds\_at(destino(C,E),T) \leftarrow horario(C,E,HC,-,L). \tag{6.32}$$
$$holds\_at(ha\_comboio(C,L,E),T) \leftarrow horario(C,E,HC,T,L). \tag{6.33}$$

Estas regras significam que o destino de um comboio é uma dada estação se existir um horário que define essa estação como estação de chegada do comboio. Por outro lado, num dado instante de tempo, existe um comboio numa linha de uma estação, se existir um horário compatível com estes dados (assume-se a pontualidade dos comboios).

De modo a modelar o comportamento do empregado é necessário, ainda, definir algumas regras pragmáticas que permitam uma interacção mais *inteligente*. Assim, o empregado não deve informar sobre o que é desnecessário, deve dizer somente o estritamente necessário para responder aos pedidos feitos pelos passageiros. Por outro lado, o empregado deve assumir que o comportamento mais *habitual* dos passageiros é o de obter informações sobre comboios que partem, e não sobre comboios que chegam à estação (excepto se houver um pedido explícito de informação nesse sentido).

A primeira regra (restrição de integridade) é a seguinte:

$$\begin{aligned} \Leftarrow \quad & holds\_at(int(E, informref(E, P, T_1, comboio(X))), T), \\ & holds\_at(bel(E, bel(P, ref(T_1, comboio(X)))), T). \end{aligned} \tag{6.34}$$

Esta restrição significa que é contraditório alguém ter como intenção informar outro agente sobre um dado termo de um comboio se acreditar que esse outro agente já sabe esse termo.

A segunda regra,

$$\begin{aligned} & bel(E, bel(P, ref(horaSaida(T), comboio(X)))) \\ & if \\ & bel(E, bel(P, ref(hora(T), comboio(X)))), \\ & not\ bel(E, bel(P, ref(horaChegada(T), comboio(X)))). \end{aligned} \tag{6.35}$$

Esta regra indica que um empregado pode assumir que o tempo mencionado num diálogo é o tempo de partida, se não houver evidências contrárias. Esta é uma regra que modela comportamentos por omissão.

A regra é traduzida por:

$$\begin{aligned} & holds\_at(bel(E, bel(P, ref(horaSaida(H), comboio(X)))), T) \leftarrow \\ & \quad holds\_at(bel(E, bel(P, ref(hora(H), comboio(X)))), T), \\ & \quad not\ holds\_at(bel(E, bel(P, ref(horaChegada(H), comboio(X)))), T). \end{aligned} \tag{6.36}$$

É necessário, ainda, definir um conjunto de restrições de integridade entre as intenções de realizar acções do domínio e as crenças sobre propriedades dessas acções.

No âmbito deste exemplo apresentarei somente uma dessas regras, que será necessária para o processamento do primeiro exemplo:

$$\begin{aligned} \Leftarrow \quad & holds\_at(int(A, embarcar(B, C, E)), T), \\ & holds\_at(bel(A, ref(horaSaida(HS), comboio(C))), T), \\ & horario(C, E, HC, HS', L), \\ & not(HS = HS'). \end{aligned} \tag{6.37}$$

Esta restrição de integridade impede que se tenha como intenção que alguém embarque num comboio num dado horário e, simultaneamente, acreditar que a hora de partida desse comboio é distinta.

De um modo análogo seria possível criar restrições de integridade para as outras propriedades do domínio (horas de chegada, linhas, estações) e para as outras acções do domínio.

Atitudes do agente

Neste exemplo, existem algumas atitudes iniciais por parte do empregado: o seu modelo comportamental e o modelo que possui sobre o passageiro, bem como as suas crenças sobre os horários dos comboios.

É, portanto, necessário definir o modelo do agente/sistema computacional no instante inicial t_0 (cooperativo, verdadeiro, crédulo e reactivo):

$$initiates(e_0, t_0, bel(e, cooperativo(e))). \tag{6.38}$$
$$initiates(e_0, t_0, bel(e, verdadeiro(e))). \tag{6.39}$$
$$initiates(e_0, t_0, bel(e, credulo(e))). \tag{6.40}$$
$$initiates(e_0, t_0, bel(e, reactivo(e))). \tag{6.41}$$

O modelo do passageiro (agente cooperativo, verdadeiro, credulo e pró-activo), do ponto de vista do empregado/sistema computacional é :

$$initiates(e_0, t_0, bel(e, cooperativo(p))). \tag{6.42}$$
$$initiates(e_0, t_0, bel(e, verdadeiro(p))). \tag{6.43}$$
$$initiates(e_0, t_0, bel(e, credulo(p))). \tag{6.44}$$
$$initiates(e_0, t_0, bel(e, proactivo(p))). \tag{6.45}$$

As crenças sobre os horários de comboios são criadas a partir da consulta aos factos que representam esses horários.

$$holds\_at(bel(emp, ref(de(L), comboio(X))), T) \leftarrow \tag{6.46}$$
$$\quad horario(X, L, -, H, G).$$
$$holds\_at(bel(emp, ref(para(L), comboio(X))), T) \leftarrow \tag{6.47}$$
$$\quad horario(X, L, H, -, G).$$
$$holds\_at(bel(emp, ref(horaSaida(HS), comboio(X))), T) \leftarrow \tag{6.48}$$
$$\quad horario(X, L, HC, HS, G).$$
$$\quad holds\_at(em(e, L), T).$$

$$holds\_at(bel(emp, ref(horaChegada(HC), comboio(X))), T) \leftarrow \quad (6.49)$$
$$horario(X, L, HC, HS, G).$$
$$holds\_at(em(e, L), T).$$
$$holds\_at(bel(emp, ref(linha(G), comboio(X))), T) \leftarrow \quad (6.50)$$
$$horario(X, L, HC, HS, G),$$
$$holds\_at(em(e, L), T).$$

Estas regras permitem obter crenças sobre referências de comboios com base nos horários. A primeira define o local de partida do comboio; a segunda o local de chegada; a terceira define a hora de partida da estação onde está o empregado; a quarta a hora de chegada a essa estação; e a quinta a linha de chegada/saída.

6.2.2 Planeamento de acções

Conforme referido anteriormente, o primeiro exemplo pretende demonstrar essencialmente o processo de planeamento de acções a realizar, nomeadamente o processo de abdução de acções e de actualização do modelo do empregado.

Para efeitos de simplificação, apresento, novamente, o exemplo na sua totalidade:

- P: Para ir para o Porto?
- E: Pode embarcar no comboio às 8:30 para Campanhã. Tem de comprar bilhete e ir para a linha 7.
- P: Queria que a hora de partida fosse próximo das 15:00.
- E: Não há nenhum comboio com essas características.
- P: Então quero um bilhete para o comboio das 8:30.

A primeira frase do passageiro cria os seguintes factos:

$$happens(e_1, t_1, t_2). \quad (6.51)$$
$$act(e_1, inform(p, e, ach(p, em(p, porto)))). \quad (6.52)$$

Estes factos representam o acto de informar sobre o seu objectivo: estar no Porto.

Com base na definição do acto de fala $inform$ (3.29) e nos axiomas temporais (regras 2.1 a 2.15) tem-se que:

$$holds\_at(bel(e, bel(p, ach(p, em(p, porto)))), t_2). \quad (6.53)$$

Utilizando as regras de transferência de crenças 3.16 e 3.18 tem-se:

$$holds\_at(bel(e, ach(p, em(p, porto))), t_2). \quad (6.54)$$

Através da regra de transferência de objectivos 3.52:

$$holds\_at(ach(e, em(p, porto)), t_2). \tag{6.55}$$

Ou seja, o empregado adopta como seus os objectivos que lhe foram transmitidos pelo passageiro.

O processo de planeamento proposto no capítulo 5 cria como restrição de integridade a satisfação do objectivo inferido:

$$holds\_at(em(p, porto), t_\infty) \Leftarrow \tag{6.56}$$

O modelo do empregado acrescido desta restrição de integridade (modelo hipotético), suporta a abdução das acções necessárias à satisfação desse estado. Relembre-se que, de acordo com a restrição de integridade 2.14, não é possível existir um estado sem o evento que o iniciou.

De um modo resumido, o processo é o seguinte:

A restrição de integridade é satisfeita se o seguinte facto for válido (usando a regra 6.17):

$$holds\_at(em(p, campanha), t_\infty). \tag{6.57}$$

A acção $ir\_de\_comboio\_a(p, C, campanha)$ é abduzida como modo de satisfazer o estado anterior (quarta regra do domínio):

$$happens(e'_1, t'_1, t'_2). \tag{6.58}$$

$$act(e'_1, ir\_de\_comboio\_a(p, C, campanha)). \tag{6.59}$$

As pré-condições desta acção têm de ser satisfeitas (regra 2.15):

1. $tem\_bilhete(p, campanha)$.
2. $destino(C, campanha)$.
3. $em(p, C)$.

A primeira pré-condição é satisfeita através da abdução da acção:

$$happens(e'_2, t'_3, t'_4). \tag{6.60}$$

$$act(e'_2, comprar\_bilhete(p, campanha)). \tag{6.61}$$

A segunda pré-condição é satisfeita através da regra 6.32 e da consulta dos horários de comboio:

$$horário(a, campanha, 10:30, -, 5). \tag{6.62}$$

A terceira pré-condição implica a abdução de outra acção (embarcar no comboio):

$$happens(e'_3, t'_5, t'_6). \tag{6.63}$$

$$act(e'_3, embarcar(p, a, staApolonia)). \tag{6.64}$$

No entanto, esta acção tem como pré-condições:

1. $em(p, E)$
2. $ha\_comboio(a, E, L)$
3. $na\_linha(p, L)$.

A primeira pré-condição é satisfeita pelo evento inicial e_0 (sendo $E = staApolonia$).

A segunda pré-condição é satisfeita pela utilização da regra 6.33 e pela consulta aos horários (sendo $E = staApolonia$, $HP = 8:30$, $HC = -$, $L = 7$ e $t'_5 = HP$).

A terceira pré-condição é satisfeita através da abdução da acção de ir para uma linha de uma estação:

$$happens(e'_4, t'_7, t'_8). \tag{6.65}$$
$$act(e'_4, ir\_para\_linha(p, 7, staApolonia)). \tag{6.66}$$

As pré-condições desta acção estão satisfeitas no modelo do empregado.

Resumindo, o processo de planeamento abduz quatro acções necessárias à satisfação do objectivo do empregado:

$$ir\_de\_comboio\_a(p, a, campanha).$$
$$embarcar(p, a, staApolonia).$$
$$comprar\_bilhete(p, campanha).$$
$$ir\_para\_linha(p, 7, staApolonia).$$

Tendo o comboio a as seguintes características:

$$horario(a, staApolonia, -, 8:30, 7).$$

Com base nas intenções de executar estas acções, é possível gerar frases em Língua Natural semelhantes à descrita no exemplo. Como referi anteriormente, este processo não será analisado neste trabalho.

Continuando a análise do exemplo apresentado, a segunda frase do passageiro cria uma nova restrição em relação à hora de partida do comboio em análise:

$$happens(e_2, t_3, t_4). \tag{6.67}$$
$$act(e_2, inform(p, e, ref(horaSaida(15:00), comboio(X)))). \tag{6.68}$$

Utilizando a regra que descreve a acção $inform$ (3.29) e o processo de transferência de crenças (regras 3.16 e 3.18), temos:

$$holds\_at(bel(e, ref(horaSaida(15:00), comboio(X)))), t_4). \tag{6.69}$$

O processo de planeamento descrito no exemplo anterior não consegue obter um modelo consistente. De facto, a intenção de realizar a acção

$$embarcar(p, C, staApolonia).$$

não pode ser iniciada porque não há nenhum comboio nas condições requeridas (crença anterior e restrição de integridade 6.37).

Como consequência, o empregado/sistema computacional poderá informar o passageiro de que não consegue satisfazer os seus objectivos — segunda frase do empregado.

Através da análise do processamento deste exemplo, é possível concluir que o sistema proposto tem a capacidade para efectuar um planeamento abdutivo das acções a realizar e tem a capacidade para actualizar esse plano, de acordo com nova informação que lhe é transmitida.

6.2.3 Diálogo com informação incompleta

Vejamos, novamente, o segundo exemplo de diálogos sobre comboios a analisar neste capítulo:

- P: O comboio das 8:30?
- E: Para Campanhã? Linha 7.
- P: Não, o que chega de Leiria.
- E: Ah, chega na linha 5!
- P: Obrigado.

A primeira frase do passageiro cria os seguintes factos (o passageiro é representado por p e o empregado/sistema computacional por e):

$$happens(e_1, t_1, t_2). \tag{6.70}$$
$$act(e_1, request(p, e, informref(e, p, P, comboio(X)))). \tag{6.71}$$
$$happens(e_2, t_1, t_2). \tag{6.72}$$
$$act(e_2, inform(p, e, ref(hora(8:30), comboio(X)))). \tag{6.73}$$

Estes factos descrevem o pedido de informação efectuado pelo passageiro acerca de uma referência de um comboio. Além disso, é dada uma informação sobre um tempo de referência de um comboio.

Optou-se, neste processo de reconhecimento, por considerar que houve dois eventos simultâneos, em contrapartida à possibilidade de reconhecer um evento com duas acções com execução paralela. Deste modo é possível identificar e distinguir os eventos e as suas acções associadas.

Com base no modelo do empregado e nos actos de fala efectuados, o modelo bem fundado do programa verifica o seguinte facto:

$$holds\_at(bel(e, int(p, informref(e, p, P, comboio(X)))), t_2). \tag{6.74}$$

usando os axiomas temporais 2.1 a 2.15 e a regra 3.32 para o acto *request*.

Este facto significa que o empregado acredita que o passageiro pretende ser informado sobre uma dada propriedade de um comboio.

A partir da regra de cooperatividade 3.47 e do facto anterior obtém-se que:

$$holds\_at(int(e, informref(e, p, P, comboio(X)))) \tag{6.75}$$

que significa que o empregado passou a ter como intenção informar o passageiro.

Por outro lado, a regra 3.29 do acto *inform* e o evento e_2 permitem obter:

$$holds\_at(bel(e, bel(p, ref(hora(8:30), comboio(X)))), t_2). \tag{6.76}$$

que significa que o empregado acredita que o passageiro está interessado num comboio com uma dada referência — hora igual a 8:30.

Utilizando a segunda regra do domínio 6.36 e o facto anterior obtém-se:

$$holds\_at(bel(e, bel(p, ref(horaSaída(8:30), comboio(X)))), t_2). \tag{6.77}$$

que significa que o comboio tem uma hora de saída de 8:30 (não existe nenhuma evidência que indique que o passageiro está interessado num comboio que esteja a chegar).

O processo de planeamento descrito no capítulo anterior faria a selecção da intenção inferida 6.75 e tentaria verificar a validade das suas pré-condições.

Tendo em conta a definição 3.30 da acção $informref$, a restrição de integridade do domínio 6.34, as regras que definem as atitudes do empregado a partir dos horários dos comboios (6.46 a 6.50) é possível obter quais as acções que são suportadas pelo modelo do empregado:

$$informref(e, p, ref(para(campanha), comboio(a))). \tag{6.78}$$

$$informref(e, p, ref(linha(7), comboio(a))). \tag{6.79}$$

Estas intenções representam a informação que o empregado crê que o passageiro ainda não possui sobre o comboio. De notar que a informação sobre a hora de saída do comboio não é suportada pelo modelo (restrição de integridade 6.34). Além disso, não está representado o processo que permite resolver o problema da existência de variáveis não instanciadas no programa em lógica estendida que modela o empregado. Este processo depende directamente de opções relativas à construção do protótipo do sistema e será abordado no capítulo 7.

O processo de geração descrito no capítulo anterior conduziria, então, à geração dos seguintes actos de fala:

$$happens(e_3, t_3, t_4). \tag{6.80}$$

$$act(e_3, informref(e, p, ref(para(campanha), comboio(a)))). \tag{6.81}$$

$$happens(e_4, t_3, t_4). \tag{6.82}$$

$$act(e_4, informref(e, p, ref(linha(7), comboio(a)))).. \tag{6.83}$$

Após a geração destes eventos, as intenções existentes são terminadas, assumindo-se uma correcta realização das acções.

Se, conforme foi apresentado no exemplo, o passageiro não aceita a informação fornecida e clarifica o termo 8:30 como um tempo de chegada do comboio de Leiria, teremos o seguinte evento:

$$happens(e_5, t_5, t_6). \tag{6.84}$$
$$act(e_5, inform(p, e, ref(de(leiria), comboio(X)))). \tag{6.85}$$
$$happens(e_6, t_5, t_6). \tag{6.86}$$
$$act(e_6, inform(p, e, ref(horaChegada(8:30), comboio(X)))). \tag{6.87}$$
$$t_4 < t_5 \leq t_6. \tag{6.88}$$

Nestas condições, o processo de inferência será efectuado de um modo análogo ao apresentado, e obterá (a partir do evento e_5 e da acção de informar):

$$holds\_at(bel(e, bel(p, ref(de(leiria), comboio(X)))), t_6). \tag{6.89}$$
$$holds\_at(bel(e, bel(p, ref(horaChegada(8:30), comboio(X)))), t_6). \tag{6.90}$$

Efectuando o processo de planeamento semelhante ao descrito anteriormente, temos que a seguinte intenção é inferida:

$$informref(e, p, ref(linha(5), comboio(b)))). \tag{6.91}$$

Esta intenção representa a informação que o empregado crê que o passageiro ainda não possui sobre o comboio (já conhece a hora de chegada e a estação de origem).

O processo de geração criaria o seguinte acto de fala:

$$happens(e_7, t_7, t_8). \tag{6.92}$$
$$act(e_4, informref(e, p, ref(linha(5), comboio(b)))). \tag{6.93}$$
$$t_6 < t_7 \leq t_8. \tag{6.94}$$

Deste modo, e conforme foi apresentado, o sistema proposto suporta diálogos em que existem falhas de informação por parte dos agentes intervenientes.

6.3 Interacções entre agentes autónomos

Neste exemplo pretende-se modelar uma situação em que um agente interage com outros agentes, com a intenção de satisfazer os seus próprios objectivos.

O ambiente a modelar possui vários agentes, cada qual com seu tipo de comportamento. Pretende-se demonstrar que o agente tem a capacidade de se adaptar ao interlocutor corrente, utilizando para tal o modelo que possui sobre esse interlocutor.

De modo a modelar este tipo de ambiente, recorreu-se ao exemplo do capítulo 3 sobre diálogos multiagente.

Assim, existem as seguintes personagens/agentes a modelar:

1. Obélix, o agente a modelar pelo sistema computacional. Este agente é sincero, cooperativo, crédulo e pró-activo;

2. Astérix, agente interlocutor. Também é sincero, cooperativo, crédulo e pró-activo (do ponto de vista do Obélix);

3. Mentirix, agente interlocutor. Este agente é mentiroso, não cooperativo, céptico e pró-activo (do ponto de vista do Obélix).

O agente Obélix possui como objectivo inicial sentir-se realizado. Para tal, tem um conjunto de acções disponíveis que vão desde caçar javalis até bater em romanos.

A situação a modelar é a seguinte:

1. Mentirix diz a Obélix: Estão romanos no campo de Babaorum!

2. Obélix não acredita, pelo que não altera o seu comportamento.

3. Astérix diz a Obélix: Estão javalis no bosque!

4. Obélix acredita e planeia ir até ao bosque para caçar javalis.

Este exemplo permite ilustrar o processo de planeamento abdutivo e de actualização do modelo do agente, após cada acto de fala reconhecido.

Tal como na secção anterior, em primeiro lugar é apresentado o modelo do agente a representar — Obélix —, sendo posteriormente descrito o processo de inferência e planeamento.

6.3.1 Modelo do agente

Conforme referido, o agente Obélix deverá ter um comportamento cooperativo, verdadeiro, crédulo e pró-activo.

O agente será modelado através da seguinte estrutura:

$$M = < At, Rc, Ac, T, Rr, CM >$$

As diversas componentes desta estrutura serão analisadas nas sub-secções seguintes.

Axiomas Temporais

Os axiomas temporais (T) são os definidos no capítulo 2 e estão representados pelas regras 2.1 a 2.15.

Regras de Racionalidade

A racionalidade (Rr) está definida pelas regras 3.1 a 3.15.

Regras de Comportamento

As regras de comportamento utilizadas são:

1. Crédulo (regras 3.16 e 3.18);
2. Verdadeiro (analisado na sub-secção seguinte);
3. Cooperativo (regras 3.47 e 3.52);
4. Pró-activo (o agente possui objectivos iniciais próprios — sentir-se realizado). Este objectivo será representado por uma propriedade válida no instante inicial:

$$happens(e_0, t_0, t_0). \tag{6.95}$$
$$act(e_0, start). \tag{6.96}$$
$$initiates(e_0, t_0, ach(obelix, realizado(obelix))). \tag{6.97}$$

Descrição de acções

Os actos de fala são os descritos para emissores verdadeiros (regras 3.37 a 3.41) e para receptores crédulos (regras 3.29 a 3.36).

Como regras do domínio passíveis de serem executadas, temos:

1. ir_para(Agente, Local)
2. caçar_javalis(Agente)
3. bater_romanos(Agente)

A primeira acção pode ser representada, de um modo bastante simplificado, por:

$$ir\_para(Agente, Local) \qquad causes\ em(Agente, Local).$$

Esta expressão significa que os agentes podem ir para qualquer local, sem nenhuma restrição.

A tradução em regras de programação em lógica será:

$$enabled(E, T_i) \leftarrow act(E, ir\_para(A, L)). \tag{6.98}$$
$$initiates(E, T_f, em(A, L)) \leftarrow happens(E, T_i, T_f), \quad act(E, ir\_para(A, L)). \tag{6.99}$$

A segunda acção é definida por:

$$caçar\_javalis(Agente)\ causes\ realizado(Agente)$$
$$if\ em(Agente, Local),$$
$$bel(Agente, ha(javalis, Local)).$$

Esta expressão significa que um agente caça javalis se estiver no local onde acredita que eles estão. Como consequência, sente-se realizado (é o modelo do agente Obélix).

A sua tradução será:

$$\begin{aligned} enabled(E,T_i) \leftarrow\ & act(E,caçar\_javalis(A)), \\ & holds\_at(em(A,L),T_i), \\ & holds\_at(bel(A,ha(javalis,L)),T_i). \end{aligned} \tag{6.100}$$

$$\begin{aligned} initiates(E,T_f,realizado(A)) \leftarrow\ & happens(E,T_i,T_f), \\ & act(E,caçar\_javalis(A)), \\ & holds\_at(em(A,L),T_i). \\ & holds\_at(bel(A,ha(javalis,L)),T_i). \end{aligned} \tag{6.101}$$

A terceira acção é definida por:

$$\begin{aligned} & bater\_romanos(Agente)\ causes\ realizado(Agente) \\ & if\ em(Agente,Local), \\ & \quad bel(Agente,ha(romanos,Local)). \end{aligned}$$

Esta expressão significa que um agente bate em romanos se estiver no local onde acredita que eles estão. Como consequência, sente-se realizado.

A sua tradução será:

$$\begin{aligned} enabled(E,T_i) \leftarrow\ & act(E,bater\_romanos(A)), \\ & holds\_at(em(A,L),T_i), \\ & holds\_at(bel(A,ha(romanos,L)),T_i). \end{aligned} \tag{6.102}$$

$$\begin{aligned} initiates(E,T_f,realizado(A)) \leftarrow\ & happens(E,T_i,T_f), \\ & act(E,bater\_romanos(A)), \\ & holds\_at(em(A,L),T_i). \\ & holds\_at(bel(A,ha(romanos,L)),T_i). \end{aligned} \tag{6.103}$$

Conhecimento do Mundo

É necessário definir o agente/sistema computacional e os seus interlocutores:

$$eu(obelix). \tag{6.104}$$

$$voce(asterix). \tag{6.105}$$

$$voce(mentirix). \tag{6.106}$$

Atitudes do agente

Neste exemplo existe o objectivo inicial do agente se sentir realizado. Esse objectivo já foi apresentado nas regras 6.95 e 6.97.

É, também, necessário definir o modelo do agente e dos seus interlocutores:
— Obélix:

$$initiates(e_0, t_0, bel(obelix, cooperativo(obelix)). \quad (6.107)$$
$$initiates(e_0, t_0, bel(obelix, verdadeiro(obelix)). \quad (6.108)$$
$$initiates(e_0, t_0, bel(obelix, credulo(obelix)). \quad (6.109)$$
$$initiates(e_0, t_0, bel(obelix, proactivo(obelix)). \quad (6.110)$$

— Astérix:

$$initiates(e_0, t_0, bel(obelix, cooperativo(asterix)). \quad (6.111)$$
$$initiates(e_0, t_0, bel(obelix, verdadeiro(asterix)). \quad (6.112)$$
$$initiates(e_0, t_0, bel(obelix, credulo(asterix)). \quad (6.113)$$
$$initiates(e_0, t_0, bel(obelix, proactivo(asterix)). \quad (6.114)$$

— Mentirix:

$$initiates(e_0, t_0, bel(obelix, nao\_cooperativo(mentirix)). \quad (6.115)$$
$$initiates(e_0, t_0, bel(obelix, mentiroso(mentirix)). \quad (6.116)$$
$$initiates(e_0, t_0, bel(obelix, ceptico(mentirix)). \quad (6.117)$$
$$initiates(e_0, t_0, bel(obelix, proactivo(mentirix)). \quad (6.118)$$

6.3.2 Planeamento de acções

Na primeira situação descrita, Mentirix diz a Obélix que há romanos em Babaorum.

Este processo dá origem ao reconhecimento dos seguintes actos de fala (nas regras seguintes *mentirix* será abreviado por m, *asterix* por a e *obelix* por o):

$$happens(e_1, t_1, t_2). \quad (6.119)$$
$$act(e_1, inform(m, o, ha(romanos, babaorum))). \quad (6.120)$$

Após este evento, o modelo do agente Obélix é actualizado (função descrita no capítulo 4).

No entanto, o agente não aceita transferir qualquer tipo de informação, dado acreditar que o agente Mentirix é mentiroso (regra 3.29):

$$holds\_at(bel(o, mentiroso(m)), t_2). \quad (6.121)$$

Após o processo de actualização, Obélix irá fazer o seu planeamento de acções a realizar.

Para tal, irá incorporar como restrição de integridade os seus objectivos (no modelo hipotético de planeamento):

$$holds\_at(realizado(o), t_\infty) \Leftarrow \tag{6.122}$$

O processo de planeamento abdutivo não consegue satisfazer esta restrição, não abduzindo nenhuma acção. De facto, qualquer uma das acções que poderiam levar o agente ao estado desejado não pode ser abduzida, pois não é possível satisfazer as suas pré-condições. Tanto no caso de caçar javalis, como no caso de bater em romanos, não há qualquer informação no modelo que permita satisfazer a pré-condição de acreditar onde estão os javalis e/ou os romanos.

Não possuindo acções a realizar, o agente mantém os seus objectivos e aguarda novos eventos.

Com o segundo evento, o modelo de Obélix é actualizado com os seguintes factos:

$$happens(e_2, t_3, t_4). \tag{6.123}$$

$$act(e_2, inform(a, o, ha(javalis, bosque))). \tag{6.124}$$

Tendo em conta o modelo que o Obélix possui sobre o Astérix, o processo de actualização do seu estado mental cria novas crenças:

— A partir da regra de informar 3.29:

$$holds\_at(bel(o, bel(a, ha(javalis, bosque))), t_4). \tag{6.125}$$

— A partir das regras de transferência de crenças 3.16 e 3.18:

$$holds\_at(bel(o, ha(javalis, bosque)), t_4). \tag{6.126}$$

O processo de planeamento desencadeado após a actualização do estado mental do agente permite a abdução de acções que levem à satisfação do objectivo de se sentir realizado.

A acção $caçar\_javalis$ é abduzida:

$$happens(e'_1, t'_1, t'_2). \tag{6.127}$$

$$act(e'_1, caçar\_javalis(o)). \tag{6.128}$$

Esta acção tem os seguintes pré-requisitos:

1. $bel(o, há(javalis, L))$.
2. $em(o, L)$.

O primeiro pré-requisito é satisfeito pela regra 6.126.

O segundo pré-requisito é satisfeito abduzindo a acção de ir para um local:

$$happens(e'_3, t'_5, t'_6). \tag{6.129}$$
$$act(e'_3, ir\_para(o, bosque)). \tag{6.130}$$

Note-se que, devido à satisfação do primeiro pré-requisito, $L = bosque$.

Após este processo de planeamento abdutivo, o sistema cria intenções de realizar as seguintes acções:

$$int(o, ir\_para(o, bosque))$$
$$int(o, caçar\_javalis(o))$$

Estas intenções representam o plano de ir para o bosque e caçá-los.

Através deste exemplo, ilustrou-se o modo como o sistema proposto suporta situações multiagente com planeamento abdutivo das acções a realizar.

6.4 Conclusões

Neste capítulo foram apresentados exemplos do processo de participação em diálogos proposto.

Para tal, demonstrou-se a sua aplicabilidade em exemplos clássicos desta área — diálogos cooperativos com busca de informação —, realçando-se a capacidade do sistema em suportar situações não ideais, em que existe informação incompleta e ambiguidades.

Como complemento, foi apresentado um sistema que pretende modelar um agente autónomo, que interage com outros agentes através de acções que não exclusivamente actos de fala, e onde os agentes interlocutores podem ter comportamentos distintos.

O sistema proposto permite resolver o conjunto de problemas apresentado, utilizando as ferramentas definidas nos capítulos anteriores: representação do conhecimento, modelos de utilizadores, inferência de atitudes e geração de planos.

Nomeadamente, a capacidade de actualização do modelo do agente/sistema computacional permite ultrapassar situações de contradição provocadas pela adição de nova informação incompatível com o estado mental anterior, através de um processo de revisão. Este processo permite, ainda, que o agente possua memória sobre os seus diversos estados mentais e possa, em qualquer momento, efectuar uma análise introspectiva sobre o seu estado mental num dado instante de tempo anterior e fazer a sua revisão.

Esta característica permite ultrapassar os problemas existentes noutras abordagens (como por exemplo em [LA87, Car88, Pol90]), que não têm a capacidade de rever modelos mantendo informação sobre os estados anterioes.

O sistema proposto permite participar em diálogos mantendo, por exemplo, informação sobre os eventos que ocorreram ao longo do tempo, as atitudes que foram suportadas após cada um desses eventos, as acções que foi planeando executar e as revisões efectuadas.

Capítulo 7

Construção do sistema de diálogos

Neste capítulo apresento o processo de desenvolvimento e construção de um protótipo do sistema proposto nos capítulos anteriores, de modo a modelar uma participação activa e intencional em diálogos.

O desenvolvimento é efectuado sobre o ambiente de programação em lógica REVISE ([DNP94, SDP96]), sendo apresentados, para cada conceito, os problemas existentes e as soluções propostas.

Finalmente, e com o objectivo de demonstrar a aplicabilidade dos conceitos propostos, são apresentados e analisados alguns resultados experimentais obtidos.

7.1 Introdução

Um dos objectivos do trabalho que apresentei ao longo dos capítulos anteriores, foi o de permitir a construção de um protótipo com capacidade para participar activa e intencionalmente em diálogos.

O ambiente de programação em lógica utilizado é a programação em lógica estendida com negação explícita, com a revisão de programas em lógica com negação explícita, a partir do trabalho de Alferes, Damásio e Pereira ([AP96, Dam96, DNP94, SDP96]).

De modo a construir o protótipo, é necessário:

1. Definir o tipo de regras de programação em lógica estendida com a negação explícita que podem ser utilizadas, de modo a estar garantida a coerência e completude dos procedimentos de prova do demonstrador existente ([DNP94, SDP96]). Nomeadamente, não podem ocorrer ciclos infinitos e literais por omissão não completamente instanciados.

2. Construir o formalismo do Cálculo de Eventos com as alterações propostas neste trabalho;

3. Construir o processo de tradução da linguagem A_{EC} para regras de programação em lógica;

4. Representar os actos de fala;

5. Representar os operadores epistémicos;

6. Modelar os agentes;

7. Modelar o processo de participação em diálogos.

Os cinco primeiros pontos foram apresentados no capítulo 2 e estão relacionados com a representação do conhecimento necessário para a participação em diálogos. Para cada um deles existem alguns problemas relacionados com a sua construção, que serão descritos em detalhe nas secções seguintes (7.2 a 7.6).

A construção do processo de modelação dos agentes é descrita na secção 7.7, sendo o processo de participação em diálogos apresentado na secção 7.8. Este processo contempla a construção dos operadores definidos no capítulo 5, bem como a integração dos diversos módulos necessários à participação em diálogos. Serão abordadas, também, questões relativas à complexidade da solução desenvolvida.

Na secção 7.9 analiso a possibilidade de construção de um sistema distribuído em que cada agente é modelado por um processo autónomo que tem a capacidade de comunicar com os outros agentes (PVM-Prolog [MC95, Gei94])

Finalmente, na secção 7.10 apresento e analiso alguns resultados experimentais obtidos com o protótipo desenvolvido. Para a obtenção destes resultados foram utilizados os

exemplos do capítulo anterior. Estes exemplos permitem diferenciar os diversos processos necessários à participação do agente no diálogo:

1. O processo de actualização e revisão do modelo do agente;
2. O processo de planeamento de acções a realizar;
3. O processo de geração dos actos de fala.

Os resultados permitem afirmar que o sistema desenvolvido poderá ser a base de um sistema global de participação em diálogos totalmente desenvolvido num ambiente de programação em lógica.

7.2 Ambiente de Programação em Lógica

Como base do processo de participação em diálogos utilizou-se a programação em lógica estendida com a negação explícita, com a revisão de programas em lógica com negação explícita proposta no sistema REVISE [SDP96].

A opção por este ambiente de programação em lógica foi devido aos seguintes factores:

1. Existência de uma semântica declarativa e operacional que, através do recurso a procedimentos descendentes, permite a prova dos literais que pertencem ao modelo do programa em lógica;
2. Existência de procedimentos que permitem a revisão do valor lógico de assumpções, de modo a remover eventuais contradições;
3. Possibilidade de modelar diversos tipos de raciocínio não-monótono, nomeadamente, raciocínio abdutivo, hipotético e por omissão;
4. Possibilidade de construção imediata do sistema de gestão de diálogos, devido à existência de um protótipo do ambiente de programação em lógica referido (REVISE 2.4).

O procedimento REVISE é baseado no procedimento de prova SLX da WFSX descrito em [AP96], nomeadamente na versão paraconsistente da WFSX, e pode ser definido do seguinte modo (a partir de [SDP96]):

- O procedimento é baseado na construção de dois tipos de árvores-AND (árvores-T e árvores-TU), a cujos nós é atribuído o estado de sucesso ou falha. Árvores-T calculam se um literal é verdadeiro; árvores-TU calculam se é verdadeiro ou indefinido. Uma árvore com o estado de sucesso (falha) tem o nó raiz com o estado sucesso (falha). Se um literal L é raiz de uma árvore-T de sucesso, então L pertence

ao modelo paraconsistente do programa; caso contrário, se todas as árvores-T de L forem de falha, então L não pertence ao modelo.

Uma árvore-T é construída como uma árvore $SLDNF$. No entanto, quando $not\ L$ é encontrado, a árvore subsidiária de L é construída como uma árvore-TU: $not\ L$ é verdadeiro se a tentativa de provar L como verdadeiro ou indefinido, falhar. Quando $not\ L$ é encontrado numa árvore-TU, a árvore subsidiária de L é construída como uma árvore-T.

A recursividade cíclica positiva infinita é detectada localmente nas árvores-T e árvores-TU verificando se um literal depende de ele próprio. A recursividade cíclica negativa infinita é detectada através do registo do conjunto de literais antecedentes.

De referir que, embora o procedimento de prova existente no demonstrador actual — REVISE 2.4 — não garanta a terminação para o caso geral de programas não instanciados, garante-o para algumas classes de programas, como, por exemplo, os programas instanciados e os programas não instanciados que sejam admissíveis e com termos limitados (ver capítulo 2). Além disso, se o procedimento terminar então os resultados obtidos são completos e coerentes.

Neste sentido, foi necessário transformar as regras definidas nos capítulos anteriores, de modo a garantir a terminação e, como consequência, a coerência e a completude do procedimento de prova.

De acordo com o apresentado no capítulo 2, as regras de programação em lógica a utilizar deveriam possuir as seguintes características:

1. Conter termos limitados, isto é, não devem existir regras do seguinte tipo:

$$p(x) \leftarrow p(f(x))$$

2. Não existirem literais por omissão não instanciados. Isto é, em regras do tipo:

$$H \leftarrow ..., not\ C, ...$$

C deverá estar instanciado.

Pereira et al. provaram que o procedimento de prova SLX para programas com estas características termina e, consequentemente, é completo e coerente.

A primeira característica é violada pela existência de regras que não contêm termos limitados. Esta situação sucede somente em algumas regras que relacionam propriedades válidas num dado instante de tempo ($holds\_at$) como, por exemplo, as regras que relacionam as crenças dos agentes

$$holds\_at(bel(A, P), T) \leftarrow holds\_at(bel(A, bel(B, P)), T).$$

Neste tipo de regras, a limitação dos termos terá de ser garantida através da limitação do nível de recursividade das crenças. No domínio de participação em diálogos, este

nível não necessita de ser superior a 3 (crenças de um agente sobre as crenças que o outro agente possui sobre ele — $bel(A, bel(B, bel(A, P)))$). Em termos de construção do protótipo, foi adicionado um terceiro argumento que define o nível de recursividade e que permite garantir a limitação dos termos. No exemplo anterior, a regra é transformada em (utilizando o predicado $prolog/1$ do REVISE que permite executar comandos Prolog):

$$\begin{aligned} holds\_at(bel(A,P),T,N) \leftarrow\ & prolog(N < MAX\_LEVEL), \\ & prolog(N1\ is\ N+1), \\ & holds\_at(bel(A,bel(B,P)),T,N1). \end{aligned}$$

A segunda característica, exige que as regras deverão ter os literais por omissão completamente instanciados, o que implica a transformação de algumas regras apresentadas anteriormente. Este problemas está relacionado com o facto de a semântica $WFSX$ estar definida por referência ao conjunto instanciado das cláusulas. Damásio, em [Dam96], definiu uma semântica com negação construtiva e os respectivos procedimentos de prova, estando a implementação em curso recorrendo a técnicas de tabelação. Neste sentido, é de esperar que, como trabalho futuro, esta limitação possa vir a ser eliminada.

No entanto, e tendo em conta a versão do REVISE existente actualmente (2.4), é necessário garantir que os literais por omissão estão completamente instanciados. A transformação a efectuar relaciona-se com o facto de o sistema a desenvolver ir focar um domínio específico e delimitado, pelo que é admissível assumir que as entidades envolvidas estão perfeitamente definidas sempre que se efectua um processo de cálculo do modelo bem fundado do programa em lógica estendida. Deste modo, são adicionados novos factos aos programas em lógica definindo quais as entidades existentes, sejam elas agentes, acções, eventos ou instantes temporais. Note-se que este processo não implica que estas entidades estejam definidas no início do processo de modelação dos diálogos. De facto, cada evento poderá introduzir novas entidades e implicar a correspondente actualização do programa em lógica.

Os seguintes predicados foram, portanto, criados:

1. $agente(X)$ — define quais os agentes existentes que poderão interagir no diálogo;
2. $accao(A)$ — define quais as acções passíveis de serem realizadas;
3. $evento(E)$ — define os nomes de eventos possíveis;
4. $tempo(T)$ — define os instantes temporais disponíveis.

Para cada aplicação o número de constantes é tão grande quanto se queira, dado que a linguagem permite um número infinito de constantes e functores.

As secções seguintes apresentam em detalhe as alterações efectuadas às regras apresentadas ao longo da tese, de modo a garantir que os programas obtidos terminem e sejam coerentes e completos.

O protótipo existente permite, também, efectuar a revisão dos programas de modo a eliminar possíveis contradições. Esta característica é fundamental ao processo de actualização e revisão do modelo do agente descrito no capítulo 4.

O desenvolvimento do sistema de diálogos foi efectuado recorrendo ao sistema REVISE 2.4 ([DNP94, SDP96]) o qual, por sua vez, foi desenvolvido recorrendo ao Prolog. O Prolog utilizado foi o SICSTUS Prolog versão 3, e o computador um PC Pentium a 133MHz.

Os resultados experimentais obtidos são apresentados na secção 7.10 e pressupõem esta combinação de *hardware/software*.

7.3 Cálculo de Eventos

As regras de programação em lógica que descrevem o cálculo de eventos, conforme proposto no capítulo 2 para permitir a representação de eventos não instantâneos e concorrentes, têm de ser alteradas de acordo com os requisitos definidos na secção anterior, de modo a serem garantidas as condições de coerência, completude e terminação do procedimento de prova. Isto é, têm de ser alteradas as regras em que existem literais por omissão não completamente instanciados em tempo de execução.

Durante o processo de planeamento abdutivo, a abdução de eventos e de acções é permitida definindo os predicados $happens/3$ e $act/2$ como revisíveis:

$$revisable(happens(E, Ti, Tf)). \tag{7.1}$$

$$revisable(act(E, A)). \tag{7.2}$$

As regras que definem os axiomas temporais do Cálculo de Eventos são as seguintes, após a efectivação da transformação descrita anteriormente, que garante a instanciação total dos literais por omissão:

$$\begin{aligned} holds\_at(P,T) \leftarrow\ & tempo(T), \\ & tempo(Ti), \\ & tempo(Tf), \\ & evento(E), \\ & happens(E, Ti, Tf), \\ & initiates(E, T_P, P), \\ & T_P < T,\ T_i \leq T_P, \\ & persists(T_P, P, T). \end{aligned} \tag{7.3}$$

$$\begin{aligned} persists(T_{ef}, P, T) \leftarrow\ & tempo(T_{ef}), \\ & tempo(T), \\ & not\ clipped(T_{ef}, P, T). \end{aligned} \tag{7.4}$$

$$
\begin{aligned}
clipped(T_{ef}, P, T) \leftarrow\ & evento(C), && (7.5)\\
& tempo(T_{ci}),\\
& tempo(T_{cf}),\\
& happens(C, T_{ci}, T_{cf}),\\
& terminates(C, T_C, P),\\
& not\ out(T_C, T_{ef}, T).\\
out(T_C, T_P, T) \leftarrow\ & T \leq T_C. && (7.6)\\
out(T_C, T_P, T) \leftarrow\ & T_C < T_P. && (7.7)
\end{aligned}
$$

Nas alterações propostas é de salientar que, embora os parâmetros do predicado *persists* já se encontrem instanciados e não existam variáveis locais, optou-se por garantir a instanciação das variáveis, de modo a permitir a utilização deste predicado em qualquer situação. Assumiu-se, no entanto, que o fluente P se encontra instanciado (devido ao predicado *initiates* utilizado na definição de *holds_at*). Já em relação ao predicado $out/3$ optou-se por não efectuar nenhuma alteração dado ser um predicado auxiliar ao predicado $clipped/3$.

Os predicados $<$ e $\leq$ foram definidos com recurso à definição dos predicados correspondentes em Prolog sobre os inteiros. De facto, por questões de simplificação da implementação, considerou-se que os instantes de tempo seriam identificados por números inteiros.

Os axiomas temporais, com as alterações descritas, permitem a inferência das propriedades que são válidas em cada instante de tempo. Para tal, poderá ser efectuada a abdução de eventos que iniciem as propriedades a provar. No caso de existir um modelo contraditório, serão efectuados os procedimentos de revisão descritos no capítulo 4 e 5.

As restrições de integridade existentes sobre os predicados definidos para o Cálculo de Eventos (regras 2.6 a 2.15) ou são admissíveis ou podem ser transformadas de modo a não possuirem literais por omissão não instanciados (utilizando a mesma técnica descrita para o predicado $holds\_at/2$).

7.4 Tradução da Linguagem A_{EC}

No capítulo 2 foi apresentado o processo de tradução da linguagem de acções proposta nesta tese A_{EC} em regras de programação em lógica estendida.

De acordo com o apresentado nesse capítulo, o processo de tradução tem de ser reformulado, de modo a garantir as condições desejadas para o programa em lógica.

Assim, as regras são as seguintes:

- Proposições-e

As proposições-e do tipo:

$A\ causes\ F\ if\ P_1, ..., P_n$, onde $A = A_1 \| ... \| A_m$

são traduzidas para as seguintes regras (assumiu-se que os argumentos poderiam não ser constantes):

$$\begin{aligned}
enabled(E, T_i) \ \leftarrow\ & evento(E), \\
& tempo(T_i), \\
& accao(A_1), ..., accao(A_m), \\
& act(E, A_1), ..., \ act(E, A_m), \\
& holds_at(P_1, T_i), \ldots, holds_at(P_n, T_i). \\
initiates(E, T_f, F) \ \leftarrow\ & evento(E), \\
& tempo(T_i), \\
& tempo(T_f), \\
& T_i \leq T_f, \\
& accao(A_1), ..., accao(A_m), \\
& happens(E, T_i, T_f), \\
& act(E, A_1), \ ..., \ act(E, A_m), \\
& holds_at(P_1, T_i), \ldots, holds_at(P_n, T_i).
\end{aligned}$$

Este processo de tradução garante a instanciação dos literais passíveis de serem abduzidos: $happens/3$ e $act/2$.

Note-se ainda que, conforme referido no capítulo 2, no caso do efeito ser negativo, isto é,

$$A\ causes\ \neg F\ if\ P_1, ..., P_n$$

então a segunda regra será relativa ao predicado $terminates/2$.

Se a acção não tiver efeitos, as proposições-e poderão ser escritas:

$$A\ if\ P_1, \ldots, P_n$$

e traduzidas somente na primeira regra apresentada.

- Proposições-p

As proposições-p do tipo:

$E_1(T_i, T_f), ..., E_n(T_i, T_f)\ precedes\ E'_1(T'_i, T'_f), ..., E'_m(T'_i, T'_f)$

são traduzidas para a seguinte restrição de integridade (assumiu-se que os argumentos poderiam não ser constantes):

$$\begin{aligned}\leftarrow\ & evento(E_1), ..., evento(E_n),\\ & tempo(T_i), tempo(T_f),\\ & evento(E'_1), ..., evento(E'_m),\\ & tempo(T'_i), tempo(T'_f),\\ & T_f < T'_i,\\ & happens(E_1, T_i, T_f), ..., happens(E_n, T_i, T_f),\\ & happens(E'_1, T'_i, T'_f), ..., happens(E'_m, T'_i, T'_f).\end{aligned}$$

Estas restrição garante a ordenação desejada para os eventos.

Note-se que o predicado $happens/3$ não deverá poder abduzir novos eventos que, eventualmente, violem a própria restrição de integridade. Para tal, definiu-se um predicado ($call/1$) que não permite a abdução de novos factos (somente verifica a sua validade).

A restrição de integridade anterior, será, então:

$$\begin{aligned}\leftarrow\ & evento(E_1), ..., evento(E_n),\\ & tempo(T_i), tempo(T_f),\\ & evento(E'_1), ..., evento(E'_m),\\ & tempo(T'_i), tempo(T'_f),\\ & T_f < T'_i,\\ & call(happens(E_1, T_i, T_f)), ..., call(happens(E_n, T_i, T_f)),\\ & call(happens(E'_1, T'_i, T'_f)), ..., call(happens(E'_m, T'_i, T'_f)).\end{aligned}$$

- Proposições-o

As proposições-o do tipo:

$A\ occurs\_in\ E(T_i, T_f)$

onde $A = A_1||...||A_n$ são traduzidas para as seguintes restrições de integridade (o evento e os instantes de tempo poderão não ser constantes):

$$\begin{aligned}\leftarrow\ & evento(E),\\ & tempo(T_i), tempo(T_f),\\ & not\ happens(E, T_i, T_f).\\ \leftarrow\ & evento(E),\\ & tempo(T_i), tempo(T_f),\\ & happens(E, T_i, T_f),\\ & not(act\_ev(E)).\\ act\_ev(E)\ \leftarrow\ & act(E, A_1), ..., act(E, A_n).\end{aligned}$$

Estas restrições significam que o evento $happens(E, T_i, T_f)$ existe e está associado a uma expressão de acções A.

- Proposições-v

As proposições-v do tipo:

$$F\ after\ E_1(T_{1i}, T_{1f}), ..., E_n(T_{ni}, T_{nf})$$

são traduzidas para as seguintes regras (assumiu-se que nas proposições-v os argumentos seriam constantes):

$$\begin{aligned} initiates(E_n, T_{nf}, F) \leftarrow\ & T_{1i} \leq T_{1f}, ..., T_{ni} \leq T_{nf}, \\ & happens(E_1, T_{1i}, T_{1f}), \\ & ..., \\ & happens(E_n, T_{ni}, T_{nf}), \\ & T_{1f} \leq ... \leq T_{nf}. \end{aligned}$$

ou, então, no caso das proposições do tipo:

$$\neg F\ after\ E_1(T_{1i}, T_{1f}), ..., E_n(T_{ni}, T_{nf})$$

serão traduzidas para regras:

$$\begin{aligned} terminates(E_n, T_{nf}, F) \leftarrow\ & T_{1i} \leq T_{1f}, ..., T_{ni} \leq T_{nf}, \\ & happens(E_1, T_{1i}, T_{1f}), \\ & ..., \\ & happens(E_n, T_{ni}, T_{nf}), \\ & T_{1f} \leq ... \leq T_{nf}. \end{aligned}$$

Esta regra define o início/término de propriedades após determinados eventos.

Utilizando as transformações descritas nesta secção é possível obter programas em lógica estendida que não possuem literais por omissão não completamente instanciados.

O processo de tradução foi construído através do recurso à elaboração de um analisador lexical (utilizando o programa LEX-Lexical analyser) e de um analisador gramatical (utilizando o programa YACC-Yet Another Compiler Compiler).

Este processo de tradução permite que o processo de tradução seja completamente automatizado, a partir de um ficheiro de descrição de acções elaborado na linguagem A_{EC}.

7.5 Actos de Fala

A descrição dos diversos actos de fala, em termos de regras de programação em lógica estendida com negação explícita, também necessita ser alterada, de acordo com os procedimentos descritos anteriormente.

Dado que o processo de transformação é o indicado na secção anterior, apresento somente o resultado dessa tradução para o acto de fala $inform$ para um agente receptor Y totalmente crédulo :

$$\begin{aligned} enabled(E,T_i) \leftarrow\ & evento(E), tempo(T_i), \\ & accao(inform(X,Y,P)), \\ & act(E,inform(X,Y,P)), \\ & eu(Y). \\ initiates(E,T_f,bel(Y,bel(X,P))) \leftarrow\ & evento(E), \\ & tempo(T_i), tempo(T_f), \\ & T_i \leq T_f, \\ & accao(inform(X,Y,P)), \\ & happens(E,T_i,T_f), \\ & act(E,inform(X,Y,P)), \\ & eu(Y). \end{aligned}$$

O predicado $accao/1$ deverá validar os agentes dessa acção:

$$\begin{aligned} accao(inform(X,Y,P)) \leftarrow\ & agente(X), \\ & agente(Y). \end{aligned}$$

O predicado $agente/1$ deverá ser definido por:

$$\begin{aligned} agente(X) &\leftarrow eu(X). \\ agente(X) &\leftarrow voce(X). \end{aligned}$$

onde $eu/1$ e $voce/1$ são factos do programa em lógica estendida que modela o agente (pertencem ao parâmetro CM do modelo).

7.6 Operadores Epistémicos

Em relação aos operadores epistémicos (*int*, *bel*, *ach*), foi proposto no capítulo 3 um conjunto de regras e de restrições de integridade que pretendem definir as relações existentes entre os diversos operadores.

Estas regras podem ser transformadas, de um modo análogo ao descrito nas secções anteriores, com o objectivo de as impedir de possuir literais por omissão não instanciados em tempo de execução.

Apresento somente o resultado final da transformação de uma das regras, devido à sua extensão e semelhança com os procedimentos utilizados nas secções anteriores.

Assim, a restrição que relaciona as crenças com os objectivos

$$\Leftarrow holds\_at(bel(X,P),T), holds\_at(ach(X,P),T).$$

será transformada em:

$$\begin{aligned} \leftarrow \; & tempo(T), \\ & holds\_at(bel(X,P),T), holds\_at(ach(X,P),T). \end{aligned}$$

De notar que não é necessário garantir a instanciação do primeiro argumento de $holds\_at/2$ porque o modo como está definido o predicado $holds\_at/2$ garante a sua não utilização como literal por omissão.

7.7 Modelação de Agentes

De modo a efectuar a modelação dos agentes e de acordo com o capítulo 3, é necessário representar, além das regras apresentadas nas secções anteriores, as regras de racionalidade e as regras de comportamento.

De facto, o tuplo que define o modelo do agente $< At, Rc, Ac, T, Rr, CM >$ é constituído por:

1. At — atitudes do agente;
2. Rc — regras de comportamento;
3. Ac — acções do domínio e actos de fala;
4. T — axiomas temporais;
5. Rr — regras de racionalidade;
6. CM — conhecimento do mundo.

As atitudes do agente, as acções do domínio e o conhecimento do mundo são específicos do domínio a modelar, pelo que não existem regras gerais a construir. Como exemplo, temos os factos que identificam os agentes interlocutores de um diálogo — João e Maria:

$$\begin{aligned} & eu(joao). \\ & voce(maria). \end{aligned}$$

A construção dos actos de fala e dos axiomas temporais já foi analisada em secções anteriores (secção 7.5 e 7.3, respectivamente).

As regras de racionalidade foram já apresentadas no capítulo 3 e permitem definir as relações entre as diversas atitudes. Estas regras deverão ser transformadas de um modo análogo ao descrito nas secções anteriores, de modo a garantir que têm termos limitados e não possuem literais por omissão não completamente instanciados.

7.7.1 Regras de Comportamento

As regras de comportamento também foram apresentadas no capítulo 3 e, por exemplo, as regras de transferência de crenças para agentes crédulos (3.16 e 3.17) passam a ter a seguinte forma (omitindo o terceiro argumento necessário para garantir que os termos são limitados):

$$
\begin{aligned}
holds\_at(bel(H,P),T) \leftarrow\ & agente(H), && (7.8)\\
& agente(S),\\
& H \neq S, && (7.9)\\
& tempo(T),\\
& holds\_at(bel(H,bel(S,P)),T),\\
& holds\_at(transf(S,H,P),T).\\
holds\_at(transf(S,H,P),T) \leftarrow\ & agente(H), && (7.10)\\
& agente(S),\\
& H \neq S, && (7.11)\\
& tempo(T),\\
& holds\_at(bel(H,ingenuo(H)),T).
\end{aligned}
$$

7.8 Participação em Diálogos

O processo de participação em diálogos, conforme proposto no capítulo 5, implica a execução das seguintes fases por parte de um agente, após o reconhecimento de um acto de fala de outro agente:

1. Actualização e, eventual revisão do modelo obtido, de modo a remover possíveis contradições;

2. Planeamento de acções (escolhendo a solução abdutiva preferida);

3. Geração dos actos de fala e de outros actos;

Para cada um destes pontos foram criados procedimentos para a sua execução. A execução final das acções deverá estar a cargo de subsistemas especializados na geração de frases em língua natural e na execução de outro tipo de acções.

De seguida, iremos analisar os procedimentos desenvolvidos e as complexidades envolvidas na sua execução.

A actualização e, eventual revisão do estado mental do agente implica o cálculo do modelo revisto do programa em lógica actualizado com a descrição dos novos eventos (capítulo 4):

$$M_1 = solution(actualiza(M, E_1 \times ... \times E_n))$$

O predicado *solution* obtém a revisão a 2 valores, de acordo com a ordem de preferência existente (ver REVISE 2.4 [SDP96]).

O processo de planeamento de acções está dividido nas seguintes fases:

1. Obtenção dos objectivos do agente, $O_a(a,t)$, a partir do modelo anterior M_1. Esta função faz uma pesquisa sobre o modelo do agente M_1 e obtém os objectivos que são suportados nesse modelo.

2. Transformação dos objectivos em restrições de integridade e cálculo dos modelos respectivos:

   $$M_2 = solution(M_1 \cup \{\leftarrow not\ holds\_at(Obj, t_\infty)\})$$

   Este processo implica a criação de um modelo hipotético e um novo processo de revisão, seleccionado a solução preferida.

3. Obtenção das acções abduzidas, $Ab_a(t)$, pelo processo anterior. Esta função é linear com o número de literais do modelo.

4. Actualização do programa em lógica com as intenções de realizar as acções abduzidas. Esta actualização tem uma complexidade linear com o número de acções abduzidas.

Finalmente, o processo de geração dos actos de fala pode ser decomposto em duas fases:

1. Validação de pré-condições das acções;

2. Execução do acto de fala e actualização do modelo do agente.

A primeira fase implica o recurso ao procedimento de prova para validar as pré-condições dos actos de fala. A segunda fase tem como complexidade um factor constante, dependente somente da adição dos eventos gerados ao modelo do agente.

A sua descrição será (assumindo que e_i e α_i representam o i-ésimo evento/acção da lista temporalmente ordenada de acções abduzidas $A_a(t)$):

proc geração$(A_a(t)) \equiv$

1. $M_1 = \emptyset$

2. $for\ j = 1\ to\ \#A_a(t)$

3. $\quad if\ PCFXSM(M_a(t)) \models possible(e_j, t)\ then$

4. $M_1 = M_1 \cup \{act(e_j, \alpha_j), happens(e_j, t_i, t_f), t \leq t_i \leq t_f\}$

5. fi

6. $M_a(t') = M_1$

O procedimento *geração* permite obter as acções que podem ser executadas e permite efectuar a actualização do modelo do agente após a geração dessas acções.

Os procedimentos apresentados nesta secção permitem suportar o processo de participação em diálogos, conforme descrito no capítulo 5. A partir da análise da complexidade destes procedimentos resulta que o factor crítico está relacionado com a necessidade de efectuar diversos cálculos da revisão preferida de um programa em lógica estendida. A complexidade global de participação em diálogos está, portanto, directamente relacionada com a complexidade do cálculo da revisão preferida de modelos que é NP. Esta é uma área que está a ser objecto de trabalho [Alf93, AP96, Dam96] e que se espera venha a ter resultados positivos num futuro próximo, nomeadamente na possibilidade de evitar recálculos durante o processo de revisão de um programa.

No entanto, os procedimentos de participação em diálogos apresentados possuem algumas características próprias que poderão possibilitar a introdução de algumas melhorias. De facto, o programa em lógica, sobre o qual é efectuada a revisão durante o processo de planeamento abdutivo, difere do programa em lógica após a actualização dos eventos somente pelo conjunto de restrições de integridade adicionais. Esta característica sugere que será possível fundir os dois processos de modo a permitir reutilizar cálculos efectuados. Este é, contudo, trabalho que será objecto de uma análise posterior.

7.9 Agentes Distribuídos

Dado que um agente é representado através de um programa em lógica, é possível modelar situações de interacções multi-agente, recorrendo a uma arquitectura distribuída. Neste tipo de arquitectura cada agente será modelado por um processo autónomo e interagirá com os outros agentes via mecanismos de comunicação de mensagens.

Tendo em conta que o sistema de revisão utilizado (REVISE 2.4) está construído sobre Prolog e que existe uma versão do Prolog que permite processos distribuídos — PVM Prolog [MC95]) —, é possível criar uma arquitectura multi-agente distribuída. De facto, o PVM Prolog estende a linguagem Prolog com um interface para uma máquina virtual paralela (PVM [Gei94]). O PVM é uma ambiente de programação paralelo e distribuído que permite a comunicação e a sincronização entre processos sobre uma rede heterogénea de computadores.

Nesta arquitectura, cada agente será modelado através de um predicado de alto nível: $gestao/2$ (ver capítulo 5). Este predicado utiliza os procedimentos de revisão e de prova definidos no sistema REVISE. Cada agente (processo Prolog) comunicará com os outros

agentes (processos) através dos mecanismos de comunicação de mensagens fornecidos pelo PVM Prolog.

Assim, a obtenção de novos eventos (*obter_evento*) é definida com base na primitiva de recepção de mensagens:

$$obter\_evento(Evento) :- pvm\_nrecv(-1, -1, De, Para, Evento).$$

A geração de acções será modelada através do predicado de envio de mensagens:

$$\begin{aligned} gerar\_accoes([Accao|Accoes]) \; :- \; & obter\_id(Accao, To), \\ & pvm\_send(To, 1, Accao), \\ & gerar\_accoes(Accoes). \end{aligned}$$

O predicado $obter\_id/2$ obtém o identificador do agente (processo) destinatário da acção.

7.10 Resultados Experimentais

Nesta secção são apresentados alguns resultados obtidos pela aplicação do protótipo, construído de acordo com o apresentado neste capítulo, a domínios específicos de aplicação.

Em concreto, são apresentados resultados dos exemplos apresentado no capítulo 6 de um diálogo sobre comboios e de um diálogo entre agentes autónomos com diferentes características.

Para tal, foram redefinidas as regras apresentadas anteriormente no capítulo 6, de acordo com a metodologia agora proposta.

O *hardware* utilizado foi um computador PC Pentium a 133 MHz, com o SICSTUS Prolog 3 e o sistema REVISE 2.4 ([SDP96]).

A análise dos tempos de processamento foi dividida em vários pontos:

1. Actualização e revisão do novo modelo do sistema;
2. Cálculo do plano do sistema;
3. Geração dos actos de fala.

Para cada um destes pontos foi recolhido o tempo de processamento (a unidade utilizada foi o segundo).

O primeiro exemplo focava essencialmente questões relacionadas com planeamento abdutivo de acções, enquanto o segundo exemplo pretendia demonstrar a capacidade do sistema em lidar com informações não correctas.

Os resultados obtido para o primeiro exemplo dos comboios foram os seguintes:

A primeira frase — "*Para o Porto*!" — implica um processo de inferência dos objectivos do passageiro e, posteriormente, o cálculo do plano adequado para os satisfazer.

Frase	Inferência do modelo	Cálculo do plano	Geração dos actos de fala
1	6.20	8.21	0.3
2	6.32	8.43	0.3

Tabela 7.1: Resultados em segundos do primeiro exemplo dos comboios

A segunda frase — "*Queria que a hora de partida fosse às 15:00*" — implica um processo idêntico. Existe, no entanto, um número superior de regras no programa em lógica estendida, pelo que os tempos de processamento são ligeiramente superiores.

Para o segundo exemplo dos comboios, obtiveram-se os resultado da tabela seguinte.

Frase	Inferência do modelo	Cálculo do plano	Geração dos actos de fala
1	6.10	7.21	0.4
2	6.14	7.25	0.3

Tabela 7.2: Resultados em segundos do segundo exemplo dos comboios

Neste exemplo, o processo é idêntico ao anterior, pelo que tanto a primeira frase — "*O comboio das 8:30?*" —, como a segunda — "*Não, o que chega de Leiria.*" —, apresentam tempos semelhantes aos tempos do primeiro exemplo. Os tempos de planeamento do primeiro exemplo são ligeiramente superiores aos do segundo exemplo, dado ser necessário abduzir um conjunto de acções superior.

Relativamente ao exemplo dos agentes com diferentes características, obtiveram-se os seguintes tempos de planeamento para o agente Obélix após as frases de Mentirix e Astérix (exemplo do capítulo 6):

Agente	Inferência do modelo	Cálculo do plano	Geração dos actos de fala
Mentirix	4.52	6.41	0.04
Astérix	4.70	6.63	0.15

Tabela 7.3: Resultados em segundos do exemplo do Obélix (cap. 6)

Utilizando o exemplo do capítulo 3 (Obélix e Astérix), obtiveram-se os resultados apresnetados na tabela seguinte.

Os valores obtidos nos quatro exemplos demonstram que o grande peso computacional do processo de participação em diálogos proposto nesta aplicação é resultante do cálculo da revisão preferida dos programas em lógica estendida.

Efectivamente, este procedimento é executado nos dois primeiros passos do processamento (inferência do modelo e cálculo do plano) e não é executado na geração dos actos

Cenário	Inferência do modelo
1	7.3
2	8.1
3	9.2

Tabela 7.4: Resultados em segundos do exemplo do Obélix (cap. 3)

de fala (é somente verificado se determinados predicados são válidos no modelo). Devido a este facto, resulta a diferença de tempos registada para a terceira fase (geração). Note-se, no entanto, que o processo de geração utilizado é bastante simplificado e que não efectua a geração de frases em língua natural.

Dado que os procedimentos de prova SLX e o cálculo da revisão de um programa em lógica estendida estão a ser objecto de uma contínua investigação, é de esperar que, num futuro próximo, seja possível reduzir os tempos obtidos. Há, no entanto, limitações de princípio relativas ao crescimento do tempo com o tamanho dos problemas a tratar que, de algum modo, limitam o âmbito dos domínios a modelar.

7.11 Conclusões

Neste capítulo foi apresentada uma construção possível do sistema de participação em diálogos proposto nesta tese.

Esta construção é baseada num sistema de programação em lógica estendida com semântica bem fundada e com a revisão de predicados para eliminar contradições, a partir do trabalho de Alferes, Damásio e Pereira [Alf93, DNP94, SDP96, AP96].

As regras definidas anteriormente foram alteradas de modo a garantir que o programa em lógica que modela os agentes tenha as características necessárias para que o procedimento de prova existente termine, seja coerente e completo.

Sobre os programas em lógica que modelam os agentes em cada instante de tempo foram definidos procedimentos que permitem a criação de novas atitudes e a geração de planos e de actos de fala. A complexidade do procedimento global de participação em diálogos é definida pela complexidade do cálculo do modelo revisto preferido dos programas em lógica que modelam os agentes (NP). A complexidade do procedimento global poderá vir a ser melhorada, através de uma melhor reutilização de resultados.

A interacção entre os agentes pode ser modelada através de um sistema distribuído (PVM Prolog), onde cada agente é um processo autónomo.

Finalmente, foram apresentados resultados experimentais da aplicação do protótipo definido aos exemplos do capítulo 6.

Em suma, a abordagem proposta nesta tese permitiu a construção de um prototótipo de um sistema gestor de participação em diálogos que possui uma semântica bem definida

e que possui procedimentos de prova completos e coerentes.

Capítulo 8

Conclusões

Neste capítulo faço uma análise das principais características do sistema de participação em diálogos que é proposto nesta tese. São descritas as inovações propostas, realçando o tipo de problemas que permitem solucionar.

Apresento, ainda, alguns dos problemas em aberto existentes na teoria e aponto algumas metodologias a aplicar, de modo a permitir a sua resolução futura.

8.1 Resultados Obtidos

O sistema de participação em diálogos que proponho nesta tese possui as seguintes características fundamentais:

- É construído recorrendo a um ambiente de programação em lógica, estendido com a negação explícita, com semântica bem fundada e com remoção de contradições.

  O recurso a um ambiente de programação em lógica com a semântica bem fundada e com remoção de contradições ([AP96]) permite que o sistema proposto tenha um procedimento de prova completo e coerente (SLX). Esta característica marca uma diferença fundamental em relação aos sistemas mais tradicionais ([Lit85, Car85, Pol86]) que utilizam processos de inferência sem uma semântica bem definida.

- Utiliza uma versão alterada do Cálculo de Eventos, para suportar a representação temporal e o raciocínio sobre os eventos que ocorrem nos diálogos.

  A versão proposta do Cálculo de Eventos permite a representação de acções não-instantâneas e concorrentes. A capacidade para representar e raciocinar sobre os eventos que sucederam no tempo é também um característica inovadora relativamente aos sistema de participação em diálogos referidos ([Lit85, Car85, Pol86]) que contemplavam, somente, a capacidade de efectuar inferências em determinados instantes de tempo, não representando explicitamente o aspecto temporal dos actos de fala e das atitudes. Esta limitação impede esses sistemas de possuir memória sobre os eventos e sobre as atitudes dos agentes. Pelo contrário, o sistema proposto permite efectuar cortes transversais no tempo e obter o modelo mental do agente nesses instantes.

- Utiliza uma linguagem de representação de acções que, além de permitir uma tradução coerente e completa para regras de programação em lógica, permite efectuar estudos comparativos com outros sistemas de representação de acções (por exemplo, Li e Pereira [LP96a].

  A linguagem proposta A_{EC} extende a linguagem proposta inicialmente por Gelfond e Lifschitz (linguagem A [GL92a]) de modo a suportar acções não-instantâneas e concorrentes. Foi demonstrado que a linguagem A é um subconjunto desta linguagem, e que existe uma tradução coerente e completa para regras de programação em lógica. Estas duas características permitem comparar os resultados obtidos com trabalhos já existentes. De facto, foram apresentados exemplos que permitem concluir pela capacidade do sistema proposto de lidar com situações típicas da teoria de acções ([Lif93]). Também esta característica representa uma capacidade adicional em relação aos sistemas existentes de participação em diálogos em que as acções são representadas por estruturas que permitem o processo de planeamento (pré-condições, efeitos), mas que não permitem a inferência de estados (mentais ou

não) nos diversos instantes de tempo que ocorrem antes, durante e após as referidas acções.

- Define e integra no ambiente de programação em lógica os operadores epistémicos (crenças, intenções e objectivos) necessários à modelação do estado mental dos agentes.

  Esta característica permite a integração dos operadores epistémicos, necessários à representação das atitudes, num ambiente formal de programação em lógica. Deste modo, o processo de inferência das atitudes é um processo semelhante ao processo de inferência de qualquer facto, e é efectuado através de procedimentos de prova genéricos. A utilização de operadores epistémicos como factor fundamental no processo de participação em diálogos foi iniciado por Pollack ([Pol86]). Posteriormente, este trabalho tem vindo a sofrer alterações, nomeadamente através do trabalho conjunto de Appelt e Pollack ([AP92]) que propõem um ambiente de abdução ponderada, de modo a suportar o raciocínio. A principal vantagem do sistema que proponho é a integração num ambiente de programação em lógica que possui procedimentos de prova de complexidade polinomial e que permite a construção de protótipos com as características desejadas.

- Define, na linguagem de representação de acções proposta, os diversos actos de fala passíveis de serem realizados durante os diálogos.

  Os actos de fala analisados ([CP79, Tra94]) foram definidos tendo em conta a linguagem de representação de acções, os operadores epistémicos existentes e os diversos modelos de agentes. A partir desta definição obteve-se um sistema que permite a realização de inferências sobre os estados mentais subjacentes aos actos de fala. Esta é, também, uma característica que estende o trabalho de Pollack sobre estados mentais, ao permitir o relacionamento destes com os actos de fala através de um processo formal de prova. Por outro lado, a definição dos efeitos e das pré-condições dos actos de fala, tendo em conta o modelo dos agentes, é uma característica inovadora em sistema de diálogos, e que permite a modelação de ambientes com agentes com comportamentos distintos.

- Propõe a modelação dos agentes através de um tuplo de regras de programação em lógica que permite a representação de diversas propriedades:

  1. Sinceridade (verdadeiro vs mentiroso)
  2. Credulidade (ingénuo vs crédulo vs vs céptico)
  3. Cooperatividade (cooperativo vs não cooperativo)
  4. Tipo de actividade (pró-activo vs reactivo)

  A modelação dos agentes proposta neste trabalho representa o estado mental dos agentes através de:

1. Informação sobre as suas atitudes;
2. Regras que definem a sua racionalidade e o seu comportamento;
3. O seu conhecimento do mundo exterior.

Este processo permite modelar diversos tipos de comportamento e simular situações distintas em ambientes multiagente de diferentes características. Os sistemas de participação em diálogos existentes modelam somente um tipo de agente, nomeadamente os agentes cooperativos, verdadeiros, crédulos e reactivos, e não permitem alterar o seu modelo ([LA87, Car88, Pol90, Fer95, TH92, Tra94, Per90, CL95]). De facto, não possuem a modelação do agente separada do processo de inferência; as regras de comportamento estão incluídas no processo de inferência. Com o sistema proposto nesta tese, é possível modelar modos distintos de participação em diálogos através da simples substituição do modelo de um agente. A adição de novos modelos de comportamento, nomeadamente, propriedades como a agressividade, o não auto-prejuízo (benefício), etc., podem permitir modelar situações com um maior grau de complexidade.

- Suporta a inferência (dedutiva ou abdutiva) das atitudes do agente a modelar, com base no seu estado mental e nos novos eventos associados às diversas acções, nomeadamente, aos actos de fala que vão ocorrendo ao longo de uma conversa em que participa intencionalmente.

  O processo de inferência das atitudes pode ser vista como um processo geral de actualização de atitudes, dado que, após cada acto de fala, atitudes anteriores podem ser terminadas e novas são criadas. A função de actualização e de revisão apresentada baseia-se no cálculo do modelo bem fundado do programa em lógica que representa o estado mental do agente após os actos de fala. O processo de revisão de atitudes é uma característica que não é suportada pelos sistemas de diálogos existentes. De facto, a actualização do estado mental pode provocar a necessidade de revisão desse mesmo estado, dado ser contraditório. O processo de revisão é efectuado recorrendo à semântica de remoção de contradições ([AP96]). Os sistemas de diálogos existentes não consideram a necessidade de efectuar revisões ao estado mental do agente, dado não detectarem as contradições ou não suportarem situações em que elas existam.

- Efectua a geração dos planos dos agentes, com base nos seus objectivos e recorrendo à abdução das acções necessárias à sua satisfação.

  Os planos dos agentes são representados pelas intenções que eles possuem de realizar determinadas acções. Esta abordagem permite integrar a abordagem clássica do planeamento, em que os planos são sequências de acções, com a abordagem em que os planos são estados mentais. Deste modo, o sistema proposto congrega as características mais positivas das duas abordagens. A inferência dos planos é,

portanto, efectuada através da abdução das acções que permitiriam atingir os objectivos dos agentes, efectuando a ordenação das soluções não contraditórias obtidas. Este processo de planeamento abdutivo e de escolha de uma solução preferida é inovador no domínio de diálogos e permite que sejam suportados diálogos em que existem situações de contradição, num dado instante de tempo. Com esta abordagem, é possível modelar situações típicas de diálogos como diálogos de clarificação e mudanças de tópico.

- Gera os actos de fala necessários à execução dos planos inferidos.

  O processo de geração de actos de fala transforma as intenções de realizar acções, em eventos dessas acções e actualiza o modelo do agente com a informação correspondente. Note-se que este processo não integra a componente de geração das frases em Língua Natural, que pela sua complexidade deverá ser objecto de análise separada. A proposta desta tese separa os níveis de gestão do diálogo dos níveis de interpretação e geração de acções, permitindo efectuar, de um modo independente, uma análise dos problemas existentes.

Em suma, o sistema proposto integra sob um ambiente de programação em lógica todas as componentes necessárias a uma participação activa e intencional em diálogos:

1. Representação de conhecimento (tempo, eventos, atitudes);
2. Modelação de agentes;
3. Reconhecimento de actos de fala (em interligação com um módulo especializado de interpretação);
4. Inferência de atitudes;
5. Geração de planos;
6. Geração de actos de fala (em interligação com um módulo especializado de geração).

Conforme foi demonstrado ao longo da tese, a abordagem efectuada permite a resolução de problemas clássicos em diálogos, como, por exemplo, diálogos em que existe:

- informação incompleta;
- ambiguidades;
- situações de erro;
- planeamentos incorrectos;
- intenções contraditórias.

Os resultados obtidos permitem a afirmação de que o sistema proposto poderá vir a servir de base a uma arquitectura global de agentes com a capacidade de participar intencionalmente em diálogos.

Por outro lado, a partir da análise às características do sistema proposto, é possível salientar como factores mais inovadores em relação aos sistemas já existentes para participação em diálogos, a capacidade de:

- Integrar sob um ambiente de programação em lógica estendida com a negação explícita e com a semântica bem fundada os diversos componentes necessários à participação em diálogos. Este ambiente possui procedimentos de prova completos e coerentes.

  Conforme referido anteriormente, o processo de integração dos diversos componentes, bem como as alterações efectuadas de modo a possibilitar essa integração, foram factores fundamentais à obtenção de um sistema com a capacidade de intervir de um modo intencional e activo em diálogos. Nomeadamente, foi necessário:

  1. Estender o Cálculo de Eventos de modo a suportar eventos não simultâneos e concorrentes;
  2. Propôr uma nova linguagem de representação de acções e relacioná-la com a programação em lógica e com outras linguagens de representação de acções;
  3. Definir os actos de fala em termos do seu relacionamento com as atitudes dos agentes e do modelo desses agentes;
  4. Propôr um processo de modelação de agentes que permite a representação de diversos tipos de comportamento;

- Efectuar inferências, dedutivas ou abdutivas, e revisões das atitudes dos agentes.

  Neste ponto contempla-se a capacidade de efectuar actualização de atitudes e de criar novas intenções de realizar acções que venham a satisfazer os objectivos dos agentes. Para tal, se necessário, será efectuada a revisão dessas intenções de modo a remover contradições. A introdução dos conceitos de programação em lógica de revisão de literais e de supressão de contradições em sistemas de participação em diálogos é uma das características mais inovadoras do sistema proposto.

- Gerar actos de fala, com base nos planos inferidos.

  Este ponto permite concluir o processo de participação em diálogos, gerando os actos de fala e dando início a novas intervenções de outros agentes. É de salientar que, embora a geração não seja o tópico fundamental deste trabalho, o sistema proposto permite gerar os actos de fala e efectuar a ligação a módulos especializados de geração de frases em Língua Natural. Alguns dos sistemas propostos na literatura que definem uma teoria de atitudes, como por exemplo os de Cohen e Levesque

[CL95], não permitem completar o processo de participação em diálogos e gerar actos de fala.

- Obter resultados experimentais.

  A implementação efectuada sobre o sistema REVISE ([DNP94]) permitiu obter resultados experimentais sobre diálogos. Estes resultados, embora realcem as capacidades do sistema proposto, demonstram a sua dependência dos procedimentos de prova existentes, indicando zonas de trabalho a realizar de modo a optimizar o processo de participação em diálogos. A possibilidade de obter resultados experimentais é uma característica que distingue este sistema de alguns dos sistema propostos ([CL95]) e que permite a sua inserção numa arquitectura global de participação em diálogos, com a criação de protótipos demonstrativos dos resultados obtidos.

8.2 Trabalho Futuro

Ao longo desta tese foram referidos alguns problemas em aberto que poderão vir a ser objecto de trabalho futuro.

Do conjunto de problemas existentes, destaco:

- Complexidade

  A complexidade do sistema proposto é bastante elevada, o que origina problemas na construção de um protótipo que possa suportar situações de diálogo reais, com algum grau de interactividade.

  Conforme foi analisado no capítulo 7, a complexidade do processo é dependente directamente da complexidade dos procedimentos de prova utilizados e, nomeadamente, do cálculo do modelo revisto preferido do programa em lógica. Dado que esta é uma área em que existe bastante investigação a decorrer, particularmente na Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa e em diversas outras universidades, é possível que resultados futuros venham a permitir alterar o sistema proposto, de modo a obter desempenhos mais optimizados. Por outro lado, foi necessário efectuar alterações às regras de programação em lógica propostas inicialmente, de modo a garantir a terminação do procedimento de prova. Algum trabalho adicional poderá ser feito com o objectivo de optimizar as regras criadas.

- Geração/reconhecimento de frases em Língua Natural

  No capítulo 1 foi efectuada uma introdução à problemática de geração e reconhecimento de frases em Língua Natural a partir dos actos de fala. Como trabalho futuro existe a necessidade de efectuar um estudo aprofundado das relações entre as diversas teorias semânticas das Línguas Naturais e a construção de linguagens a cujas

expressões seja atribuída uma semântica precisa [KR93, Rod94, GS86a, BP83]. Em adição, é necessário relacionar essas linguagens com os actos de fala descritos nesta tese.

- Extensão às capacidades de modelação de agentes

  O formalismo proposto para a modelação dos agentes não analisa o processo de efectuar alterações comportamentais ao longo do tempo e dependentes do interlocutor. Assim, é possível modelar um agente com um comportamento cooperativo num dado instante em relação a um outro agente e, posteriormente, alterar o seu comportamento deixando de ser cooperativo para com esse agente. No entanto, não foi analisado o processo que poderá levar a alterações comportamentais. Esta extensão poderá ser efectuada através da definição e inserção de regras que modelem o processo de alteração comportamental.

- Extensão às capacidades de representação de actos de fala

  O formalismo proposto para a representação de actos de fala não permite a modelação de actos de fala em que haja transmissão no mesmo acto de um conjunto elevado de informação. Nomeadamente, no sistema proposto, o acto de fala *inform* tem como terceiro argumento a proposição a informar. De modo a modelar diálogos com situações mais complexas, existe a necessidade de estender as definições apresentadas para que passem a suportar a comunicação de conjunções ou disjunções de proposições, ou mesmo de estruturas mais complexas representativas de discurso [LQR97].

- Extensão à capacidade de modelação do processo de transmissão e correcção de informação entre agentes

  O processo de transmissão de informação entre agentes está restringido à execuação de actos de fala e à actualização do modelo dos agentes com base nesses eventos. Este trabalho poderá ser estendido de modo a modelar a transmissão não-monótona de informação através de uma semântica de actualizações para a programação em lógica. Além disso, a correcção mútua de informação entre agentes poderá ser modelada através da noção de depuração declarativa recíproca aplicada à programação em lógica.

- Extensão à capacidade de representação de atitudes

  Neste trabalho foram focadas essencialmente três atitudes: crenças, objectivos e intenções. Existe a necessidade de estender este trabalho a outras atitudes importantes para modelar o estado mental dos agentes: expectativas e obrigações. A primeira para controlo dos efeitos desejados e a segunda para que haja a possibilidade de forçar determinados comportamentos. Será importante relacionar estas atitudes com as anteriores e com os actos de fala existentes, alterando, se necessário, a sua definição (pré-condições e efeitos).

- Interacção com o mundo exterior

  Embora o sistema suporte a existência de eventos externos que possam vir a influenciar o comportamento dos agentes, é necessário efectuar trabalho adicional sobre o modo como esse eventos alteram o processo de inferência de atitudes e de geração de actos de fala. Um problema possível é a existência de eventos externos que criem situações de conflito com intenções já existentes, ou mesmo com acções que estão a decorrer. Outra área está relacionada com reconhecimento de outras acções sensoriais como, por exemplo, a visão e o olfacto, o tacto e o ouvido.

- Testes com diálogos reais

  Os resultados obtidos com a implementação efectuada não permitem a sua extrapolação para situações com diálogos reais. De facto, foram obtidos sobre diálogos pouco extensos e sobre domínios restritos. De modo a obter resultados mais significativos, será necessário estender o sistema de modo a suportar diálogos sobre domínios mais vastos e em que o número de acções passíveis de serem executadas seja maior. Note-se que a metodologia desenvolvida e proposta é independente da Língua Natural e é aplicável a qualquer língua, pelo que poderão ser construídos sistemas sobre problemas já analisados noutros trabalhos, com o objectivo de comparar os resultados obtidos.

  O processo de aplicação do sistema a um domínio real mais vasto passará, necessariamente, pelas seguintes fases:

  1. Recolha de um conjunto alargado de diálogos tipo;
  2. Análise dos diálogos recolhidos, com vista à obtenção dos conceitos utilizados, nomeadamente entidades envolvidas e acções possíveis de serem executadas;
  3. Definição, de acordo com a linguagem de representação de acções, das acções a modelar;
  4. Representação, através de regras de programação em lógica estendida com a negação explícita, dos conceitos envolvidos e das suas relações;
  5. Definição do processo de representação de conhecimento do domínio (base de conhecimento) e dos *interfaces* entre o sistema de diálogos e as bases de conhecimento;
  6. Definição dos módulos de reconhecimento/geração de frases em Língua Natural.

  Com base no resultado destas fases será possível simular os diálogos recolhidos, utilizando o modelo de agente desejado, e obter, então, resultados mais significativos. Eventualmente, poder-se-á, também, utilizar este trabalho num outro tipo de situações, em que o sistema assiste a eventos entre outros agentes e pemite a representação dessas interacções: o agente/sistema computacional age com um espectador de interacções [AL97, J.G97, Poe97].

- Integração numa arquitectura de gestão de diálogos

  O sistema proposto não é uma arquitectura global de gestão de diálogos. Nomeadamente, não contempla o processo de reconhecimento e geração de frases em Língua Natural. De modo a suportar uma participação em diálogos em toda a sua extensão, este sistema deverá ser integrado numa arquitectura global que suporte estes procedimentos. Lopes ([Lop91]) propõe uma arquitectura em que há um gestor de diálogos que mediante as situações ou o seu estado assim contrata diversos módulos para a execução de determinadas tarefas. Este trabalho foi estendido em [QL92] de modo a integrar um nível de reflexão adicional. Estas arquitecturas tinham como desvantagem a necessidade de recorrer a estruturas de dados tradicionais (árvores, grafos) para representar o processo de interacção. A gestão destas estruturas era efectuada por módulos especializados coordenados por uma gramática de interacções. O sistema que proponho permite ultrapassar este problema, dado associar a componente temporal a cada evento e a cada atitude que é suportada pelo estado mental do agente/sistema computacional. Deste modo, não é necessário utilizar estruturas de dados adicionais para representar a evolução das interacções.

  Como trabalho a realizar, poder-se-ia definir uma arquitectura global de participação em diálogos, com características semelhantes às arquitecturas referidas, sobre um ambiente de programação em lógica estendida com a negação explícita e com a semântica bem fundada. Esta arquitectura teria como grande vantagem a integração dos diversos componentes sobre o mesmo ambiente e a existência de procedimentos de prova completos e coerentes definidos sobre esse ambiente.

Apêndice A

Prova da coerência e completude da tradução da linguagem A_{EC}

Neste apêndice apresento a prova da coerência e completude da tradução apresentada da linguagem A_{EC} para programas em lógica estendida com negação explícita. Em concreto, prova-se que um literal pertence ao modelo de uma descrição do domínio de A_{EC} sse pertencer ao modelo bem fundado do programa em lógica estendida obtido a partir da tradução da descrição desse domínio (assume-se, nesta demonstração, que o modelo não é contraditório).

Na próxima secção apresento a definição de modelos da linguagem A_{EC} (conforme o capítulo 2) e na secções seguintes apresento a prova da coerência e completude da tradução.

A.1 Linguagem A_{EC}

Uma estrutura $\Psi = (\Phi, \sigma_0)$ é um par onde σ_0 é o estado inicial e Φ é um mapeamento do conjunto de pares (E^n, σ) para o conjunto de estados (E^n é um conjunto de eventos e σ é um estado);

Um fluente F é válido num estado σ se $F \in \sigma$; $\neg F$ é válido em σ se $F \notin \sigma$.

$F\ after\ E_1(T_{1i}, T_{1f}), ..., E_m(T_{mi}, T_{mf})$ é verdadeira numa estrutura Ψ se F for válida no estado $\Phi(E_1(T_{1i}, T_{1f}), ..., E_m(T_{mi}, T_{mf}), \sigma_0)$.

Para qualquer conjunto de eventos simultâneos $E_1(T_i, T_f), ..., E_n(T_i, T_f)$ (designados por $E^n(T_i, T_f)$) e para qualquer estado σ, seja $Effects(E^n(T_i\ , T_f), \sigma)$ o conjunto de fluentes F tais que, para cada expressão de acções A e fluentes $P_1, ..., P_m$, a descrição do domínio D contém ambas as proposições:

$$A\ occurs\_in\ E_i(T_i, T_f), onde E_i(T_i, T_f) \in E^n(T_i, T_f);$$
$$A\ causes\ F\ if\ P_1, ..., P_m, sendo P_1, ..., P_m \text{ válidos em } \sigma.$$

e não existe nenhum evento $E_j(T_i, T_f) \in E^n(T_i, T_j)$ tal que:

$$A'\ occurs\_in\ E_j(T_i, T_j)$$
$$A'\ causes\ \neg F\ if\ P'_1, ..., P'_k, sendo P'_1, ..., P'_k \text{ válidos em } \sigma.$$

Uma estrutura $\Psi = (\Phi, \sigma_0)$ é um modelo de D se todas as proposição-v incluídas em D forem verdadeiras em Ψ e se para qualquer proposição de ordem

$$E_1, ..., E_n\ precedes\ E'_1, ..., E'_m$$

de D que seja não redundante, e para todos os fluentes F:

1. Se $F \in Effects(E'^m, \Phi(E^n, \sigma_0))$ então $F \in \Phi(E'^m, \sigma_0)$
2. Se $\neg F \in Effects(E'^m, \Phi(E^n, \sigma_0))$ então $F \notin \Phi(E'^m, \sigma_0)$
3. Se nenhuma das situações anteriores se verificar, então $F \in \Phi(E'^m, \sigma_0)$ sse $F \in \Phi(E^n, \sigma_0)$.

De acordo com a semântica apresentada, uma proposição-v é suportada pela descrição de domínio D se for verdadeira em todos os modelos de D.

A.2 Coerência

Seja D uma descrição do domínio na linguagem A_{EC}, seja P o programa em lógica estendida obtido a partir da tradução de D segundo as regras definidas no capítulo 2 e seja $V : F\ after\ E_1(T_{1i}, T_{1f}), ..., E_m(T_{mi}, T_{mf})$ uma proposição-v. Nestas condições:

$$\forall P, \exists D, \forall V : holds\_at(F, T_{mf}) \in WFM(P) \Rightarrow D \models V$$

Ou seja, dada uma proposição-v $F\ after\ E_1(T_{1i}, T_{1f}), ..., E_m(T_{mi}, T_{mf})$, se a propriedade $holds\_at(F, T_{mf})$ pertence ao modelo bem fundado do programa em lógica estendida resultado da tradução de D (assume-se T_{mf} o maior dos tempos finais de ocorrência dos eventos), então a proposição-v é suportada pela descrição do domínio D.

A prova da completude da transformação é efectuada por redução ao absurdo.

Suponhamos que $holds\_at(F, T_{mf})$ pertence ao modelo de P e que $F \notin \Phi(E_1(T_{1i}, T_{1f}),$ ..., $E_m(T_{mi}, T_{mf}), \sigma_0)$ (ver definição de suporte de proposições-v em D).

Nestas condições existem as seguintes possibilidades:

1. $F \notin \sigma_0$ e não existe uma proposição de ordem $E'_1, ..., E'_n\ precedes\ E_i, ..., E_j, 1 \leq i \leq j \leq m$, em que $F \in Effects(E_i, ..., E_j, \Phi(E'^n, \sigma_0))$;

2. Existe uma proposição de ordem $E'_1, ..., E'_n\ precedes\ E_i, ..., E_j, 1 \leq i \leq j \leq m$, em que $\neg F \in Effects(E_i, ..., E_j, \Phi(E'^n, \sigma_0))$ e não existe nenhum outro conjunto de eventos simultaneos, posteriores a E'_n e anteriores a E_m , que inicie F.

Na primeira situação, de acordo com a definição de $Effects$, não existe nenhum evento com as seguintes características:

$$A\ occurs\_in\ E$$
$$A\ causes\ F\ if\ P_1, ..., P_k$$

No entanto, analisando o processo de tradução proposto, verifica-se que nestas condições não haverá nenhum conjunto de factos com as seguintes características:

$$happens(E, T_i, T_f)$$
$$initiates(E, T_f, F)$$

Note-se que a tradução da regra $A\ occurs\_in\ E$ é o único processo de definir a ocorrência de eventos.

De acordo com o predicado $happens$, não é possível suportar F sem a existência de um evento que tenha iniciado a propriedade, o que contraria a hipótese inicial (F ser válido em $WFM(P)$).

Na segunda situação, existe uma proposição de ordem $E'_1, ..., E'_n\ precedes\ E_i$, ..., $E_j, 1 \leq i \leq j \leq m$, em que $\neg F \in Effects(E_i, ..., E_j, \Phi(E'^n, \sigma_0))$ e não existe nenhum

evento posterior que inicie F. Ou seja, existem as seguintes regras:

$$A\ occurs\_in\ E_l,\ 1 \leq l \leq n$$
$$A\ causes\ \neg F\ if\ P_1, ..., P_k$$

Recorrendo ao processo de tradução, verifica-se que a propriedade F não pode ser válida no instante T_{mf}. De facto, existe uma situação em que o predicado *clipped* é satisfeito, obrigando $holds\_at(F, T_{mf})$ a ser falso, o que também contraria a hipótese inicial.

A.3 Completude

Seja D uma descrição do domínio na linguagem A_{EC}, seja P o programa em lógica estendida obtido a partir da tradução de D segundo as regras definidas no capítulo 2 e seja $V : F\ after\ E_1(T_{1i}, T_{1f}), ..., E_m(T_{mi}, T_{mf})$ uma proposição-v. Nestas condições:

$$\forall D, \exists P, \forall V : D \models V \Rightarrow holds\_at(F, T_{mf}) \in WFM(P)$$

Ou seja, se uma proposição-v $F\ after\ E_1(T_{1i}, T_{1f}), ..., E_m(T_{mi}, T_{mf})$ é suportada pela descrição do domínio D, então $holds\_at(F, T_{mf})$ pertence ao modelo bem fundado do programa em lógica estendida resultado da tradução de D (assume-se T_{mf} o maior dos tempos finais de ocorrência dos eventos).

A prova da completude da transformação é efectuada por redução ao absurdo.

Suponhamos que que $holds\_at(F, T_{mf})$ não pertence ao modelo bem fundado de P e que $F \in \Phi(E_1(T_{1i}, T_{1f}), ..., E_m(T_{mi}, T_{mf}), \sigma_0)$ (ver definição de suporte de proposições-v em D).

Nestas condições, duas situações são possíveis (ver definição de $holds\_at$ no capítulo 2):

1. Não existe nenhum evento que inicie F;
2. Existe um evento que inicia F, mas existe um outro evento posterior que termina F.

Na primeira situação, se não existe nenhum evento que inicie F, então não existe nenhum conjunto de factos, tal que:

$$happens(E, T_i, T_f) \in WFM(P)$$
$$initiates(E, T_f, F) \in WFM(P)$$

De acordo com o processo de tradução apresentado, a propriedade acima implica que não existe nenhuma regra do tipo:

$$A\ occurs\_in\ E(T_i, T_f)$$
$$A\ causes\ F\ if\ P_1, ..., P_n$$

Ou seja, também não existe nenhum conjunto de eventos simultâneos $E_1(T_i, T_f)$,..., $E_n(T_i, T_f)$ (designados por $E^n(T_i, T_f)$) nem nenhum estado σ, tal que

$$F \in Effects(E^n(T_i, T_f), \sigma).$$

Consequentemente, F não pode pertencer ao estado

$$\Phi(E_1(T_{1i}, T_{1f}), ..., E_m(T_{mi}, T_{mf}), \sigma_0).$$

A segunda situação, implica que existe um evento $E'(T'_i, T'_f)$ que termina F num tempo posterior ao final do evento que inicia a propriedade, mas antes de T_{mf}.

Nestas condições, existirá um conjunto de factos, tal que:

$$\begin{aligned}
&happens(E, T_i, T_f) \in WFM(P)\\
&initiates(E, T_f, F) \in WFM(P)\\
&happens(E', T'_i, T'_f) \in WFM(P)\\
&terminates(E', T'_f, F) \in WFM(P)\\
&T_f < T'_f \in WFM(P)\\
&T'_f < T_{mf} \in WFM(P)
\end{aligned}$$

Novamente, de acordo com o processo de tradução apresentado, a propriedade acima implica que existem regras do tipo:

$$\begin{aligned}
&A\ occurs_in\ E(T_i, T_f)\\
&A\ causes\ F\ if\ P_1, ..., P_n\\
&A'\ occurs_in\ E'(T'_i, T'_f)\\
&A'\ causes\ \neg F\ if\ P'_1, ..., P'_n\\
&E(T_i, T_f)\ precedes\ E'(T'_i, T'_f)\\
&E'(T'_i, T'_f)\ precedes\ E_m(T_{mi}, T_{mf})
\end{aligned}$$

Nestas condições, F também não pode pertencer ao estado

$$\Phi(E_1(T_{1i}, T_{1f}), ..., E_m(T_{mi}, T_{mf}), \sigma_0)$$

porque $F \notin Effects(E^n(T_i, T_f), \sigma_0)$, dado que

$$\neg F \in Effects(E'(T'_i, T'_f), \sigma_0)$$

e

$$E'(T'_i, T'_f)\ precedes\ E_m(T_{mi}, T_{mf})$$

Apêndice B

Transformação da linguagem A na linguagem A_{EC}

Neste apêndice demonstro que existe uma transformação da linguagem A proposta por Gelfond e Lifschitz ([GL92a]) na linguagem A_{EC} proposta nesta tese.

A demonstração utiliza a abordagem utilizada por Lifschitz para relacionar a linguagem A e a linguagem AI, que permite eventos que ocorrem em intervalos de tempo ([Lif93]).

É, no entanto, necessário introduzir previamente alguns conceitos:

Definição 68 *Seja D uma descrição de um domínio na linguagem A. Diz-se que um conjunto C não infinito de sequências de acções $A_1, ..., A_m$ é admissível se:*

1. *Para cada proposição-v $F\ after\ A_1, ..., A_n$ de D, a sequência $A_1, ..., A_n$ pertence a C e*

2. *Se $A_1, ..., A_n \in C$ $(n > 0)$, então $A_1, ..., A_{n-1} \in C$.*

Para cada conjunto C admissível, $\tau(D, C)$ representa uma descrição em A_{EC} obtida da seguinte forma:

1. Os nomes dos fluentes e das acções em $\tau(D, C)$ são idênticos aos de D.

2. O conjunto de eventos é o conjunto C. O evento identificado por $A_1, ..., A_m$ representa o evento que se inicia após a execução das acções $A_1, ..., A_{m-1}$ e termina após a execução da acção A_m.

3. As proposições de efeito $(A\ causes\ F\ if\ P_1, ..., P_m)$ de $\tau(D, C)$ são idênticas às de D.

4. As proposições de valor $(F\ after\ A_1, ..., A_m)$ de D são transformadas em proposições de $\tau(D, C)$ do tipo $F\ after\ E$. No entanto, como os eventos são identificados pelas sequências de acções $A_1, ..., A_m$ as proposições são idênticas nas duas linguagens. Além disso, a definição de conjunto admissível garante a existência do evento $A_1, ..., A_m$ em $\tau(D, C)$.

5. As proposições de ocorrência de $\tau(D, C)$ são as seguintes:

$$A_i\ occurs\_in\ A_1, ..., A_i$$

   para todos A_i, $1 \leq i \leq m$.

6. As proposições de precedência de $\tau(D, C)$ são as seguintes:

$$A_1, ..., A_{i-1}\ precedes\ A_1, ..., A_i$$

   para todos os $A_i \in \{A_1, ..., A_m\}$.

Nestas condições, o teorema seguinte é válido:

Teorema 3 *Para cada descrição do domínio D na linguagem A, para cada conjunto admissível C relativamente a D, e para qualquer proposição-v P na linguagem de $\tau(D,C)$, P é válida em $\tau(D,C)$ sse P for válida em D.*

Este teorema garante a equivalência de resultados entre as duas linguagens, relativamente a proposições de valor $(F\ after\ A_1,...,A_m)$..

De modo a provar este teorema, é necessário rever as definições de modelo da linguagem A e da linguagem A_{EC} apresentadas no capítulo 2:

- Linguagem A

  Uma estrutura (σ_0,Φ) é um par onde σ_0 é um estado (conjunto de fluentes) e Φ uma função de transição (mapeamento de (A,σ), onde A é uma acção e σ um estado, no conjunto de estados).

  Uma estrutura (σ_0,Φ) é um modelo de D se todas as proposições-v de D são válidas em (σ_0,Φ) e, para cada acção A, para cada fluente F e para cada estado σ, as seguintes condições se verificarem:

  1. Se D incluir uma proposição-e que descreve F como efeito de A, em que que as pré-condições se verifiquem em σ, então $F \in \Phi(A,\sigma)$
  2. Se D incluir uma proposição-e que descreve $\neg F$ como efeito de A, em que que as pré-condições se verifiquem em σ, então $F \notin \Phi(A,\sigma)$
  3. Se D não incluir proposições-e com estas característica, então $F \in \Phi(A,\sigma)$ sse $F \in \sigma$.

  A função de transição Φ pode ser estendida de (A,σ) para sequências de acções $A_1,...,A_m$ da seguinte forma:

$$\Phi(\epsilon,\sigma) = \sigma$$
$$\Phi(A_1,...A_m,\sigma) = \Phi(A_m,\Phi(A_1,...A_{m-1},\sigma))$$

- Linguagem A_{EC}

  Uma estrutura $\Psi = (\Phi,\sigma_0)$ é um par onde σ_0 é o estado inicial e Φ é um mapeamento do conjunto de pares (E^n,σ) para o conjunto de estados (E^n é um conjunto de eventos e σ é um estado);

  Um fluente F é válido num estado σ se $F \in \sigma$; $\neg F$ é válido em σ se $F \notin \sigma$.

  $F\ after\ E_1(T_{1i},T_{1f}),...,E_m(T_{mi},T_{mf})$ é verdadeira numa estrutura Ψ se F for válida no estado $\Phi(E_1(T_{1i},T_{1f}),...,E_m(T_{mi},T_{mf}),\sigma_0)$.

  Para qualquer conjunto de eventos simultâneos $E_1(T_i,T_f),...,E_n(T_i,T_f)$ (designados por $E^n(T_i,T_f)$) e para qualquer estado σ, seja $Effects(E^n(T_i,T_f),\sigma)$ o conjunto de

fluentes F tal que, para cada expressão de acções A e fluentes $P_1, ..., P_m$, a descrição do domínio D contém ambas as proposições:

$$A\ occurs\_in\ E_i(T_i, T_f), onde E_i(T_i, T_f) \in E^n(T_i, T_f);$$
$$A\ causes\ F\ if\ P_1, ..., P_m, sendo P_1, ..., P_m \text{ válidos em } \sigma.$$

e não existe nenhum evento $E_j(T_i, T_f) \in E^n(T_i, T_j)$ tal que:

$$A'\ occurs\_in\ E_j(T_i, T_j)$$
$$A'\ causes\ \neg F\ if\ P'_1, ..., P'_k, sendo P'_1, ..., P'_k \text{ válidos em } \sigma.$$

Uma estrutura $\Psi = (\Phi, \sigma_0)$ é um modelo de D se todas as proposição-v incluídas em D forem verdadeiras em Ψ e se para qualquer proposição de ordem $E_1, ..., E_n$ *precedes* $E'_1, ..., E'_m$ de D que seja não redundante, e para todos os fluentes F:

1. Se $F \in Effects(E'^m, \Psi(E^n, \sigma_0))$ então $F \in \Psi(E'^m, \sigma_0)$
2. Se $\neg F \in Effects(E'^m, \Psi(E^n, \sigma_0))$ então $F \notin \Psi(E'^m, \sigma_0)$
3. Se nenhuma das situações anteriores se verificar, então $F \in \Psi(E'^m, \sigma_0)$ sse $F \in \Psi(E^n, \sigma_0)$.

De acordo com a semântica apresentada, uma proposição-v é suportada pela descrição de domínio D se for verdadeira em todos os modelos de D.

A demonstração do teorema apresentado anteriormente é efectuada definindo a igualdade entre a função Φ da estrutura Ψ de $\tau(D, C)$ e a função Φ da estrutura de D, através da equação:

$$\Phi_{\tau(D,C)}(A_1, ..., A_m, \sigma_0) = \Phi_D(A_1, ..., A_m, \sigma_0)$$

Note-se que o primeiro argumento de $\Phi_{\tau(D,C)}$ é um evento, enquanto o primeiro argumento de Φ_D é uma sequência de acções.

Devido à definição de $\tau(D, C)$, para qualquer sequência de acções de D existe um evento correspondente (e vice-versa).

Por outro lado, se Φ_D é a função de transição do modelo de D, então $\Phi_{\tau(D,C)}$ é, também a função de transição do modelo de $\tau(D, C)$ (e vice-versa). Isto porque a igualdade entre as funções garante a transferência das suas propriedades.

Bibliografia

[AB91] Anne Anderson and Elizabeth Byle. Forms of introduction in dialogues: Their discourse contexts and communicative consequences. Technical Report HCRC/RP-25, University of Edinburgh, December 1991.

[aCBdS94] Jo ao Carlos Balsa da Silva. Raciocínio abdutivo para correcção e explicação de erros em português. Master's thesis, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa, 1994.

[ADP95] José Júlio Alferes, Carlos Damásio, and Luís Moniz Pereira. A logic programming system for nonmonotonic reasoning. *Journal of Automated Reasoning*, 14:93–147, 1995.

[AF94] James Allen and George Ferguson. Actions and events in interval temporal logic. Technical Report 521, University of Rochester, July 1994.

[AH87] James Allen and Patrick Hayes. Moments and points in an interval-based temporal logic. Technical Report 180, University of Rochester, December 1987.

[AKPT91] Jamea Allen, Henry Kautz, Richard Pelavin, and Josh Tenenberg. *Reasoning about Plans*. Morgan Kaufman Publishers, Inc., 1991.

[AL86] James Allen and Diane Litman. Plans, goals, and language. In *Proceedings of the IEEE*, 1986.

[AL94] Nicholas Asher and Alex Lascarides. Intentions and information in discourse. In *32nd Meeting of the ACL*, 1994.

[AL97] N. Asher and A. Lascarides. Questions in dialogue. In *MunDial'97 – Munich Workshop on Formal Semantics and Pragmatics Od Dialogue*, 1997.

[Alf93] José Júlio Alferes. *Semantics of Logic Programs with Explicit Negation*. PhD thesis, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa, 1993.

[ALP95a] J. J. Alferes, Renwei Li, and Luís Moniz Pereira. Non-atomic actions in the situation calculus. In N. Mamede and C. Pinto-Correia, editors, *Proceedings of*

EPIA'95. Lecture Notes in Artificial Intelligence, volume 990, pages 273–284. Springer-Verlag, 1995.

[ALP95b] José Alferes, Renwei Li, and Luís Moniz Pereira. Concurrent actions and changes in the situation calculus. In *IBERAMIA'95*, 1995.

[AP80] J. F. Allen and C. R. Perrault. Analyzing intention in utterances. *Artificial Intelligence*, (15):143–178, 1980.

[AP92] Douglas E. Appelt and Martha E. Pollack. Weighted abduction for plan ascription. *User Modeling and User-Adapted Interaction*, 2(1), 1992.

[AP96] José Júlio Alferes and Luís Moniz Pereira. *Reasoning with Logic Programming*, volume 1111 of *Lecture Notes in Artificial Intelligence*. Springer, 1996.

[Apa93] Joaquim Nunes Aparício. *Logic Programming: A Tool for Reasoning*. PhD thesis, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa, 1993.

[Arr91] Paul Van Arragon. Modeling default reasoning using defaults. *User Modeling and User-Adapted Interaction*, 1(3):259–288, 1991.

[Ash86] Nicholas Asher. Belief in discourse representation theory. *Journal of Philosophical Logic*, (15):127–189, 1986.

[Ash87] Nicholas Asher. A typology for attitude verbs and their anaphoric properties. *Linguistics and Philosophy*, (10):125–197, 1987.

[Aus62] J. Austin. *How to do things with words*. Oxford University Press, 1962.

[Bak91] Andrew Baker. Nonmonotonic reasoning in the framework of situation calculus. *Artificial Intelligence*, (49):5–23, 1991.

[BB95] C. Boutilier and V. Becher. Abduction as belief revision. *Artificial Intelligence*, 77(1):43–94, 1995.

[BDS94] Kristof Van Belleghem, Marc Denecker, and Danny De Schreye. Representing continuous change in the abductive event calculus. In *International Conference on Logic Programming'94*, 1994.

[BG93] C. Baral and M. Gelfond. Representing concurrent actions in extended logic programming. In *IJCAI*, 1993.

[Bou96] C. Boutilier. Abduction to plausible causes: an event-based model of belief update. *Artificial Intelligence*, 83(1):143–166, 1996.

[BP83] J. Barwise and J. Perry. *Situations and Attitudes*. MIT Press, 1983.

[Bra90] Michael Bratman. *What is Intention?, in Intentions in Communication.* MIT, 1990.

[BW91a] Afzal Ballim and Yorick Wilks. *Artificial Believers: The Ascription of Belief.* Laurence Erlbaum Associates, Inc., 1991.

[BW91b] Afzal Ballim and Yorick Wilks. Beliefs, stereotypes and dynamic agent modeling. *User Modeling and User-Adapted Interaction*, 1(1):33–66, 1991.

[Car85] Sandra Carberry. *Pragmatic Modeling in Information System Interfaces.* PhD thesis, University of Delaware, 1985.

[Car88] Sandra Carberry. Modelling the user's plans and goals. *Computational Linguistics*, 14(3):23–37, 1988.

[Car92] Jean Carletta. Planning to fail, not failing to plan: risk-taking in task-oriented dialogue. In *COLING'92 - International Conference on Computational Linguistics*, 1992.

[CCC95] Jennifer Chu-Carrol and Sandra Carberry. Communication for conflict resolution in multi-agent collaborative planning. In *ICMAS*, pages 49–56, 1995.

[CG94] R. Cooper and J. Ginzburg. A compositional situation semantics for attitude reports. *Language, Logic and Computation*, 1994.

[CGL+93] Alison Cawsey, Julia Galliers, Brian Logan, Steven Reece, and Karen Jones. *Revising Beliefs and Intentions: A Unified Framework for Agent Interaction*, pages 130–139. IOS Press, 1993.

[CL90a] P. Cohen and H. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42(3), 1990.

[CL90b] Philip Cohen and Hector Levesque. *Persistence, Intention, and Commitment, in Intentions in Communication*, pages 33–70. MIT, 1990.

[CL95] Philip Cohen and Hector Levesque. Communicative actions for artificial agents. In *ICMAS*, pages 65–72, 1995.

[CMP90] P. Cohen, J. Morgan, and M. Pollack. *Intentions in Communication.* MIT Press, Cambridge, MA, 1990.

[CMP94] Luca Chittaro, Angelo Montanari, and Alessandro Provetti. Skeptical and credulous event calculi for supporting modal queries. In *11th European Conference on Artificial Intelligence*, 1994.

[Coo93] R. Cooper. Towards a general semantic framework. Technical report, DYANA-2 Deliverable R2.1 A, 1993.

[CP79] Philip Cohen and Raymond Perrault. Elements of a plan-based theory of speech acts. *Cognitive Science*, 3:177–212, 1979.

[CS95] Alessandro Cimatti and Luciano Serafini. Multiagent reasoning with belief contexts: Elaboration tolerance. In *ICMAS*, pages 57–64, 1995.

[CSSB91] Robin Cohen, Fei Song, Bruce Spencer, and Peter Van Beek. Exploiting temporal and novel information from the user in plan recognition. *User Modeling and User-Adapted Interaction*, 1(2):125–148, 1991.

[CT91] Luca Console and Pietro Torasso. A spectrum of logical definitions of model-based diagnosis. *Computational Intelligence*, 7:133–141, 1991.

[CY91] Randall Calistri-Yeh. Utilizing user models to handle ambiguity and misconceptions in robust plan recognition. *User Modeling and User-Adapted Interaction*, 4(4):289–322, 1991.

[Dal88] Robert Dale. *Generating Regerring Expressions in a Domain of Objects and Processes*. PhD thesis, University of Edinburgh, 1988.

[Dam96] Carlos Damásio. *Paraconsistent Extended Logic Programming with Constraints*. PhD thesis, New University of Lisbon, 1996.

[DMB82] Marc Denecker, Lode Missiaen, and Maurice Bruynooghe. Temporal reasoning with abductive event calculus. In *10th European Conference on Artificial Intelligence*, pages 384–388, 1982.

[DMR96] M. Denecker, B. Martens, and L. Raedt. On the difference between abduction and induction: A model theoretic perspective. In *12th European Conference on Artificial Intelligence*, 1996.

[DNP94] Carlos Damásio, Wolfgang Nejdl, and Luís Moniz Pereira. Revise: An extended logic progamming system for revising knowledge bases. In Morgan Kaufmann, editor, *KR'94*, 1994.

[DP94] Aldo Franco Dragoni and Paolo Puliti. Mental states recognition from speech acts through abduction. In *11th European Conference on Artificial Intelligence*, pages 183–187, 1994.

[DS] Marc Denecker and Danny De Schreye. Representing incomplete knowledge in abductive logic programming.

[DS92] Marc Denecker and Danny De Schreye. Sldnfa: an abductive procedure for normal abductive programs. In *Proceedings of the International Joint Conference and Symposium on Logic Programming*, 1992.

[Dun92] Phan Minh Dung. Acyclic disjunctive logic programs with abductive procedure as proof procedure. In *Proceedings of the International Conference on Fifth Generation Computer Systems*, pages 555–561, 1992.

[EC92] Rhonda Eller and Sandra Carberry. A meta-rule approach to flexible plan recognition in dialogue. *User Modeling and User-Adapted Interaction*, 2(1):27–54, 1992.

[EK89] K. Eshghi and R. Kowalski. Abduction compared with negation as failure. In *Proceedings of the 6th International Conference on Logic Programming.* ICOT, 1989.

[Esh88] Kave Eshghi. Abductive planning with event calculus. In *Proceedings of the International Conference on Logic Programming*, 1988.

[EW] Barbara Di Eugenio and Bonnie Webber. Plan recognition in understanding instructions. In *Proceedings of the First International Conference on Artificial Intelligence Planning Systems*, pages 52–61.

[Fer92] George Ferguson. Explicit representation of events, actions and plans for assumption-based plan reasoning. Technical Report 428, University of Rochester, June 1992.

[Fer95] George Ferguson. *Knowledge Representation and Reasoning for Mixed-Initiative Planning.* PhD thesis, University of Rochester, 1995.

[FM90] Steven Feiner and Kathleen McKeown. Coordinating text and graphics in explanation generation. In *Proceedings of the AAAI'90*, pages 442–449, 1990.

[FN71] R. E. Fikes and Nils J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intellligence*, (2):189–208, 1971.

[Fra91] Antônio Frainer. Interfaces inteligentes. Technical Report 214, Universidade Federal do Rio Grande do Sul, Abril 1991.

[FS91] A. Frainer and H. Strogoulski. Comportamento de interfaces inteligentes. CPGCC/UFRGS, Brasil, 1991.

[Gal90] Antony Galton. A critical examination of allen's theoty of action and time. *Artificial Intelligence*, 42:159–188, 1990.

[GAT93] Derek Gross, James Allen, and David Traum. The trains 91 dialogues. Technical Report 92-1, University of Rochester, July 1993.

[GC94] Nancy Green and Sandra Carberry. Conversational implicatures in indirect replies. In *32nd Meeting of the ACL*, 1994.

[Gei94] Al Geist. *PVM: Parallel Virtual Machine*. MIT Press, 1994.

[Gin95] J. Ginzburg. Resolving questions. *Linguistics and Philosophy*, 18(5):459–527, 1995.

[GL88] M. Gelfond and V. Lifshitz. The stable model semantics for logic programming. In *Proceedings of the 5th International Conference On Logic Programming*, 1988.

[GL92a] M. Gelfond and V. Lifshitz. Representing actions in extended logic programs. In *Proceedings of the International Symposium on Logic Programming*, 1992.

[GL92b] Bradley Goodman and Diane Litman. On the interaction between plan recognition and intelligent interfaces. *User Modeling and User-Adapted Interaction*, 2(2):83–116, 1992.

[GL93] Robert Goldman and Raymond Lang. Intentions in time. Technical Report TUTR 93-101, Tulane University, January 1993.

[GM94] Carl Gutwin and Gordon McCalla. Would i lie to you? modelling misrepresentation and context in dialogue. In *32nd Meeting of the ACL*, 1994.

[GS86a] J. Groenendijk and M. Stokhof. *Studies on the Semantics of Questions and the Pragmatics of Answers*. PhD thesis, Centrale Interfaculteit, Amsterdam, 1986.

[GS86b] B. Grosz and C. Sidner. Attention, intentions and the structure of the discourse. *Computational Linguistics*, 12(3), 1986.

[GS90] Barbara Grosz and Candace Sidner. *Plans for Discourse, in Intentions in Communication*, chapter 20, pages 417–444. MIT, 1990.

[Gug94] Alessio Guglielmi. Concurrency and plan generation in a logic programming language with a sequential operator. In *International Conference on Logic Programming*, pages 240–254, 1994.

[HA95] Peter Heeman and James Allen. The trains 93 dialogues. Technical Report 94-2, University of Rochester, March 1995.

[Had95] Afsaneh Haddadi. Towards a pragmatic theory of intentions. In *ICMAS*, pages 133–139, 1995.

[Had96] P. Haddawy. A logic of time, change and action for representing plans. *Artificial Intelligence*, 80(2):243–308, 1996.

[HE80] Jerry Hobbs and David Evans. Conversation as planned behavior. *Cognitive Science*, 4:349–377, 1980.

[Hee94] Peter Heeman. Spoken dialogue understanding and local context. Technical Report 523, University of Rochester, July 1994.

[HH95] Peter Heeman and Graeme Hirst. Collaborating on referring expressions. Technical Report 435, University of Rochester, April 1995.

[HL95] J. Halpern and G. Lakemeyer. Levesque's axiomatization of only knowing is incomplete. *Artificial Intelligence*, 74(2):381–387, 1995.

[HM87] Steve Hanks and Drew McDermott. Nonmonotonic logic and temporal projection. *Artificial Intelligence*, 33:379–412, 1987.

[HMP96] Z. Huang, M. Masuch, and L. Pólos. Alx, an action logic for agents with bounded rationality. *Articial Intelligence*, 82(1-2):75–127, 1996.

[Hob79] Jerry Hobbs. Coherence and coreference. *Cognitive Science*, 3:67–90, 1979.

[HS94] Elizabeth Hinkelman and Stephen Spackman. Communiacting with multiple agents. In *15th International Conference on Computational Linguistics*, 1994.

[HSME88] J. Hobbs, M. Stickel, P. Martin, and D. Edwards. Interpretation as abduction. In *Proceedings of the 26th Annual Meeting of ACL*, 1988.

[Jen95] N. Jennings. Controlling cooperative problem solving in industrial multi-agent systems using joint intentions. *Artificial Intelligence*, 75(2):195–240, 1995.

[J.G97] J.Ginzburg. Semantically-based ellipsis resolution with syntactic presuppositions. In *IWCS II – Second International Workshop on Computational Semantics*, 1997.

[Jon90] Andrew Jones. *Toward a Formal Theory of Communication and Speech Acts, in Intentions in Communication*, chapter 8, pages 141–160. MIT, 1990.

[Kar93] Neelakantan Kartha. Soundness and completeness theorems for three formalizations of action. In *International Joint Conference on Artificial Intelligence*, 1993.

[Kas91] Robert Kass. Building a user model implicitly from a cooperative advisory dialogue. *User Modeling and User-Adapted Interaction*, 1(3):203–258, 1991.

[Kau90] Henry Kautz. *A Circumscriptive Theory of Plan Recognition, in Intentions in Communication*, chapter 6, pages 105–134. MIT, 1990.

[KED91] Hiroaki Kitano and Carol Van Ess-Dykema. Toward a plan-based understanding model for mixed-initiative dialogues. In *29th Annual Meeting of the ACL*, 1991.

[KF88] Robert Kass and Tim Finin. Modeling the user in natural language systems. *Computational Linguistics*, 14(3):5–22, 1988.

[KM90] A. Kakas and P. Mancarella. Generalised stable models: A semantics for abduction. In *Proceedings of the 9th ECAI*, 1990.

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

[KR91] Hans Kamp and Uwe Reyle. A calculus for first order discourse representation structures. Technical Report 16/91, University of Stuttgart, 1991.

[KR93] Hans Kamp and Uwe Reyle. *From Discourse to Logic: An Introduction to Model Theoretic Semantics of Natural Language*. Kluwer, 1993.

[LA87] D. Litman and J. Allen. A plan recognition model for subdialogues in conversations. *Cognitive Science*, (11):163–200, 1987.

[LA90] Diane Litman and James Allen. *Discourse Processing and Commonsense Plans, in Intentions in Communication*, chapter 17, pages 365–388. MIT, 1990.

[LA91] Alex Lascarides and Nicholas Asher. Discourse relations and defeasible knowledge. In *Proceedings of the 29th Anual Meeting of ACL*, pages 55–62, 1991.

[LC] Lynn Lambert and Sandra Carberry. A tripartite plan-based model of dialogue.

[LC92] Lynn Lambert and Sandra Carberry. Using linguistics, world, and contextual knowledge in a plan recognition model of dialogue. In *International Conference on Computational Linguistics*, 1992.

[LC94] Lynn Lambert and Sandra Carberry. Modeling negotiation subdialogues. In *32nd Meeting of the ACL*, 1994.

[Lif] Vladimir Lifschitz. Toward a metatheory of action.

[Lif93] Vladimir Lifschitz. A language for describing actions. 1993.

[Lit85] Diane J. Litman. *Plan Recognition and Discourse Analysis: An Integrated Approach for Understanding Dialogues*. PhD thesis, Dep. of Computer Science, University of Rochester, 1985.

[Loc95] Karen Lochbaum. The use of knowledge preconditions in language processing. In *International Joint Conference on Artificial Intelligence*, 1995.

[Lop86] J. G. Lopes. *Conceptualization of an Automatic Interlocutor.* PhD thesis, I.S.T. Technical University of Lisbon, 1986. in Portuguese.

[Lop91] J. G. Lopes. Architecture for intentional participation of natural language interfaces in conversations. In C. Brown and G. Koch, editors, *Natural Language Understanding and Logic Programming III.* North Holland, 1991.

[LP96a] Renwei Li and Luís Moniz Pereira. Representing and reasoning about concurrent actions with abductive logic programs. *Annals of Mathematics and Artificial Intelligence, Special Issue for Gelfondfest*, 1996.

[LP96b] Renwey Li and Luís Moniz Pereira. Knowledge assimilation in domains of actions: A possible causes approach. *Journal of Applied Nonclassical Logics*, 5(1):31–50, 1996.

[LQR97] J. G. Lopes, P. Quaresma, and I. Rodrigues. A dialogue system for controlling question/answer dialogues. In *Proceedings of DIALOGUE'97 – International Conference on Computational Linguistics and its Applications*, 1997.

[LS92] J. G. Lopes and A. M. Santos. Portuguese lexicon acquisition interface: Plain. In *Euralex'90 Proceedings.* Bibliograph VOX, 1992.

[MA94] Nathaniel Martin and James Allen. Statistical probabilities for planning. Technical Report 474, University of Rochester, 1994.

[May92] James Mayfield. Controlling inference in plan recognition. *User Modeling and User-Adapted Interaction*, 2(1):55–82, 1992.

[MC95] R. Marques and J. Cunha. Using pvm with a logic programming interface. Technical report, FCT, New University of Lisbon, 1995.

[McC88] Kathleen McCoy. Reasoning on a highlighted user model to respond to misconceptions. *Computational Linguistics*, 14(3):52–63, 1988.

[McD78] Drew McDermott. Planning and acting. *Cognitive Science*, 2:71–109, 1978.

[MH93a] Susan McRoy and Graeme Hirst. Abductive explanation of dialogue misunderstandings. In *EACL'93*, 1993.

[MH93b] Susan McRoy and Graeme Hirst. Abductive explanation of dialogue misunderstandings. In *EACL*, pages 277–285, 1993.

[Mil94] Rob Miller. Situation calculus specifications for event calculus logic programs. Technical report, Imperial College of Science, Technology and Medicine, 1994.

[Mil95] Rob Miller. Narratives in the context of temporal reasoning. Technical Report 94/3, Imperial College of Science, Technology and Medicine, January 1995.

[Mis91] Lode Missiaen. *Localized Abductive Planning with the Event Calculus.* PhD thesis, Univ. Leuven, 1991.

[MLC95] Michael Móra, Gabriel Lopes, and Helder Coelho. Modelling intentions with extended logic programming. In Jacques Wainer and Ariadne Carvalho, editors, *Advances in Artificial Intelligence - 12th Brazilian Symposium on Artificial Intelligence*, Campinas, October 1995.

[MP89] Johanna Moore and Cécile Paris. Planning text for advisory dialogues. In *27th Annual Meeting of ACL*, pages 203–211, 1989.

[MP92] Johanna Moore and Cécile Paris. Exploiting user feedback to compensate for the unreliability of user models. *User Modeling and User-Adapted Interaction*, 2(4):287–330, 1992.

[MS95] Rob Miller and Murray Shanahan. Narratives in the situation calculus. *Journal of Logic & Computation*, 4(5), October 1995.

[Mye92] John Myers. B-sure: A believed situation and uncertain-action representation environment. In *International Conference Computational Linguistics*, 1992.

[NA94] Shin'ya Nakajima and James Allen. A study on prosody and discourse structure in cooperative dialogues. Technical report, University of Rochester, 1994.

[PAA91a] Luís Moniz Pereira, José Júlio Alferes, and Joaquim Nunes Aparício. The extended stable models of contradiction removal semantics. In P. Barahona, L. M. Pereira, and A. Porto, editors, *5th Portuguese AI Conference, Volume 541 of LNAI*, pages 105–119. Springer-Verlag, 1991.

[PAA91b] Luís Moniz Pereira, J. N. Aparício, and J. J. Alferes. Nonmonotonic reasoning with well founded semantics. In Koichi Furukawa, editor, *8th Int. Conf. On LP*, pages 475–489. MIT Press, 1991.

[PAA94] Luís Moniz Pereira, José Júlio Alferes, and Joaquim Nunes Aparício. Contradiction removal semantics with explicit negation. In M. Masuch and L. Pólos, editors, *Knowledge Representation and Reasoning Under Uncertainty, Volume 808 of LNAI*, pages 91–106. Springer-Verlag, 1994.

[PDA93a] Luís Moniz Pereira, Carlos Damásio, and José Júlio Alferes. Diagnosis and debugging as contradiction removal. In L. M. Pereira and A. Nerode, editors, *2nd Int. Ws. On LP and NMR*, pages 316–330. MIT Press, 1993.

[PDA93b] Luís Moniz Pereira, Carlos Damásio, and José Júlio Alferes. Diagnosis and debugging as contradiction removal in logic programs. In L. Damas and M. Filgueiras, editors, *6th Portuguese AI Conf. LNAI vol. 727,*. Springer-Verlag, 1993.

[PE92] C. Preist and K. Eshghi. Consistency-based and abductive diagnoses as generalised stable models. In *Proceedings of the International Conference on 5th Generation Computer Systems*, pages 514–521, 1992.

[Ped89] Edwin Pednault. Adl: Exploring the middle ground between strips and the situation calculus. In *Proc. Of the First International Conference on Principles of Knowledge Representations and Reasoning*, pages 324–332, 1989.

[Per90] Raymond Perrault. *An Application of Default Logic to Speech Act Theory, in Intentions in Communication*, chapter 9, pages 161–186. MIT, 1990.

[Poe97] M. Poesio. The dynamics of discourse situations. In *MunDial'97 – Munich Workshop on Formal Semantics and Pragmatics of Dialogue*, 1997.

[Pol86] Martha E. Pollack. *Inferring Domain Plans in Question-Answering.* PhD thesis, Dep. of Computer and Information Science, University of Pennsylvania, 1986.

[Pol90] Martha Pollack. *Plans as Complex Mental Attitudes, in Intentions in Communication*, chapter 5, pages 77–104. MIT, 1990.

[Prz] Teodor Przymusinski. Static semantics for normal and disjunctive logic programs.

[QDF88] Alex Quilici, Michael Dyer, and Margot Flowers. Recognizing and responding to plan-oriented misconceptions. *Computational Linguistics*, 14(3):38–51, 1988.

[QL92] P. Quaresma and J. G. Lopes. A two-headed architecture for intelligent multimedia man-machine interaction. In *B. de Boulay and V. Sgurev (eds). Artificial Intelligence V - methodology, systems, applications.* North Holland, 1992.

[QL93a] P. Quaresma and J. G. Lopes. Abduction of plans and intentions in dialogues. In P. Codognet, P. M. Dung, and A. C. Kakas, editors, *Proceedings of the Workshop on Abductive Reasoning, International Conference on Logic Programming*, 1993.

[QL93b] P. Quaresma and J. G. Lopes. Reconhecimento de intenções para uma interacção robusta com bases de conhecimento médicas. In *EPLP'93 - Encontro Português da Língua Portuguesa*, 1993.

[QL94] Paulo Quaresma and José Gabriel Lopes. A logic programming framework for the abductive inference of intentions in cooperative dialogues. In Frank Pfenning, editor, *5th Internatinal Conference on Logic Programming and Automated Reasoning, Lecture Notes in Artificial Intelligence 822*, pages 189–199. Springer Verlag, 1994.

[QL95] Paulo Quaresma and José Gabriel Lopes. Unified logic programming approach to the abduction of plans and intentions in information-seeking dialogues. *Journal of Logic Programming*, (54), 1995.

[Qui91] Alexander Quilici. *The Correction Machine: A Computer Model of Recognizing and Producing Belief Justifications in Argumentative Dialogues.* PhD thesis, University of California, 1991.

[Qui92] Alex Quilici. Arguing about planning altenatives. In *International Conference on Computational Linguistics*, 1992.

[Ram89] Lance Arthur Ramshaw. *Pragmatic Knowledge for Resolving Ill-Formedness.* PhD thesis, University of Delaware, 1989.

[Rei80] Raymond Reiter. A logic for default reasoning. *Artificial Intelligence*, (13):81–132, 1980.

[Rei85] R. Reichman. *Getting Computers to Talk Like You and Me.* MIT Press, Cambridge, MA, 1985.

[Rei91] Raymond Reiter. The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result fot goal regression. *Artificial Intelligence and Mathematical Theory of Computation*, pages 359–380, 1991.

[RL92a] Irene Rodrigues and José Gabriel Lopes. A framework for text interpretation. In B. Du Boulay and V. Sgurev, editors, *Artificial Intelligence V - Methodology, Systems and Applications.* North Holland, 1992.

[RL92b] Irene Rodrigues and José Gabriel Lopes. Temporal structure of discourse. In *Proceedings of the COLING'92*, 1992.

[Roc94] Sheila Rock. Understanding repetition in natural language instructions - the semantcs of extent. In *32nd Meeting of the ACL*, 1994.

[Rod94] Irene Rodrigues. *Processamento de Texto: Interpretação Temporal.* PhD thesis, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa, 1994.

[RP91] William Rodi and Stephen Pimentel. A nonmonotonic assumption-based tms using stable models. In *Principles of Knowledge Representation and Reasoning*, pages 485–495, 1991.

[RS91] Gudula Retz-Schmidt. Recognizing intentions, interactions, and causes of plan failures. *User Modeling and User-Adapted Interaction*, 1(2):173–202, 1991.

[RVH+91] M. Reinders, E. Vinkhuyzen, H.Akkermans, J. Balder, B. Bartsch-Spoerl, B. Bredeweg, U. Drouven, F. Harmelen, W. Karbach, Z. Karssen, G. Schreiber, and B. Wielinga. A conceptual modelling framework for knowledge-level reflection. *AIICOM*, 4(2), September 1991.

[Sac77] Earl D. Sacerdoti. *A Structure for Plans and Behavior*. American Elsevier, New York, 1977.

[SC92] Margaret Sarner and Sandra Carberry. Generating tailored definitions using a multifaceted user model. *User Modeling amd User-Adapted Interaction*, 2(3):181–210, 1992.

[Sch92] Emanuel Schegloff. Repair after next turn: The last structurally provided defense of intersubjectivity in conversation. *American Journal of Sociology*, 5(97):1295–1345, 1992.

[SDP96] Michael Schroeder, Carlos Damásio, and Luís Moniz Pereira. Revise report: An architecture for a diagnosis agent. In *ECAI workshop on Integrating Non-Monotonicity into Automated Reasoning Systems*, 1996.

[Sea90] John Searle. *Collective Intentions and Actions, in Intentions in Communication*, chapter 19, pages 401–416. MIT, 1990.

[Sha89] Murray P. Shanahan. Prediction is deduction but explanation is abduction. In *Proceedings of the IJCAI*, 1989.

[Sha94] Murray Shanahan. A circumscriptive calculus of events. Technical report, Imperial College, April 1994.

[Sha95] M. Shanahan. A circumsciptive calculus of events. *Artificial Intelligence*, 77(2):249–284, 1995.

[Sho87] Yoav Shoam. Temporal logics in ai: Semantical and ontological considerations. *Artificial Intelligence*, 33:89–104, 1987.

[SM94] L. Stein and L. Morgenstern. Motivated action theory: a formal theory of causal reasoning. *Artificial Intelligence*, 71(1), 1994.

[Son91] Fei Song. *A Processing Model for Temporal Analysis and its Application to Plan Recognition.* PhD thesis, University of Waterloo, 1991.

[TA94] David Traum and James Allen. Discourse obligations in dialogue processing. In *32nd Meeting of the ACL*, 1994.

[TCM96] Jasper Taylor, Jean Carletta, and Chris Mellish. Requirements for belief models in cooperative dialogue. *User modeling and user-adapted interaction*, 6(1):23–68, 1996.

[TH92] David Traum and Elizabeth Hinkelman. Conversation acts in task-oriented spoken dialogue. Technical Report 425, University of Rochester, 1992.

[THM96] R. Thmason, J. Hobbs, and J. Moore. Communicative goals. In *ECAI 96 Workshop Gaps and Bridges: New Directions in Planning and Natural Language Generation*, 1996.

[TK] Francesca Toni and Robert Kowalski. Reduction of abductive logic programs to normal logic programs.

[TNB89] M. Taylor, F. Néel, and D. Bouwhuis. *The Structure of Multimodal Dialogue.* North-Holland, 1989.

[TOI94] Hisashi Tomatsu, Norihiro Ogata, and Akira Ishikawa. Towards a dynamic theory of belief-sharing in cooperative dialogues. In *15th International Conference on Computational Linguistics*, 1994.

[Tra94] David Traum. *A Computational Theory of Grounding in Natural Language Conversation.* PhD thesis, University of Rochester, 1994.

[Wal92] Marilyn Walker. Redundancy in collaborative dialogue. In *International Conference on Computational Linguistics*, 1992.

[Wal94] Marilyn Walker. Discourse and deliberation: Testing a collaborative strategy. In *15th International Conference on Computational Linguistics*, 1994.

[Wal96] Marilyn Walker. Inferring acceptance and rejection in dialogue by default rules of inference. *Language and Speech*, 39(2), 1996.

[WBE+95] B. Webber, N. Badler, B. Di Eugenio, C. Geib, and L. Levinson Amd M. Moore. Instructions, intentions and expectations. *Articial Intelligence*, 73(1-2):253–269, 1995.

[WC88] Robert Wilenski and David Chin. The berkeley unix consultant project. *Computational Linguistics*, pages 35–84, 1988.

[Wil78] Robert Wilenski. Why john married mary: Understanding stories involving recurring goals. *Cognitive Science*, 2:235–266, 1978.

[Wil81] Robert Wilenski. Meta-planning: Representing and using knowledge about planning in problem solving and natural language understanding. *Cognitive Science*, 5:197–233, 1981.

[WS] Steve Whittaker and Phil Stenton. Cues and control in expert-client dialogues.

[WW90] Marilyn Walker and Steve Whittaker. Mixed initiative in dialgue: An investigation into discourse segmentation. In *28th Annual Meeting of ACL*, pages 70–78, 1990.

[YA93] Ed Yampratoom and James Allen. Performance of temporal reasoning systems. Technical Report 93-1, University of Rochester, May 1993.

[Yam94] Ed Yampratoom. Using simulation-based projection to plan in an uncertain and temporally complex world. Technical Report 531, University of Rochester, September 1994.

[ZM93] Ingrid Zukerman and Richard McConachy. Consulting a user model to address a user's inference during content planning. *User Modeling and User-Adapted Interaction*, 3(2):107–154, 1993.