UNIVERSIDADE TÉCNICA DE LISBOA
INSTITUTO SUPERIOR TÉCNICO

Gestão e Representação de Domínios em Sistemas de Diálogo

Márcio Duarte Albasini Mourão

(Licenciado em Engenharia Informática e de Computadores, I.S.T.)

Dissertação para obtenção do Grau de Mestre em
Engenharia Informática e de Computadores

Tese realizada sob a orientação de

Nuno Mamede
Professor Auxiliar do Instituto Superior Técnico
da Universidade Técnica de Lisboa

Constituição do Júri

Presidente:	Doutor Nuno João Neves Mamede
Vogais:	Doutor Paulo Miguel Torres Duarte Quaresma
	Doutor João Paulo da Silva Neto
	Doutor David Manuel Martins de Matos

Setembro de 2005

Resumo

Um sistema de diálogo de língua falada permite a um utilizador interagir com uma máquina em língua natural. Para além de um conjunto de serviços relacionados com a adaptação ao domínio, um sistema deste tipo compreende tipicamente as seguintes etapas: (i) reconhecimento de fala; (ii) representação das intenções do utilizador; (iii) gestão do diálogo; e (iv) síntese de fala.

Neste trabalho propõe-se um gestor de diálogo e estende-se um gestor de serviços. O gestor de diálogos é responsável pela escolha da interpretação das intenções do utilizador, pela selecção do comportamento do sistema e pela geração de uma resposta textual. Por sua vez, o gestor de serviços é responsável pela gestão do domínio, e comporta um modelo de representação de domínios que facilita a sua construção e inclusão em sistemas de diálogo. Adicionalmente, foi elaborado um modelo de integração do gestor de diálogo com o gestor de serviços, permitindo que o gestor de diálogo seja independente do domínio.

Estes elementos foram integrados num sistema de diálogo e testados em dois domínios diferentes: o da Domótica e dos Autocarros (venda de bilhetes). Do trabalho no domínio da Domótica resultou um sistema que pode ser visto na "Casa do Futuro", no Museu das Telecomunicações em Lisboa.

Abstract

A spoken dialogue system allow users to interact with a machine in natural language. In addition to a set of services related to a domain, such systems tipically perform the following steps: (i) speech recognition; (ii) representation of user intentions; (iii) dialogue management; and (iv) speech synthesis.

This work proposes a dialogue manager and extends an existing service manager. The dialogue manager is responsible for user intentions interpretation choice, for the selection of the system behavior, and for the generation of a textual answer. The service manager is responsible for domain management and includes a domain representation model that facilitates its construction and inclusion in dialogue systems. Adicionally, an integration model for these two components was defined. This integration model allows the dialogue manager to be domain independent.

These components were incorporated in a dialogue system and tested in two different domains: the Domotic domain and the Bus domain (ticket sales). The Domotic domain is in use in "Casa do Futuro" (House of the Future), in the Lisbon Telecommunications Museum.

Palavras Chave
Keywords

Palavras chave

Sistema de Diálogos
Gestor de Diálogos
Serviço
Acto de Resolução de Problemas
Interpretação
Obrigação de Discurso

Keywords

Dialogue System
Dialogue Manager
Service
Problem Solving Act
Interpretation
Discourse Obligation

Agradecimentos

Obrigado a todos por tudo.

Conteúdo

Lista de Figuras

Convenções Tipográficas

Nesta tese utiliza-se um conjunto de convenções tipográficas que se enunciam de seguida:

- usa-se texto [entre parêntesis rectos] para as referências bibliográficas;
- usa-se texto em *Itálico* para frases ilustrativas no meio de um texto e para as palavras em Inglês;
- usa-se texto em **Negrito** para palavras ou frases chave.

Capítulo 1

Introdução

Nos últimos anos, com o crescimento da tecnologia, surgiu um novo conjunto de dispositivos electrónicos que proporcionou a realização de certas tarefas mais facilmente. Devido à menos boa preparação das pessoas para lidar com as novas tecnologias e às interfaces menos adequadas que são colocadas à disposição, o contacto com os dispositivos é muitas vezes realizado com dificuldade. Porque é o meio de comunicação usado entre humanos, a utilização de diálogos de língua falada como interface de comunicação com os dispositivos faz com que a interacção se torne natural [6].

Um sistema de diálogo é um sistema que permite a utilizadores interagirem com uma máquina, com o objectivo de pesquisar, inserir informação, conduzir transacções, e controlar sistemas domóticos, utilizando língua natural. Um sistema de diálogo de língua falada envolve a integração de vários componentes: (i) reconhecimento da fala; (ii) compreensão da linguagem; (iii) gestão e representação do diálogo; e (iv) geração da linguagem e síntese de fala.

O objectivo desta tese foi construir um gestor de diálogos independente do domínio. Para tal, desenvolveu-se o componente de gestão e representação de diálogos (gestor de diálogos), que inclui: (i) *Gestor de interpretações*; (ii) *Gestor do contexto*; (iii) *Gestor das tarefas*; (iv) *Gestor de comportamentos*; (v) *Gestor de geração*; (vi) *Gerador de superfície*; e (vii) *Gestor de serviços*. Os resultados obtidos são apresentados com exemplos de diálogos em dois domínios distintos: (i) Domótica; e (ii) Autocarros. O domínio da Domótica está relacionado com o controlo de dispositivos numa casa, como as luzes, os estores, os vidros, as televisões, e outros. O domínio dos Autocarros relaciona-se com a compra de bilhetes de autocarro de longo curso.

Este capítulo pretende introduzir o leitor ao tema da interacção dos humanos com as máquinas através de diálogos. A primeira secção apresenta as razões que levaram à prossecução deste trabalho, estabelecendo a motivação de base para a elaboração desta dissertação. A segunda secção pretende alertar o leitor para os problemas, as vantagens e as desvantagens do uso da língua natural como meio de comunicação. A terceira secção introduz o tema principal desta tese, evidenciando características presentes em qualquer diálogo, definindo e esclarecendo as diferenças entre sis-

temas e gestores de diálogo. A secção 4 faz um resumo das contribuições dadas por esta tese ao grupo de trabalho do L^2F. O capítulo termina com a apresentação da estrutura da dissertação.

1.1 Motivação

Para além das motivações intrínsecas ao desenvolvimento de um gestor de diálogos, a realização de um trabalho final de curso na área de sistemas de diálogo contribuiu de forma decisiva para a escolha deste tema de trabalhos. A realização do trabalho final de curso no L^2F permitiu enraizar conhecimentos sobre a forma como os humanos dialogam usando a língua natural. Permitiu também modelar a comunicação a ser usada com as máquinas. No âmbito deste trabalho desenvolveu-se uma primeira versão do gestor de diálogos, denominado STAR[1].

Para além do conhecimento prévio adquirido sobre sistemas de diálogo, foi também uma motivação conhecer préviamente as pessoas que trabalhavam no L^2F, nomeadamente o orientador desta tese. A aplicação do trabalho no projecto "Casa do Futuro" [2] foi também um factor importante na escolha da realização da tese, uma vez que permitiu a utilização real do trabalho, e facilitou a identificação de muitos dos problemas que ocorreram no desenvolvimento do gestor de diálogos.

1.2 A Língua Natural como Meio de Comunicação

A capacidade do ser humano em receber estímulos, interpretar, raciocinar com base nessas mesmas interpretações e responder de forma racional, ou seja, de acordo com o seu conhecimento, sempre foi das mais importantes definições do próprio ser humano. A mutável e inexorável capacidade de relacionamento e de diálogo foi, sem dúvida, aquela que fez com que não se vivêsse apenas como sistemas isolados, mas em grupos populacionais, sociedades, e que permitiu um desenvolvimento cada vez maior destas capacidades de interpretação, de raciocínio, e de resposta, e inevitavelmente, um desenvolvimento científico e cultural.

O conhecimento científico actual permite afirmar, desde já, aquela que é incontornavelmente uma das principais, senão a principal potencialidade do ser vivo. O ser vivo pode comunicar, transmitir informação, independentemente do lugar de onde foi transmitida, do como, do porquê, e do modo como é interpretada por outros. Ainda há cerca de meio século atrás podia-se estabelecer uma analogia directa entre os seres inanimados e as máquinas. Sem ser considerada pura ficção científica, não era de modo nenhum considerada a hipótese de se poder vir a comunicar com uma máquina. Porém, principalmente com os desenvolvimentos surgidos na área de Inteligência Artificial, uma nova visão e expectativa paira sobre o assunto. A ideia de se poder comunicar com

[1] Alusivo à arquitectura de cinco componentes distribuídos em forma de estrela usados na implementação do gestor de diálogos.

[2] Projecto num espaço do Museu das Telecomunicações em Lisboa, onde é possível controlar aparelhos como as luzes, os estores, os vidros e as televisões, usando o sistema de diálogos.

uma máquina, transformar um objecto inanimado em animado, transcende os mais ousados pensamentos, e já há quem fale na "atribuição" de vida. A reforçar essa ideia há os inúmeros filmes de ficção científica que aguçam o engenho dos cientistas desta área: veja-se o caso do "David" no filme "Inteligência Artificial", do "Sonny" no filme "I, Robot", ou do "C3PO" no filme "Star Wars".

A interacção entre o utilizador e a máquina por meio de língua natural falada, não só torna os processos mais eficientes, como mais intuitivos, e o esforço despendido na utilização das máquinas poderá permitir uma maior aproximação entre as pessoas e as máquinas. Com os esforços de especialistas como James Allen[3], na compreensão da língua natural, discurso e representação de conhecimento, raciocínio de senso comum e planeamento, e Rosalind Picard[4], na investigação de formas de implantação de emoções nas máquinas, as futuras gerações poderão um dia dialogar com muitos dos objectos que fazem parte do seu ambiente quotidiano.

Os sistemas de diálogos poderão satisfazer uma variedade considerável de objectivos, desde a obtenção dos mais variados tipos de informação, ao controlo de diversos sistemas. Um exemplo deste tipo de sistemas são os sistemas domóticos. Nos domicílios, que se querem pessoais e intimistas, a automação começa a dar os primeiros passos e a utilização da língua natural e do diálogo pode aumentar a qualidade de vida.

A língua natural é uma das muitas interfaces que podem ser utilizadas num diálogo entre o utilizador e a máquina. Contudo, dada a simplicidade com que os humanos utilizam a língua natural para interagir, existe um grande apelo para que a interacção com a máquina seja feita, utilizando a mesma linguagem. Esta não é uma tarefa simples. A grande maioria das interacções pessoa-máquina deixa de parte a língua natural como interface, com a justificação de que a linguagem é ambígua. Usualmente, o grau de ambiguidade da língua natural é considerado demasiado elevado para que possa ser utilizado eficazmente numa interface [8]. As implementações de língua natural com sucesso são caracterizadas como sendo restrictas na sintaxe ou no léxico, como forma de evitar as ambiguidades. Quando os sistemas usam estas restrições nas suas estruturas, é assumido que o utilizador terá de aprender que estruturas são aceitáveis, tornando a língua natural não mais útil do que uma linguagem formal de comandos. Apesar destas contrariedades, alguns sistemas mantêm a maioria das vantagens das interfaces de língua natural.

Uma interface de língua natural sem restrições, se pudesse ser implementada, ofereceria muitas vantagens: (i) seria fácil de aprender e de lembrar, porque a sua estrutura e vocabulário seriam já familiares para o utilizador; (ii) a transferência de conhecimento entre aplicações seria fácil de realizar, porque a mesma linguagem poderia ser utilizada para muitas aplicações; (iii) haveria várias formas de executar uma acção, e de forma bastante eficiente, já que o utilizador não teria de navegar por entre ecrãs de menus para atingir o seu objectivo; (iv) permitiria uma flexibilidade considerável nas várias fases de execução de uma tarefa [9].

[3] James Allen é actualmente investigador no departamento de ciências da computação na universidade de Rochester, e é autor do livro "Natural Language Understanding" [4].

[4] Rosalind Picard é actualmente directora do grupo de computação afectiva do Media Laboratory do Massachussets Institute of Technology (MIT) e autora do livro "Os computadores Emocionais" [5].

A diferença entre língua natural escrita e falada é geralmente ignorada, mas a utilização da fala torna a interacção muito mais eficiente. Mesmo que a fala seja apenas utilizada numa linguagem restricta de comandos para o computador, o seu uso liberta as mãos para outras tarefas, permitindo que o utilizador execute mais do que uma tarefa ao mesmo tempo.

Infelizmente, a língua natural é geralmente ambígua e dependente do conhecimento do mundo. Os próprios humanos têm dificuldade em resolver as ambiguidades. Os mal entendidos, por exemplo, são tipicamente resultado da exploração de interpretações diferentes de uma mesma mensagem. Para implementar um sistema de língua natural, é muitas vezes necessário restringi-lo a apenas um subconjunto limitado de vocabulário e sintaxe de uma língua natural completa. Isto permite reduzir a ambiguidade e manter o tempo de processamento dentro de limites razoáveis, uma vez que elimina uma série de interpretações possíveis. Para continuar a ser considerado uma interface de língua natural, a maioria dos aspectos positivos desse tipo de interfaces devem ser mantidos. As propriedades de facilidade de uso e de lembrança são mantidos se, de algum modo, forem fornecidas as limitações do sistema ao utilizador, sem que este tenha de as aprender explicitamente. Adicionalmente, muito por causa dos filmes e séries de ficção científica, a grande maioria dos utilizadores atribui às interfaces de língua natural mais inteligência do que a que elas realmente têm. Isto contribui para a criação de expectativas irreais das capacidades do sistema, o que torna difícil a aprendizagem das restrições do mesmo, e origina decepções quando o sistema falha ou não produz o(s) resultado(s) esperado(s) [9][10].

Em qualquer fase do processamento de língua natural podem ocorrer erros que são difíceis de detectar e de corrigir. Véronis [11] propôs uma tipologia de erros que podem ocorrer em interacções pessoa-máquina. Os erros são distinguidos como sendo erros produzidos pelo sistema ou pelo operador humano, também denominados de erros de desempenho ou erros de competência, respectivamente. Enquanto que os erros de competência surgem da falta de conhecimento das regras linguísticas, os erros de desempenho ocorrem apesar da correcta compreensão das regras. Todas as classes de erros podem ocorrer nos níveis lexical, sintáctico e semântico de processamento.

A decisão de usar, ou não, a língua natural como interface de um sistema, deve ser baseada no desempenho esperado desta componente na aplicação. Se a língua natural for usada como interface, vários aspectos devem ser tidos em consideração no desenho das mensagens utilizadas na interacção com o utilizador:

- **As mensagens devem refectir a linguagem restrita usada pelo sistema:** as mensagens de retorno do sistema em língua natural devem modelar as capacidades de interpretação do sistema. No exemplo da figura 1.1, o sistema utiliza o termo “vocabulário”, mas desconhece o seu significado.

- **As mensagens não devem ser antropomórficas:** a presença de pronomes na primeira pessoa nas mensagens resulta em frases mais compridas e em mais pedidos indirectos. Estes factores conduzem a uma maior dificuldade no processo de interpretação e a um aumento da probabilidade de ocorrência de disfluências (ver figura 1.2).

```
UTILIZADOR: Roqueira
SISTEMA: Essa palavra não faz parte do meu vocabulário.
UTILIZADOR: O que faz parte do teu vocabulário?
SISTEMA: Não percebi o que disse.
```

Figura 1.1: Exemplo de diálogo quando a resposta do sistema não reflecte a linguagem restrita usada.

```
UTILIZADOR: Quero comprar um bilhete de avião para lá.
SISTEMA: Não nos é possível neste momento comprar um
         bilhete de avião para Londres.
UTILIZADOR: Não quero bilhete para Londres mas sim para
            Amesterdão.
```

Figura 1.2: Exemplo de diálogo com dificuldade de interpretação devido ao uso de pronomes.

- **As mensagens de erro devem corrigir as falhas do utilizador, e se possível, fornecer alternativas:** se se puder determinar que um erro do utilizador foi causado por erros de competência linguística do mesmo, este deve ser explicitamente informado das regras correctas. Caso contrário, o erro pode levar a uma degradação da qualidade do diálogo. Se a fonte do erro não puder ser determinada, será proveitoso providenciar um diálogo de clarificação, onde o utilizador possa indicar uma correcção apropriada (ver figura 1.3).

```
UTILIZADOR: Quero partir de Lisboa às 19:00 para chegar ao
            Porto às 17:00.
SISTEMA: A hora de chegada não pode ocorrer mais cedo do
         que a hora de partida.
         Por favor, volte a indicar as horas em que
         pretende a sua viagem.
UTILIZADOR: Perdão, quero chegar ao Porto às 22:00.
```

Figura 1.3: Exemplo de diálogo com correcção do sistema a uma informação contraditória dada pelo utilizador.

1.3 Gestor de Diálogos

Esta secção introduz o tema principal desta tese, o gestor de diálogos. Começa por evidenciar características dos diálogos inter-humanos, para depois definir e esclarecer quais as diferenças subjacentes a um sistema e a um gestor de diálogos. Uma vez que se quer que os diálogos a realizar com as máquinas sejam próximos dos realizados entre humanos, a estrutura ideal de um sistema e de um gestor de diálogos será aquela que permita essa proximidade. Só assim se poderão obter diálogos mais realistas e naturais.

Em geral, a estrutura de um diálogo é composta por sete características fundamentais [17]:

1. Mudança de Turno

2. Pares Adjacentes e Inserções
3. Interpretação e Correcção
4. Contexto do Diálogo
5. Elipses
6. Resolução de Referências
7. Iniciativa Mútua

1. **Mudança de Turno:** corresponde à determinação de quando, como e por quanto tempo deve cada participante do diálogo falar (ver figura 1.4).

```
HUMANO1: Para onde quer ir?
HUMANO2:   (não diz nada)
HUMANO1: Peço desculpa, não o compreendi.
         Por favor, especifique um destino.
```

Figura 1.4: Exemplo de mudança de turno.

2. **Pares adjacentes e Inserções:** para que num diálogo se obtenham as respostas mais adequadas, é necessário que os participantes sejam capazes de corrigir o que foi dito ou de pedir informação adicional (ver figura 1.5).

```
HUMANO1: Para onde quer ir?
HUMANO2: Efectuam viagens para o Porto?
HUMANO1: Sim, efectuamos.
HUMANO2: Então quero um bilhete para o Porto.
```

Figura 1.5: Exemplo de par adjacente e inserção.

3. **Interpretação e Correcção:** os participantes num diálogo devem poder confirmar que a sua contribuição para o diálogo foi compreendida. Quando essa compreensão não é alcançada, os participantes devem ter forma de corrigi-la (ver figura 1.6).

```
HUMANO1: Para onde quer ir?
HUMANO2: Para o Porto.
HUMANO1: Então quer um bilhete para Coimbra.
HUMANO2: Não, para o Porto.
HUMANO1: Ok. Deseja ir para o Porto.
```

Figura 1.6: Exemplo de interpretação e correcção.

4. **Contexto do Diálogo:** todas as contribuições e conclusões prévias de um participante devem ser utilizadas para interpretar a contribuição corrente e decidir a próxima contribuição (ver figura 1.7).

```
HUMANO1: De onde quer partir?
HUMANO2: Quero partir de Lisboa.
HUMANO1: E para onde quer ir?
HUMANO2: Quero ir para o Porto.
HUMANO1: Então quer um bilhete de Lisboa para o Porto.
```

Figura 1.7: Exemplo de contexto de diálogo.

```
HUMANO1: Para onde quer ir?
HUMANO2: Porto
HUMANO1: Então quer comprar um bilhete para o Porto.
```

Figura 1.8: Exemplo de elipse.

5. **Elipses:** os participantes num diálogo devem saber interpretar as respostas fragmentadas geralmente usadas em situações de diálogo (ver figura 1.8).

6. **Resolução de Referências:** os participantes de um diálogo devem conseguir resolver a ambiguidade de expressões de referência que surgem em situações de diálogo (ver figura 1.9).

```
(O utilizador encontra-se em Lisboa)
HUMANO1: Para onde quer ir?
HUMANO2: Para o Porto.
HUMANO1: Quer partir daqui?
HUMANO2: Sim.
Sistema: Então quer partir de Lisboa com destino ao Porto.
```

Figura 1.9: Exemplo de resolução de referências.

7. **Iniciativa Mútua:** qualquer diálogo deve ter em conta a quantidade, qualidade e o nível de contribuição de cada participante (ver figura 1.10).

```
HUMANO1: Para onde quer ir?
HUMANO2: Para o Porto.
HUMANO1: Quer partir aqui de Lisboa? E a que horas?
HUMANO2: Sim, as 20:00.
```

Figura 1.10: Exemplo de iniciativa mútua.

Nos últimos anos tem-se assistido ao desenvolvimento de sistemas de diálogo de língua falada em muitos países, quer como protótipos de pesquisa, quer como aplicações comerciais. Estes sistemas permitem a utilizadores interagirem com uma máquina, com o objectivo de pesquisar, inserir informação, conduzir transacções e controlar sistemas domóticos, utilizando língua natural. Um sistema de diálogo de língua falada envolve a integração dos seguintes componentes (ver figura 1.11):

1. Reconhecimento de fala

2. Compreensão da linguagem
3. Gestão e representação do diálogo
4. Geração da linguagem
5. Síntese de fala

Figura 1.11: Componentes fundamentais de um sistema de diálogos.

1. **Reconhecimento de fala:** o reconhecimento de fala consiste na identificação de palavras faladas por parte de uma máquina. Este módulo converte um sinal sonoro vocal correspondente ao que foi dito em texto.
2. **Compreensão da linguagem:** a compreensão da linguagem é uma das partes mais importantes de um sistema de diálogo. Esta componente tem como objectivo reconhecer as intenções do utilizador. Esta tarefa só será possível de realizar mediante o conhecimento do domínio e do contexto sobre o qual o diálogo se desenrola. As várias possíveis interpretações ou reconhecimento, por parte de uma máquina, das intenções de um utilizador, tornam esta tarefa uma das mais complexas.
3. **Gestão e representação do diálogo:** a gestão e representação do diálogo tem um papel fundamental na estrutura, organização e fluxo de mensagens no sistema. A principal função de um gestor de diálogos consiste na gestão de todas as operações necessárias à realização de um objectivo, tornado comum ao utilizador e à máquina. Para o efeito, é indispensável que o gestor seja capaz de: (i) criar as novas obrigações do sistema, face ao reconhecimento das intenções do utilizador; (ii) interagir com as componentes responsáveis pela comunicação com as aplicações externas; e (iii) providenciar a geração de uma resposta ao utilizador.

   A principal função da componente de gestão e representação do diálogo é controlar o fluxo do diálogo. Isto envolve as seguintes tarefas:

   - Verificar se o utilizador já forneceu informação suficiente para permitir a comunicação com a aplicação externa;
   - Comunicar com a aplicação externa;
   - Fornecer uma resposta ao utilizador.

   O Gestor gere o conhecimento tendo em conta:

   - **Contexto:** A história do diálogo, o perfil do utilizador, as expectativas do utilizador, referências a elementos passados e as respostas geradas, são consideradas parte do contexto de um diálogo;

- **Raciocínio:** O raciocínio é feito sobre as intenções do utilizador, sobre o domínio do diálogo, sobre as intenções do Gestor, as acções a executar, as frases, e as respostas a gerar;
- **Clarificação:** Quando é detectada uma ambiguidade, deve ser iniciado um diálogo específico de forma a esclarecê-la;
- **Reparação:** Quando são detectados erros, o sistema deve tomar todas as medidas necessárias para os corrigir. A correcção pode passar por um aviso do sistema ao utilizador, informando-o que deve corrigir o erro;
- **Alerta:** Quando um acontecimento externo importante ocorre, o Gestor deve ser capaz de informar o utilizador do sucedido.

4. **Geração da linguagem:** a geração da linguagem consiste em gerar as respostas textuais do sistema que mais se adequem às intenções da máquina, durante o desenrolar do diálogo com o utilizador.

5. **Síntese de fala:** a síntese de fala tem como objectivo converter as respostas textuais do sistema em ondas sonoras, que serão emitidas como fala. Na maioria das vezes, as frases chegam a este módulo como uma sequência de palavras sem nenhum espaçamento temporal entre elas.

O desenvolvimento da tecnologia de fala, processamento de língua e modelação de diálogos, assim como o aparecimento de computadores cada vez mais potentes, fez com que, as companhias de telecomunicações, entre outras, procurassem incorporar tecnologia de fala nos seus produtos. As aplicações já lançadas no mercado estão relacionadas com o preenchimento de formulários, portais de voz e postos de atendimento telefónico automático. Foram também já criados sistemas para pesquisa de informação, controlo de dispositivos e para sistemas tutoriais.

O exemplo da figura 1.12 mostra um diálogo entre um estudante e um tutor para o envio de fotos via Internet [7]. Apesar dos avanços já alcançados, é importante realçar que as aplicações actuais são ainda bastante limitadas.

Numa arquitectura simplificada, estas tarefas podem ser vistas como um conjunto ordenado. O sistema descobre o que o utilizador quer saber ou fazer, consulta uma possível aplicação externa e responde ao utilizador. Tipicamente, o que o utilizador quer saber ou fazer é uma tarefa problemática e a informação dada pelo utilizador pode não ser suficiente para permitir ao sistema consultar a aplicação externa. A falha de componentes como o reconhecimento de fala e de compreensão da linguagem pode também fazer com que o sistema não interprete de forma correcta aquilo que foi dito pelo utilizador. Vários processos de gestão de erros envolvem clarificação e verificação de sub-diálogos entre o utilizador e o sistema. Isto implica uma arquitectura mais complexa, em que a componente de gestão de diálogos tem um papel central.

```
TUTOR-1: Alright, let's go on. You want to take photos and
send them over the Internet. Consider this problem. What
hardware do you need to take photos and send them over
the Internet?
STUDENT-1: First of all you need to have access to
the internet.
TUTOR-2: What else?
STUDENT-2: a camera
TUTOR-3: Yeah. For what type of connection do you need
a network card?
STUDENT-3: network card
TUTOR-4: Okay. Now, if you have access to the Internet
through a network card, then your connection is ...
STUDENT-4: I don't know.
TUTOR-5: Is direct. Okay. A network card is needed if
you have a direct connection to the Internet. Okay. How
does the user get hooked up to the internet?
STUDENT-5: by a plug.
TUTOR-6: Once again, a network card is needed if you
have a direct connection to the Internet. Okay, what does
a modem do?
STUDENT-6: connects network card and internet
TUTOR-7: Okay. Try this. A modem is required when
there is ...
STUDENT-7: a connection
TUTOR-8: a dial up connection.
..
..
TUTOR-30: Let's review. To send your photos on the
Internet, you need either a digital camera or a regular
camera to take the photos. If you use a regular camera,
you need a scanner to scan them onto a computer disk.
If you have a direct connection to the Internet, then you
need a network card. A modem is needed if you have a
dial up connection.
```

Figura 1.12: Exemplo do diálogo no Tutor.

1.4 Principais Contribuições

As principais contribuições desta tese podem ser resumidas como se segue:

1. Introdução dos módulos do gestor de diálogo na arquitectura de comunicação *Galaxy*;

2. Introdução das obrigações de discurso (intenções do sistema) (ver secção 3.6.2) no gestor de diálogo;

3. Construção da representação de um domínio e de um dispositivo;

4. Reestruturação da comunicação com o Gestor de serviços;

5. Construção da representação do domínio da Domótica e do domínio dos Autocarros.

1.5 Organização da Tese

Esta tese está organizada como se segue:

- **Capítulo 2: Estado da Arte**

  Neste capítulo é feita uma descrição do trabalho relacionado com sistemas de diálogos. Apresenta-se cada um dos componentes e é pormenorizado o papel de um gestor de diálogos na arquitectura de um sistema de diálogos. O capítulo termina com a descrição de alguns dos sistemas de diálogos mais bem sucedidos até à data.

- **Capítulo 3: Gestor de Diálogos**

  Este capítulo é inteiramente dedicado ao gestor de diálogos desenvolvido no âmbito deste trabalho. Neste capítulo são descritas as funcionalidades de cada um dos componentes do gestor, bem como as principais estruturas de dados envolvidas nos processos.

- **Capítulo 4: Resultados e Avaliação**

  Apresentam-se neste capítulo os resultados mais importantes deste trabalho através de exemplos de diálogos produzidos em dois domínios distintos: (i) Domótica; e (ii) Compra de bilhetes de autocarros. Para cada exemplo é dada uma explicação das principais intervenções do sistema no decorrer do diálogo. Estas intervenções surgem na sequência das estratégias descritas no capítulo 3. Neste quarto capítulo é também feita uma avaliação do sistema de diálogo. São descritas as características do sistema e dos diálogos produzidos.

- **Capítulo 5: Trabalho Futuro e Conclusões**

  Neste capítulo são apresentadas conclusões e é descrito trabalho futuro. Aqui se inclui uma descrição da situação atingida e algumas considerações sobre o que seria o "sistema ideal". O capítulo termina com observações finais sobre o trabalho desenvolvido no L^2F.

A descrição dos domínios utilizados no sistema de diálogos encontra-se em apêndice: enquanto que no apêndice A é descrito o domínio da Domótica, no apêndice B é descrito o domínio dos Autocarros.

Capítulo 2

Estado da Arte

Este capítulo procura dar ao leitor uma compreensão pormenorizada dos sistemas e dos gestores de diálogo. Começa por contextualizar o surgimento dos sistemas de diálogo numa panorâmica histórica, onde os últimos quarenta anos deram um contributo decisivo, para depois definir cada um dos componentes usados no desenvolvimento de um sistema de diálogo. Porque é um componente fundamental de um sistema, e central ao tema desta tese, dedica-se uma secção aos requisitos necessários ao desenvolvimento de um gestor de diálogos. Na secção sobre as funcionalidades relevantes de um sistema de diálogos são apresentadas algumas estratégias, importantes para concretizar os objectivos a que se destina um gestor. Estas estratégias incluem a diferenciação das formas de identificação de interpretações. Porque os actos de fala são importantes a vários níveis de aplicação, nomeadamente nos sistemas de diálogo, e porque a entrada do gestor de diálogos desenvolvido no decorrer desta tese consiste neste tipo de actos, é dedicada uma secção ao tema, com exemplos e aplicações particulares a um sistema de diálogos. Este capítulo distingue também os três tipos de sistemas de diálogo existentes: (i) baseado em estados; (ii) baseado em enquadramentos; e (iii) baseado em agentes. O capítulo termina com uma descrição dos principais sistemas existentes.

2.1 História dos Sistemas de Diálogo

O processamento do discurso e do diálogo tem uma história relativamente recente. Por um lado, a análise do discurso e a análise conversacional foram consideradas áreas de pesquisa em linguística no princípio dos anos 70. A análise do discurso foca-se no acto de fala [25] enquanto que a análise conversacional considera o discurso uma interacção social [3]. Por outro lado, a computação na área de humanidades tem sido importante desde os anos 70 (a Associação para Computadores e Humanidades (ACH) foi fundada em 1973). Os esforços computacionais em humanidades se centraram na análise e na geração de texto [32].

Com os desenvolvimentos surgidos na área do processamento da língua natural (reconhecimento

da fala, analisadores sintácticos, síntese de fala, entre outros), e a crescente disponibilidade de recursos de hardware (velocidade dos processadores, memória, ...) aliada a novas capacidades de utilização, o papel do discurso e do processamento do diálogo tornaram-se cada vez mais importantes em sistemas de língua natural. O processamento do discurso e do diálogo em texto é um importante aspecto de investigação em diversas disciplinas como a geração de texto, extracção de informação, sumarização de texto, análise de corpus, e a tradução automática de textos.

Em 1984, a DARPA (Defense Advanced Research Projects Agency) começou a recolher fundos para o reconhecimento contínuo de fala. A capacidade de reconhecimento contínuo de fala possibilitou que uma maior pesquisa em Interacção Humano-Máquina em sistemas de língua falada fosse feita. Os tipos de características que devem ser abordados no discurso e no processamento do diálogo em sistemas de diálogos de língua falada são abrangentes, e incluem: (i) determinação da participação de cada um dos envolvidos no diálogo; (ii) interpretação do sistema para uma dada entrada; e (iii) determinação da resposta apropriada para essa mesma entrada.

Num sentido abstracto, a preocupação não reside apenas no que é dito na conversação, mas também com quem e porque participou o locutor. Alguns dos problemas da área incluem saber qual é o conteúdo da informação em sequências de afirmações, e qual o contexto relevante para determinar o significado dessas afirmações.

Aplicações correntes e áreas de pesquisa para o processamento do discurso e do diálogo incluem sistemas de diálogo de língua falada, ambientes multimédia, comunicação entre agentes inteligentes, geração automática de texto, marcadores acústicos de segmentação do discurso, discurso e planeamento, crenças partilhadas no discurso, discurso orientado à tarefa, discurso colaborativo e planos partilhados, anotação do discurso, e segmentação do diálogo.

2.2 Componentes de um Sistema de Diálogo de Língua Falada

Um sistema de diálogo de língua falada envolve a integração de componentes (ver figura 2.1) que tipicamente providenciam as seguintes funcionalidades [2]:

- **Reconhecimento de fala:** conversão de uma frase proferida pelo utilizador (passagem de uma sequência de parâmetros acústicos e fonéticos) numa sequência de palavras;
- **Reconhecimento da linguagem:** análise da sequência de palavras de forma a produzir uma representação semântica da frase, tal que esta possa ser usada pela componente de gestão de diálogo;
- **Gestão do diálogo:** controlo da interacção entre o utilizador e o sistema, incluindo a coordenação das outras componentes do sistema;
- **Comunicação com aplicações externas:** comunicação com sistemas de base de dados, sistemas periciais ou outras aplicações computorizadas;

- **Geração de respostas:** selecção e/ou construção da mensagem a ser enviada pelo sistema ao utilizador;
- **Geração de fala:** síntese de fala a partir de texto como mensagem do sistema para o utilizador.

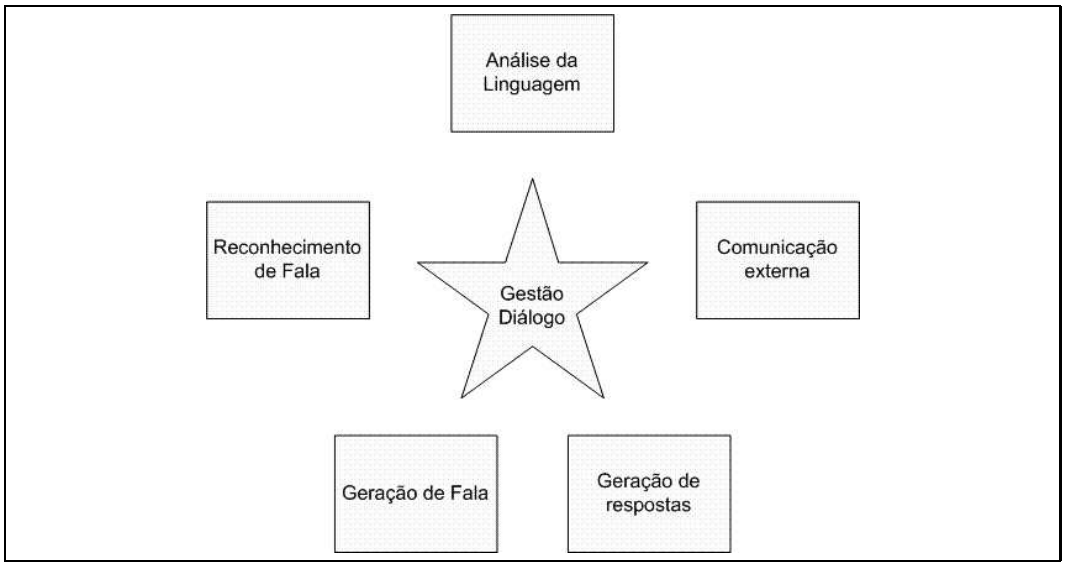

Figura 2.1: Componentes de um sistema de diálogos.

2.3 Análise de Requisitos para um Gestor de Diálogos

Agora que se conhece a principal função de um gestor de diálogos, e a missão que se lhe atribui no contexto de um sistema, é necessário fazer um levantamento dos requisitos que são necessários incubir ao mesmo para a obtenção de diálogos que respeitem a informação descrita em 1.3.

São identificados de seguida um conjunto de requisitos necessários para o desenvolvimento de um sistema conversacional:

- **Historial do diálogo:** um registo de todo o diálogo ocorrido durante o tempo de execução da aplicação. Após o término da aplicação, poderá ser filtrado o registo do historial do diálogo, de forma a guardar apenas os elementos relevantes do diálogo realizado;
- **Registo de tarefas:** representação da informação que é necessária para realizar a tarefa pretendida. Esta informação pode ser representada através de um enquadramento ou de um grafo de estados;
- **Modelo de representação de conhecimento:** suporte de mecanismos de raciocínio de senso comum, necessários para a realização de um diálogo semelhante ao realizado entre humanos;

- **Modelo genérico de conversação:** conjunto de regras inerentes à realização de um diálogo;
- **Modelo(s) do(s) domínio(s) da aplicação:** informação especifica do domínio na qual a aplicação deve ser capaz de realizar o diálogo;
- **Modelo do utilizador:** informação sobre o utilizador que sejam relevantes ao diálogo (ex: objectivos, crenças, intenções).

2.4 Funcionalidades Relevantes de um Gestor de Diálogo

Nesta secção são apresentadas algumas das estratégias que podem ser utilizadas num gestor de diálogos. Naturalmente, os resultados das estratégias seguidas influenciam directamente os tipos de diálogos produzidos pelo sistema. As estratégias não têm de ser concretizadas em exclusividade, podendo ser todas aplicadas em paralelo. Um sistema de diálogos ideal faria uso de todas estas estratégias.

2.4.1 Input mal formado ou incompleto

Uma forma de lidar com entradas de texto mal formadas ou incompletas é informar o utilizador desse problema, para que este possa reformular a entrada. Esta estratégia deverá distinguir os diferentes tipos de entrada mal formadas ou incompletas de forma a guiar o utilizador, facilitando a reformulação.

2.4.2 Interpretação baseada em actos de fala

Um acto de fala é definido como uma função da afirmação [26]. Um pedido, aviso, informação ou promessa é um tipo de acto de fala. A análise dos actos de fala envolve análise sintáctica, semântica, e requer referências para informações externas como o contexto do discurso, as crenças, e as intenções do utilizador. A teoria dos actos de fala [12] tem sido usada para o reconhecimento das intenções associadas às frases proferidas pelo utilizador. A análise dos actos de fala foi usada no desenvolvimento do projecto TRAINS, para suportar a interpretação de entradas mal formadas. Este projecto está ligado ao desenvolvimento de uma tecnologia de diálogo em suporte à resolução de problemas de colaboração [19].

2.4.3 Interpretação baseada no contexto de discurso

O contexto do discurso é usado para auxiliar a interpretação de elementos, como os pronomes ou expressões como “este”, o “próximo”, o “anterior”. Estes elementos fazem referência a um determinado valor. O método mais simples consiste em manter uma história dos valores que possam ser usados mais tarde (resolução de referências anafóricas).

2.4.4 Interpretação baseada na estrutura do diálogo

Esta interpretação faz uso das expectativas providenciadas pelo modelo de diálogo. A ideia essencial é que, em cada ponto do diálogo, existem restrições sobre o que pode ser dito a seguir. Estas restrições podem ser de vários tipos e auxiliar vários componentes do sistema de diálogo. Por exemplo, é possível informar a componente de reconhecimento de fala dos resultados que se espera de uma pergunta feita pelo sistema. O sistema poderá também prever os objectivos e os sub-objectivos que o utilizador procura completar no próximo ponto do diálogo.

2.4.5 Interpretação baseada no modelo do utilizador

A representação de informação sobre o utilizador, referida como modelo do utilizador, é um auxílio à interpretação das frases proferidas pelo utilizador. As frases são interpretadas em termos de planos dependentes do domínio, e um conjunto de objectivos candidatos é derivado e incorporado em modelos de planos do utilizador que providenciam um contexto para a interpretação de futuras declarações.

2.4.6 Confirmações ou Verificações

Os vários tipos de interpretação, vistos nos pontos anteriores, permitem ao sistema compensar as entradas mal formadas ou incompletas, sem ter que pedir ao utilizador que repita ou clarifique. Uma confirmação, como o próprio nome diz, permite confirmar valores estabelecidos no diálogo através de uma pergunta feita ao utilizador. É comum nos diálogos estabelecidos entre os humanos e permite também colmatar as falhas existentes no reconhecimento de fala. Existem diversos tipos de confirmação que o sistema de diálogo pode utilizar, alguns dos quais são descritos nas duas próximas secções.

Confirmação Explícita

A confirmação explícita baseia-se numa questão que solicita explicitamente a confirmação de uma entrada. É um método robusto para confirmar valores, mas aumenta o volume de diálogo necessário para completar uma determinada tarefa.

Confirmação Implícita

Neste tipo de confirmação, o sistema embebe na sua próxima pergunta o entendimento que teve das declarações proferidas pelo utilizador até ao momento. O utilizador pode corrigir o valor repetido, mas se responder apenas à pergunta, confirma o entendimento do sistema.

2.5 Actos de Fala

Nesta secção é definido um acto de fala, e dada a motivação para o uso destes como entrada do gestor de diálogos desenvolvido nesta dissertação. Actualmente, os actos de fala não são apenas importantes para os sistemas de diálogo, como são também importantes para outros tipos de aplicações.

Os actos de fala são actos de comunicação. Comunicar é expressar uma atitude, e o tipo de acto de fala utilizado corresponde ao tipo de atitude expressa. Por exemplo, uma declaração expressa uma crença, um pedido expressa um desejo, e uma desculpa expressa um arrependimento. Como acto de comunicação, um acto de fala tem êxito se a audiência identificar, de acordo com a intenção de quem proferiu a frase, a atitude que se pretende expressar.

Com palavras é possível expressar várias intenções: pedidos, perguntas, ordens, promessas, agradecimentos, desculpas, entre outros. Quase todo o acto de fala é na realidade produzido a partir de vários actos realizados ao mesmo tempo, distinguidos por diferentes aspectos da intenção do utilizador: existe o acto de dizer alguma coisa, o que faz alguém quando diz essa coisa, como um pedido ou uma promessa, e a forma como alguém tenta captar a atenção de uma audiência.

A teoria dos actos de fala é parte taxonómica, parte explanatória. Deve classificar sistematicamente tipos de actos de fala, bem como ter em conta o facto de que a relação entre as palavras usadas e a intenção da frase é frequentemente oblíqua.

A ideia dos actos de fala teve a sua origem na filosofia da linguagem. J. A. Austin [12] foi o primeiro a querer capturar o facto de que existe mais na função da linguagem do que apenas semântica. O mapeamento de entidades de uma proposição em referências, e a definição do verdadeiro valor de uma proposição, constituíam a maior área de interesse na semântica da linguagem. Com Austin, e o seu seguidor, J. R. Searle [13], houve uma mudança para eventos ou actos que ocorrem na linguagem, daí o nome de "actos de fala". Os efeitos destes actos mudam quer o mundo observável, quer os estados mentais dos participantes no diálogo. A abordagem de Austin introduz pragmática no estudo e na modelação da linguagem. Consequentemente, o foco é agora nas frases, e não nas proposições.

De acordo com Austin, existem três tipos de actos de fala em todas as frases:

- **Locucionário:** É o acto de realmente dizer uma frase.
- **Ilocucionário:** É o acto que é tido em dizer qualquer coisa. O acto ilocucionário não tem correspondência de um para um com a locução de onde foi derivado. Existem diferentes locuções que expressam a mesma ilocução e vice-versa. Por exemplo, existem actos de fala indirectos, ou seja, actos com uma natureza diferente daquela que é facilmente deduzível. Um exemplo típico é o da locução da frase "Pode-me passar o sal?", dita numa mesa de jantar. Para esta situação em particular, o significado desta frase é idêntico ao de "Passa-me o sal, se faz favor.", e ninguém assumiria que o pedido foi feito apenas no interesse de saber

se a pessoa a quem se dirigiu a frase era capaz ou não de passar o sal.

- **Perlocucionário:** Representa a mudança alcançada de cada vez que é dito alguma coisa, num contexto em particular. Dependendo do tipo de perlocução, diferentes condições têm de ser estabelecidas para que este acto possa ser alcançado. Por exemplo, o destinatário no exemplo do sal tem de perceber que a intenção de quem falou é a de receber o sal.

Os verbos que denominam o acto de fala que representa a intenção do utilizador, são chamados de performativos. Uma performativa que é dita pela pessoa certa, nas circunstâncias correctas, causa uma mudança no mundo.

Searle's [13] sugere a seguinte classificação de actos de fala:

- **Assertivos:** comprometem a pessoa que fala. Deste tipo, fazem parte os seguintes actos de fala: sugerir, jurar, concluir. Ex.: "Ninguém faz um bolo melhor do que eu.".
- **Directivos:** procuram fazer com que o destinatário realize uma acção. Deste tipo, fazem parte os seguintes actos de fala: perguntar, ordenar, pedir, convidar, aconselhar. Ex.: "Podes fechar a janela?".
- **Comissivos:** comprometem a fazer alguma coisa no futuro. Deste tipo, fazem parte os seguintes actos de fala: prometer, planear, apostar, opôr. Ex.: "Vou para Paris amanhã.".
- **Expressivos:** expressam os sentimentos da pessoa acerca da situação. Deste tipo, fazem parte os seguintes actos de fala: agradecer, desculpar, deplorar. Ex.: "Peço desculpa por te ter mentido.".
- **Declarativos:** mudam o estado do mundo com efeito imediato. Ex.: "Estás despedido.".

2.5.1 Exemplos

Ficam aqui exemplificados alguns dos actos de fala utilizados no dia-a-dia. Todos estes actos de fala se enquadram numa das classificações sugeridas por Searle.

- **Cumprimento:** "Boa tarde".
- **Pedido:** "Quero um bilhete de comboio para o Porto".
- **Queixa:** "Já estou à espera da máquina há três meses, quando me disseram que levaria apenas um!"
- **Convite:** "No próximo Sábado damos uma festa. Queres vir?"
- **Elogio:** "Estás tão bonita hoje, Luísa."
- **Recusa:** "Já não temos mais lugares disponíveis no autocarro."

2.5.2 Aplicações

A teoria dos actos de fala sublinha a importância da distinção entre o uso da linguagem e o significado linguístico. Esta distinção dá forma a formulações de questões sobre a natureza do conhecimento linguístico, separando questões sobre capacidades exercidas na interacção linguística, daqueles específicos à própria linguagem.

Para além de ser importante para a linguística, a teoria dos actos de fala é também importante em ciência computacional. Tem aplicações em Computação Distribuída, Inteligência Artificial Distribuída, Compreensão e Geração de Língua Natural, e em protocolos de troca de dados electrónicos:

- **Computação Distribuída:** As especificações iniciais dos sistemas distribuídos e dos protocolos a que se chega, são frequentemente descritos em termos dos actos de fala realizados pelos processos no sistema. A utilização de semântica dá noções precisas de permissões ou promessas que geralmente são usadas, ainda que informalmente. Estas noções precisas ajudam o desenho de sistemas eficazes e facilitam a verificação das suas especificações, relativamente ao desejo dos seus utilizadores.

- **Inteligência Artificial Distribuída:** As mensagens trocadas por agentes inteligentes podem ser consideradas actos de fala. A semântica dos actos de fala é útil para especificar os critérios de comunicação e outras acções dos agentes. Esses critérios podem ser usados pelos constructores dos sistemas multi-agente e pelos agentes que os compõem.

- **Interpretação das frases do utilizador em Sistemas de Diálogo:** Ajuda a considerar os actos de fala como acções de primeiro plano, em que os participantes de um discurso planeiam e depois realizam essas acções. Uma semântica formal fornece critérios claros para sucessos e falhas, que podem ser usados para melhor integrar as componentes naturais de um sistema inteligente. Note-se o desenvolvimento dos sistemas de diálogo, em que é fundamental perceber a intenção do utilizador face aos serviços disponibilizados pelo sistema. Tal como uma pessoa num diálogo interpreta as frases de outra pessoa, também um sistema de diálogo deve interpretar aquilo que o utilizador proferiu. Actualmente, encontram-se desenvolvidos alguns sistemas de diálogo para controlo médico, para consulta de horários de autocarros, comboios, ou mesmo, de aviões (ver 2.7). A importância deste tipo de sistemas é cada vez maior.

- **Protocolos de troca de dados electrónicos:** Os documentos trocados entre organizações podem ser vistos como actos de fala realizados pelas mesmas. Uma semântica para os actos de fala providencia uma base uniforme para a formalização e especificação formal deste tipo de protocolos, o que é necessário para a criação de uma linguagem comum.

2.5.3 Actos de Fala na Gestão do Diálogo

Há alguns aspectos a ter em conta para que os actos de fala possam ser usados na gestão do diálogo:

- **É necessário que sejam dadas definições precisas de cada acto de fala:** Envolve condições que especificam o que é considerado como sendo determinado acto de fala. Também envolve enumerar efeitos de um acto de fala no estado do diálogo, no estado mental dos participantes, na tarefa, quando aplicável, ou em alguma área do domínio em que o acto de fala possa ter algum efeito;
- **São necessários critérios de reconhecimento:** As definições dos actos de fala utilizam, na sua grande maioria, o estado mental de cada um dos participantes no diálogo, que não é directamente observável. Esta utilização pode tornar insuficientes as condições necessárias para o reconhecimento dos actos de fala. Nesse caso, diferentes critérios de reconhecimento terão de ser definidos, ou as definições já alcançadas deverão ser enriquecidas para permitir um reconhecimento mais robusto;
- **Deverão ser definidos esquemas de planeamento:** Engloba saber o nível a que o sistema de diálogo deve planear, investigar a suficiência de planeamento dos actos de fala da próxima frase.
- **O papel dos actos de fala no diálogo deve ser claro:** Os actos de fala traduzem intenções a serem comunicadas.

A relação dos actos de fala com a gestão do diálogo faz com que seja necessário distinguir tipos de actos de fala para diferentes contextos, para as características de um diálogo e para o contexto específico na qual eles surgem. Por exemplo, diferentes tipos de actos de fala são usados numa interacção pessoa-máquina e numa interacção entre humanos. O mesmo se pode dizer para um tipo de conversa geral, em relação a interacções que sejam orientadas a tarefas.

A grande contribuição do Searle nos actos de fala passou pela sua tentativa de definir formalmente as condições sobre os quais os diferentes tipos de actos ilocucionários são realizados [13]. Por outras palavras, ele procurou definir o contexto que faz com que os actos de fala tenham sucesso. Estas condições envolvem ter conhecimento das convenções de uma linguagem e compreender o acto locucionário.

2.6 Classificação dos Sistemas de Diálogo

Os sistemas de diálogos são sistemas computorizados com que os humanos interagem, e em que a língua natural falada desempenha um papel importante na comunicação. O principal objectivo de um sistema de diálogos de língua falada é providenciar uma interface entre um utilizador e uma aplicação baseada no computador, como uma base de dados ou um sistema pericial. Os sistemas

de diálogo podem ser classificados em três grupos de acordo com os métodos usados no controlo do diálogo [2]:

- Sistemas de estados finitos;
- Sistemas baseados em enquadramentos (templates);
- Sistemas baseados em agentes.

2.6.1 Sistemas de estados finitos

Num sistema de estados finitos, o utilizador é levado através de um diálogo que consiste numa sequência pré-determinada de estados [2]. A maior parte dos sistemas de diálogo usa esta estratégia de controlo. O fluxo do diálogo é dado por um conjunto de estados, com transições que denotam vários caminhos alternativos por um grafo. O sistema mantém controlo do diálogo, produz perguntas em cada estado, reconhece ou rejeita palavras ou frases específicas na resposta às perguntas, e produz acções baseadas na interpretação da resposta. A figura 2.2 mostra o exemplo de um diálogo produzido por um sistema de estados finitos.

```
SISTEMA: O que deseja fazer?
UTILIZADOR: Pagar uma conta.
SISTEMA: A que empresa deseja pagar?
UTILIZADOR: Caixa Geral de Depósitos.
SISTEMA: Quanto deseja pagar?
UTILIZADOR: Duzentos euros.
SISTEMA: Em que data pretende que seja efectuado o pagamento?
UTILIZADOR: Segunda-feira.
```

Figura 2.2: Exemplo de diálogo produzido por um sistema de estados finitos.

A maior vantagem de um modelo de estados finitos é sua simplicidade. Este modelo é particularmente adequado para modelar um fluxo de diálogo de uma tarefa bem estruturada, que envolve informação a ser trocada numa sequência pré-determinada. A semântica do sistema é clara e intuitiva. Como as respostas permitidas ao utilizador são muito limitadas, é colocada pouca tecnologia nas componentes do sistema, principalmente no reconhecimento de fala. Por esta razão, é o método mais utilizado nos sistemas comerciais disponíveis.

Este tipo de sistemas não são bons para modelar tarefas estruturadas, caracterizadas por subtarefas cuja ordem é difícil de prever. Quando existe dependência entre os itens de informação, a utilização deste sistema torna-se impraticável, levando a uma explosão combinatória de estados e transições.

2.6.2 Sistemas baseados em enquadramentos (*templates*)

Num sistema baseado em enquadramentos ou *templates* são colocadas questões ao utilizador que permitem ao sistema preencher espaços de informação [2]. Neste tipo de sistema, o fluxo do diálogo não é determinado à priori. Por isso, funcionam como sistemas de produção, tomando uma acção particular baseada no estado corrente. As frases que o sistema deve usar como resposta são colocadas juntamente com condições, que devem ser verdadeiras quando a(s) frase(s) é(são) apropriada(s) para execução. Este tipo de sistemas permitem um maior uso de língua natural, o que faz com que o utilizador possa responder de forma flexivel às perguntas colocadas pelo sistema. Permite também que o sistema envie mensagens adequadas à correcção de erros de reconhecimento ou de compreensão. A figura 2.3 mostra o exemplo de um diálogo produzido por um sistema baseado em enquadramentos.

```
SISTEMA: Bom dia. Este é o serviço de informação automático.
         Como posso ajudar?
UTILIZADOR: Quero ir de Lisboa para o Porto.
SISTEMA: Quando deseja ir de Lisboa para o Porto?
UTILIZADOR: Na próxima segunda-feira às 20:00.
SISTEMA: Então quer viajar na sexta-feira,
         31 de Dezembro, às 20:00?
UTILIZADOR: Não, na segunda-feira.
SISTEMA: Então quer viajar na segunda-feira, dia 3 de
         Janeiro às 20:00?
UTILIZADOR: Sim.
SISTEMA: Existe a seguinte ligação: tem o intercidades,
         partida de Lisboa às 20:02, chegada ao Porto ao 24:05.
```

Figura 2.3: Exemplo de diálogo produzido por um sistema baseado em enquadramentos.

O preenchimento múltiplo de espaços de informação permite ao sistema processar informações adicionais dadas pelo utilizador, quer ao nível das respostas quer ao nível das correcções. Deste modo, o tempo de transacção do diálogo pode ser reduzido, resultando num diálogo mais eficiente e natural. Neste tipo de sistemas o utilizador pode controlar o diálogo.

O contexto do sistema que contribui para a determinação da próxima acção do sistema é limitado, baseando-se essencialmente na análise da afirmação do utilizador, em conjunção com um enquadramento a ser preenchido e algumas prioridades para controlo do diálogo. O sistema não providencia um leque alargado de respostas para diferentes conhecimentos de variados utilizadores, não se ajusta às possíveis configurações que possam surgir da mudança dinâmica do mundo e não envolve negociação, nem planeamento ou outro tipo de interacção colaborativa.

2.6.3 Sistemas baseados em agentes

Os sistemas baseados em agentes são desenhados para permitir uma comunicação mais flexível entre o sistema, o utilizador e a aplicação de base, na resolução de um problema específico ou tarefa [2]. Existem muitas variantes de sistemas baseados em agentes, dependendo dos aspectos particulares do comportamento inteligente incluído no sistema. A comunicação de sistemas baseados em agentes é vista como uma interacção entre dois agentes, em que cada um é capaz de raciocinar sobre as suas acções e crenças, e algumas vezes sobre as acções e crenças do outro agente. O modelo de diálogo tem em conta o contexto, como resultado da evolução dinâmica do diálogo. Geralmente, este tipo de sistemas dispõe de mecanismos para detecção e correcção de erros, e pode fazer uso de mecanismos de previsão para interpretar as próximas afirmações do utilizador. Neste tipo de controlo de diálogo, a comunicação é geralmente proporcionada por iniciativa mútua, o que significa que o utilizador pode tomar controlo do diálogo e introduzir novos tópicos. A figura 2.4 mostra o exemplo de um diálogo produzido por um sistema baseado em agentes.

```
SISTEMA: Existe um fio entre o conector 84 e o conector 99?
UTILIZADOR: Não está lá.
SISTEMA: Adiciona um fio entre o conector 84 e o conector 99.
UTILIZADOR: Preciso de ajuda.
SISTEMA: Obtém um fio.
UTILIZADOR: Feito.
SISTEMA: Liga o metal de uma das pontas livres do fio ao
         conector 84.
UTILIZADOR: Como?
SISTEMA: Dobra para trás o conector 84.
```

Figura 2.4: Exemplo de diálogo produzido por um sistema baseado em agentes.

Este tipo de sistemas permite a construção de diálogos mais complexos, que envolva mais do que um simples pedido de um número de parâmetros pré-determinados para permitir o acesso à informação. Este sistema é o mais apto a resolver problemas de colaboração, em que esteja envolvida negociação, e em que tanto o utilizador como o sistema fazem perguntas, pedidos de clarificação, correcções, sugestões, e mudanças de tópico.

Num diálogo de iniciativa mútua, como é o caso, o agente tem de manter e raciocinar sobre os modelos explícitos da tarefa a ser realizada, ou seja, sobre o estado corrente de diálogo, sobre as crenças, e sobre as suas intenções e as de outros agentes. Isto implica recursos mais complexos. Colocar iniciativa mútua num diálogo que envolva negociação e colaboração, requer sofisticadas capacidades de processamento de língua natural. É necessária uma representação semântica mais profunda para interpretar as frases do utilizador. Grande parte das técnicas necessárias ao suporte de sistemas deste género são alvo de estudo.

A tabela da figura 2.5 resume a informação tratada por cada um dos tipos de sistemas de diálogo

para a *entrada*, *verificação*, *modelo de diálogo*, e *modelo do utilizador*.

Controlo do diálogo	Estados	Enquadramentos	Agentes
Entrada	Palavras ou frases	Língua natural com identificação de conceitos	Linguagem natural sem restrições
Verificação	Confirmação explícita – em cada entrada de dados ou no fim da transacção	Confirmação explícita e implícita	Confirmação explícita e implícita
Modelo de diálogo	Estado de informação representada implicitamente nos estados de diálogo. Controlo do diálogo representado explicitamente com o diagrama de estados	Representação explícita de estados de informação. Controlo do diálogo representado com o controlo do algoritmo	História do diálogo. Contexto. Modelo das intenções, objectivos e crenças do sistema
Modelo do utilizador	Modelo simples do utilizador, características ou preferências	Modelo simples do utilizador, características ou preferências	Modelo das intenções, objectivos e crenças do utilizador

Figura 2.5: Resumo do tipo de informação processada pelos três diferentes tipos de diálogo.

2.7 Sistemas de Diálogo

Existem actualmente já desenvolvidos vários sistemas. Esta secção descreve alguns dos principais sistemas actuais e define cada um deles de acordo com os tipos enumerados na secção anterior.

De entre os vários projectos que foram e estão a ser desenvolvidos pelo mundo inteiro na área de sistemas de diálogo, existem três que se destacam:

- **CMU Communicator:** o *Communicator System* da ”Carnegie Mellon University”(CMU) é um sistema de diálogo baseado em enquadramentos, que é usado para aceder a horários de aviões, para alugar quartos de hotel e carros. Os utilizadores usam língua natural (em inglês) para interagir com o agente de viagens. O agente acede posteriormente à internet para recolher informação [29];
- **TRIPS:** TRIPS, *The Rochester Interactive Planning System* [28], é o último de uma série de protótipos de agentes colaborativos de planeamento, desenvolvidos no Departamento de Ciências Computacionais na Universidade de Rochester. O objectivo do projecto é produzir um assistente de planeamento inteligente que interage com um gestor humano usando uma combinação de língua natural e gráficos. O diálogo providencia o contexto para a interpretação das frases e das acções humanas, bem como a estrutura necessária para decidir uma resposta. Este sistema é baseado em agentes;

- **Collagen:** COLLaborative AGENt [30] é uma plataforma[1] para a construção de agentes colaborativos. Um agente colaborativo é um programa que ajuda o utilizador a resolver problemas, especialmente em domínios não familiares ou complexos, corrigindo erros e sugerindo as próximas acções. Um agente colaborativo pode ser adicionado a uma interface gráfica já existente, como um simulador, ou integrado no desenho de um novo dispositivo de hardware. Este projecto está a ser desenvolvido para a companhia norte-americana Mitsubishi Electric Research Laboratories, e é baseado em agentes [20].

2.7.1 CMU Communicator

O software utilizado na construção de sistemas está organizado por uma colecção de módulos e utilitários que se denomina "Conversational Agent Toolkit"(CAT). Este software, ainda em desenvolvimento, já contém muita da funcionalidade necessária para construir sistemas de diálogo. Este sistema é financiado pela DARPA[2].

Módulos Disponíveis

O CMU Communicator dispõe de módulos que funcionam como servidores e que comunicam uns com os outros (ver figura 2.6):

1. Áudio;
2. Reconhecimento de Fala;
3. Interpretador semântico;
4. Gestor de Diálogos;
5. Geração de Língua Natural;
6. Geração de Fala;
7. Base de Dados.

Gestor de Diálogos

O gestor de diálogo recebe informação do interpretador e envia resultados para o gerador de língua natural. Também envia e recebe informação da aplicação ou base de dados. É função deste módulo

[1] Ambiente de suporte ao desenvolvimento de agentes colaborativos.

[2] DARPA, do inglês Defense Advanced Research Projects Agency, é o centro de pesquisa e organização de desenvolvimento para o Departamento de Defesa dos Estados Unidos da América.

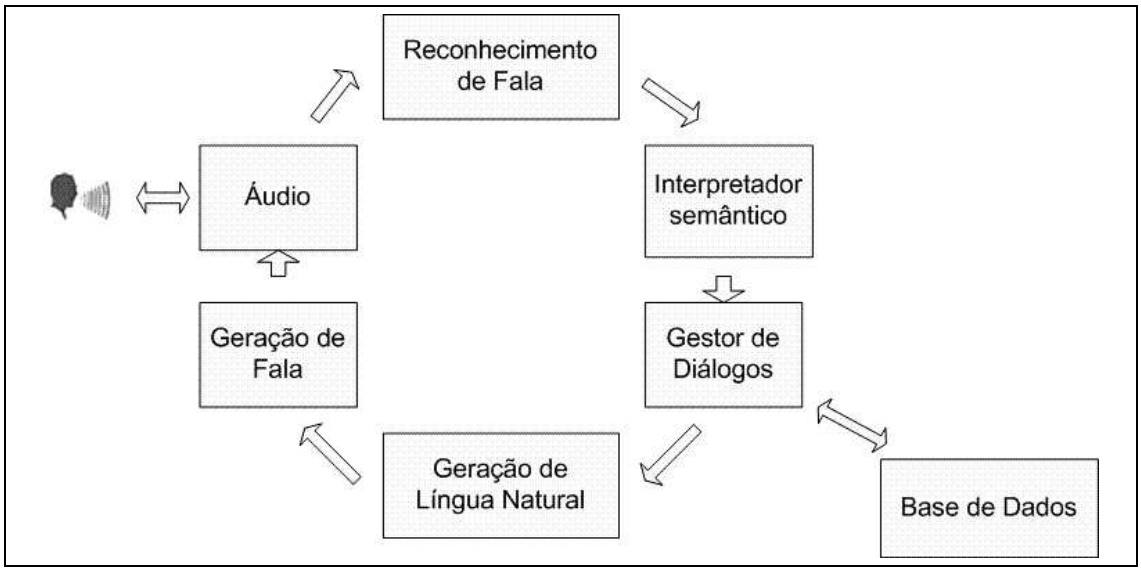

Figura 2.6: Arquitectura do CMU Communicator.

interpretar uma frase e integrá-la no contexto do diálogo, bem como controlar a interacção com o utilizador. O gestor está configurado para operar como um servidor na arquitectura *Galaxy*[3].

A estrutura básica para representação da informação são enquadramentos. Cada enquadramento tem um nome e um conjunto de espaços de informação. Um espaço de informação está destinado a ser preenchido com um valor específico. Cada espaço de informação tem um nome e pode conter vários outros espaços. Após ser extraída do interpretador, a informação é armazenada directamente em enquadramentos.

Para desenvolver uma aplicação, tem de se criar um ficheiro de tarefas. Este ficheiro contém a estrutura de enquadramentos que definem o conhecimento do domínio da aplicação a construir. O ficheiro tem também modelos para fazer pedidos, verificação de informação, e geração de perguntas em linguagem de acesso a bases de dados (no caso *SQL*).

O sistema de diálogos funciona como um sistema de produção e é responsável por diversas funções:

1. Processar a informação vinda do Interpretador;
2. Gerar questões de consulta à base de dados;
3. Gerar língua natural.

1. Processar a informação vinda do Interpretador

Nesta fase, o gestor de diálogo coloca a informação vinda do interpretador em enquadramentos.

[3]Galaxy Communicator é uma arquitectura *open source* para construção de sistemas de diálogo [31] [23]. Foi financiada pelo DARPA.

Para isso: (i) avalia; (ii) clarifica; (iii) mapea para a forma canónica; (iv) resolve anáforas; (v) executa funções de contexto; e (vi) integra os novos enquadramentos no contexto.

No âmbito desta tese, não serão detalhados todos os passos supra-mencionados, mas apenas aqueles que se consideram mais relevantes. Destacam-se assim o terceiro e quarto passos:

- **Mapeamento para a forma canónica:** Este mapeamento acontece quando o gestor de diálogo recebe algo que não está na forma de que o sistema precisa. Isto porque o sistema precisa de dados concretos, e muitas vezes a informação é recebida de forma implícita. Ex.: "amanhã" deverá ser convertido para uma data;

- **Resolução de elipses:** Quando a informação dada não está totalmente especificada, há que preencher os espaços de informação adequados face à nova informação. Este problema é resolvido com um valor por omissão, com informação do contexto, ou em último caso, com a elaboração de uma pergunta ao utilizador.

Uma estrutura é criada para cada instância específica do enquadramento, nas quais os dados extraídos são armazenados. Existem duas listas globais de enquadramentos: uma contém os enquadramentos activos no contexto corrente. A outra contém os enquadramentos extraídos da frase corrente. Depois do mapeamento para a forma canónica ter sido aplicado, os valores do interpretador são extraídos para uma sequência de enquadramentos que são colocadas na respectiva lista. As duas listas são posteriormente fundidas. Fica-se assim com um conjunto de enquadramentos que contém a informação criada até agora na interacção com o utilizador.

Depois da extracção e fusão da informação com o contexto, uma função é chamada para determinar a próxima acção do sistema. Esta função examina o contexto e toma uma acção baseada em regras.

2. Gerar questões de consulta à base de dados

As perguntas *SQL* são geradas da mesma forma que as respostas do sistema para o utilizador. As perguntas estão associadas a espaços de informação do enquadramento no ficheiro de tarefas, e são construídas a partir do preenchimento desses espaços.

3. Gerar língua natural

O gestor de diálogos gera principalmente quatro tipos de saída para o utilizador:

- faz perguntas (para completar o pedido);

- fornece informação (respondendo ao pedido);

- envia mensagens de erro (relatando alguma irregularidade);

- envia mensagens de ajuda (guiando o utilizador na tarefa).

Qualquer espaço de informação no enquadramento pode ter um modelo associado. Este modelo é usado para gerar uma questão relacionada com a informação específica do espaço de informação. Assim, para gerar a resposta a enviar ao utilizador, o gestor preenche o modelo a partir do contexto, e envia essa questão para o servidor de língua natural. A estratégia do gestor de diálogos é fazer primeiro perguntas relativas aos espaços de informação gerais e só depois aos mais específicos. Os modelos são especificados no ficheiro de tarefas [**?**].

A informação contextual do sistema está representada como uma lista de enquadramentos. O gestor não armazena o historial do diálogo e tem uma forma rudimentar (baseado em regras do tipo *if than else*) para decidir a próxima acção do sistema. A forma como a informação é representada, aliada à forma como são geradas as questões para o utilizador e as questões SQL simplificam o acesso e a manipulação da informação, o que constitui a mais valia deste sistema.

2.7.2 TRIPS

O sistema TRIPS é composto por uma rede complexa de agentes. Cada agente tem um papel linguístico ou de planeamento bem definido na interpretação do diálogo. A arquitectura é dividida em 3 áreas de funcionalidades (ver figura 2.7):

- Interpretação (realizada pelo *Gestor de Interpretações*);
- Comportamento (realizada pelo *Gestor de Comportamentos*);
- Geração (realizada pelo *Gestor de Geração*).

Cada uma das componentes é executada independente e assincronamente, o que significa que o sistema pode interpretar entradas do utilizador, interactuar com o mundo e planear uma resposta baseada em entradas/saídas prévias, ao mesmo tempo. A informação de entrada do sistema é processada e enviada para o *Gestor de Interpretações*, o qual interpreta a informação como um acto de resolução de problemas. O resultado é enviado para o *Gestor de Comportamentos*, que decide como responder a esse acto de resolução de problemas. O *Gestor de Comportamentos* informa o *Gestor de Geração* do seu estado, e este planeia um acto linguístico para comunicar a intenção daquele ao utilizador.

Gestor de Interpretações

Este módulo é responsável por determinar a interpretação mais correcta a dar à frase dita pelo utilizador. Para isso, constrói uma obrigação de discurso que corresponde ao *Acto de Resolução de Problemas Colaborativo* (ARP) pretendido. Este ARP é encontrado através de decisões específicas do *Gestor de Tarefas* baseado no *Modelo Abstracto de Resolução do Problema*, modelo este que está na base do planeador do *Gestor de Tarefas*. Depois de ter criada a obrigação de discurso,

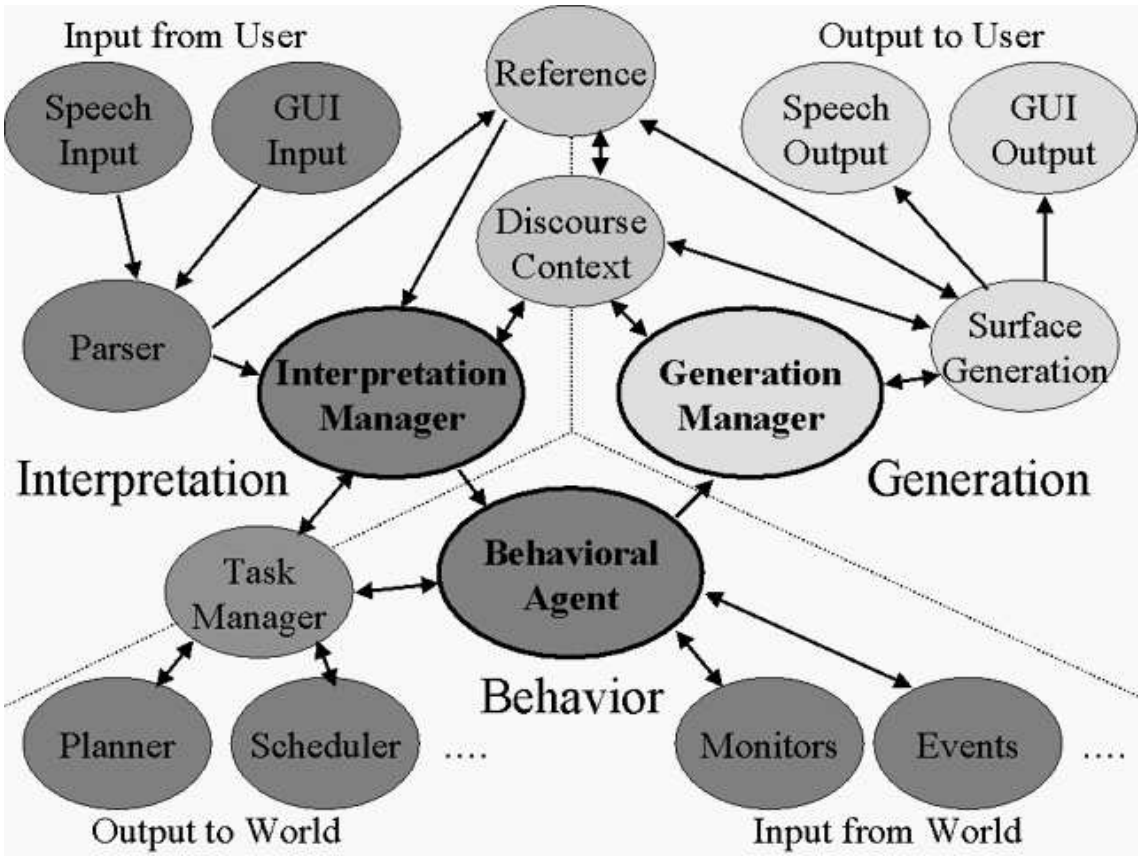

Figura 2.7: Arquitectura do sistema TRIPS.

o *Gestor de Interpretações* vai decidir quais são as obrigações do sistema provenientes do actual estado do diálogo com base no ARP.

Gestor de Comportamentos

Este módulo é responsável pelo comportamento geral do sistema na resolução de problemas. É a componente que torna o TRIPS um sistema mais realístico, oferecendo capacidades de iniciativa mútua. Este agente tem as suas próprias prioridades e intenções. O comportamento do agente baseia-se em três aspectos fundamentais:

- Na interpretação das afirmações do utilizador, em termos do problema a resolver;
- Nos objectivos e obrigações do sistema utilizados na resolução de problemas;
- Nos eventos que possam ocorrer exteriormente ao sistema.

A grande maioria, senão todos os sistemas de diálogo têm uma forma de resolver os dois primeiros pontos, quer através do uso de uma linguagem que é reconhecida pelas componentes em causa, quer através da representação explícita de regras a ter no comportamento do agente. Este agente

difere do comportamento típico pela sua habilidade para lidar com eventos ocorridos externamente. Para isso, reage aos mesmos e ordena-os por prioridades, modificando os seus próprios objectivos e obrigações. O agente pode, assim, receber notificações de eventos no mundo e decidir se quer, ou não, comunicá-las ao utilizador ou se quer adoptar novas obrigações.

Quando o agente responsável pelo comportamento recebe uma notificação de um problema a resolver, pode tomar uma de quatro acções:

- No caso de querer resolver o problema imposto através da componente de resolução de tarefas (*Gestor de Tarefas*), envia a solução encontrada para o agente de geração de mensagens (*Gestor de Geração*). Neste caso, aguarda uma nova mensagem do agente de interpretação (*Gestor de Interpretações*) para confirmar;

- Em caso de dúvida acerca do problema imediato a resolver, envia para o componente de geração de mensagens um pedido de clarificação. Neste caso, o objectivo inicial é retido enquanto se espera por uma resposta;

- Caso não encontre uma solução para o problema, envia para o agente de geração de mensagens uma notificação de falha e abandona, pelo menos temporariamente, essa obrigação;

- No caso de, fazendo uso da sua lista de prioridades, o agente decidir ignorar a questão, este procura resolver aquela que considere mais importante.

Gestor de Geração

Este módulo coordena as actividades de geração. Tem de encontrar a melhor forma de expressar o que o *Gestor de Comportamentos* pretende comunicar. O *Gestor de Geração* planea a apresentação de conteúdos. Para isso, recebe do *Gestor de Comportamentos* actos de resolução de problemas (ARP's) e obrigações de discurso do *Gestor do Contexto*. A tarefa do *Gestor de Geração* é sintetizar estas fontes de informação e formular um plano para a sua apresentação (sequência de actos de discurso).

Porque este módulo opera assincronamente com o *Gestor de Interpretações*, pode estar em planeamento contínuo. Por exemplo, é informado quando o utilizador termina de falar, e pode planear actos de mudança de turno de fala mesmo sem qualquer informação do *Gestor de Interpretações* ou do *Gestor de Comportamentos*. Em casos de acções de menor complexidade o *Gestor de Geração* usa simplesmente regras baseadas em pares adjacentes (e.g. saudações). Noutros casos pode precisar de informação do *Gestor de Comportamentos* de forma a satisfazer as obrigações de discurso. Pode também receber objectivos do *Gestor de Comportamentos* que pode tentar planear mesmo sem informação de obrigações de discurso (e.g. quando algo no mundo se altera e o *Gestor de Comportamentos* pretende notificar o utilizador). Este módulo tem acesso ao *Gestor do Contexto* bem como ao *Gestor de Tarefas*.

Gestor de Tarefas

Este módulo detém o conhecimento das tarefas específicas do domínio e disponibiliza serviços aos componentes independentes do domínio. É responsável por:

- Fazer o reconhecimento de intenções;
- Mapear actos genéricos de resolução de problemas em acções específicas do domínio;
- Responder a perguntas sobre objectos específicos da tarefa/domínio e das suas funções respectivas.

As tarefas específicas do domínio são representadas sob a forma de uma estrutura hierárquica de planos. Esta estrutura tem informação de:

- Objectivos do domínio;
- Decomposição de objectivos (sub-objectivos);
- Conjunto de acções que são usadas como soluções aos objectivos.

É importante guardar as etapas do plano: não só a solução, mas também a estrutura de objectivos e sub-objectivos. Em planeamento de iniciativa mútua é necessário mais informação. Deve ser possível ao utilizador sugerir e comparar soluções alternativas para um objectivo simples. O sistema deve permitir ao utilizador corrigir acções e voltar para soluções antigas.

Gestor do Contexto

Este módulo representa toda a informação necessária para coordenar o comportamento conversacional do sistema. É responsável por manter e gerir toda a informação sobre o discurso, incluindo o historial do diálogo (actos de fala do utilizador, respostas geradas, etc.) e o registo de tarefas (informação sobre mudança de turno, obrigações do discurso, etc.). Este módulo armazena também a informação de todas as tarefas realizadas no passado e aquelas que é necessário completar para realizar a tarefa em execução. Esta informação pode ser representada internamente como um enquadramento ou como um grafo de estados.

O *Gestor do Contexto* tem de armazenar:

- Informação suficiente para interpretar anáforas, utilizando um modelo com todas as entidades encontradas durante o diálogo consideradas importantes;
- A estrutura e a interpretação da frase proferida imediatamente antes da frase actual, de forma a permitir a resolução de elipses e questões de clarificação;

- O estado corrente da mudança de turno no diálogo;
- O historial do diálogo com as interpretações das frases proferidas no diálogo até ao momento actual;
- As obrigações do discurso actuais, necessárias para responder à última frase proferida pelo utilizador.

Gestor de Referências

Este módulo é responsável por resolver ou criar expressões contidas nas frases que representam ou referenciam objectos pertencentes ao diálogo. Essas expressões podem ser anáforas, mas também podem ser determinantes ou pronomes. O *Gestor de Referências* faz o mapeamento dessas expressões para o objecto que eles representam e vice-versa. Esta função é muito importante pois o *Gestor de Interpretações* tem que identificar todos os objectos referidos na frase, algo que é dificultado pelo uso deste tipo de expressões.

Assim, o *Gestor de Referências* é necessário nas seguintes situações:

- Quando o *Interpretador* detecta a existência de expressões referenciais, o *Gestor de Referências* resolve as referências, identificando o objecto correspondente a cada uma;
- Quando uma estrutura é inserida no *Gestor do Contexto* (normalmente obrigações de discurso), é necessário identificar todos os objectos referidos nessa estrutura. O *Gestor de Referências* adiciona os novos objectos a uma lista de referências encontradas ao longo de todo o diálogo, guardando a ordem de ocorrência de cada um;
- Quando o *Gerador de Superfície* produz a frase a enviar ao utilizador, o *Gestor de Referências* reformula-a de forma a utilizar referências, tornando a comunicação mais natural. Para isso, as expressões referenciais relativas aos objectos são substituídas por anáforas, pronomes, etc.

Gerador de Superfície

O *Gerador de Superfície* utiliza componentes baseados em templates, em gramáticas e em componentes de selecção e coordenação de saída. Quando necessário, realiza mudanças de turno no diálogo, inferência e actos de fala em paralelo e em tempo real. Se os actos de fala forem produzidos com sucesso, este módulo actualiza o *Gestor do Contexto* e informa o *Gestor de Geração*.

De um ponto de vista abstracto, a construção de frases consiste em três decisões: (i) qual a informação que deve ser incluída; (ii) qual a estrutura a dar à informação; e (iii) qual o formato da mensagem, i.e. a escolha de palavras e a estrutura sintáctica. Sob a perspectiva de construção de texto, quatro diferentes categorias de contexto podem estar envolvidas:

- **Texto inalterado:** partes da mensagem que estão sempre presentes no texto de output;
- **Informação directamente disponível:** informação que foi disponibilizada através de uma base de dados ou de uma base de conhecimento;
- **Informação computável:** informação que é derivada de alguma computação ou raciocínio (e.g., o número de registos encontrados numa base de dados para percursos entre duas cidades);
- **Informação extra:** informação que não está disponível nos dados mas que a complementa. Isto é comum em textos produzidos por humanos. Por exemplo, informação extra que um percurso não pode ser realizado porque a estrada está bloqueada devido a um acidente.

2.7.3 COLLAGEN - A COLLaborative AGENt

O CollAgen começou a ser desenvolvido baseado na evidência de que a aplicação da teoria de discurso traria maior potencial ao desenho de sistemas interactivos computorizados. O objectivo máximo seria desenvolver uma colecção de sub-rotinas, estruturas de dados e especificações contendo capacidades de discurso, de forma a que estas pudessem vir a ser facilmente utilizadas por um programador em qualquer sistema interactivo. As capacidades de discurso referidas são traduzidas em quatro pontos:

- Clarificação da acção de comunicação, presente ou anterior;
- Voltar a uma tarefa que tinha sido abandonada, enquanto mais informação vai sendo recolhida;
- Iniciativa mútua (o sistema sugere uma acção para ser realizada a seguir);
- Sumário do estado actual do processo de colaboração que está a decorrer.

O CollAgen foi desenvolvido e usado para construir um agente de software que colabora com o utilizador através de uma manipulação directa numa interface gráfica seguindo regras e convenções de um discurso humano. Um dos principais resultados é uma história de interacção que é segmentada de acordo com a estrutura dos objectivos do utilizador e do agente, sem este último necessitar de entender língua natural.

A figura 2.8 mostra como o CollAgen dispõe a interacção entre o utilizador e o agente. Uma interacção começa com a chegada de uma declaração ou acção observada (do utilizador ou do agente) ao módulo de interpretação do discurso. Este módulo, além de reconhecer os planos, actualiza o estado de discurso, fazendo com que uma nova agenda seja computada pelo módulo de geração do discurso. No caso mais simples, o agente responde escolhendo e executando uma entrada da nova agenda, através de uma declaração ou uma açção, o que gera uma nova entrada para o módulo de interpretação do discurso. Num sistema que não trabalha com língua natural,

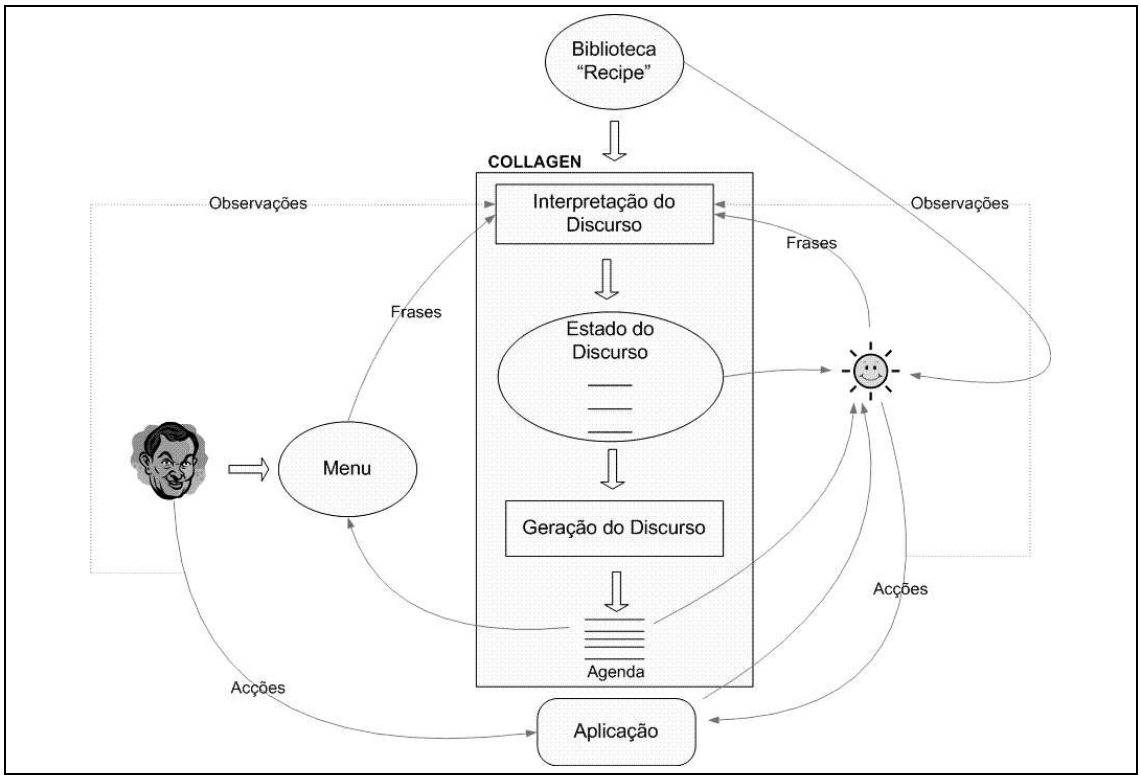

Figura 2.8: Arquitectura do COLLAGEN.

um subconjunto da agenda é apresentado ao utilizador sob a forma de menu. Como este sistema deverá suportar iniciativa mútua, o utilizador pode a qualquer momento produzir uma declaração, escolhendo no menu, ou executar uma determinada acção, produzindo uma nova entrada para o módulo da interpretação do discurso. Como se pode ainda ver na figura, a aplicação específica ao agente tipicamente faz perguntas à aplicação e consulta o estado do discurso. Recebe ainda informação da *biblioteca recipe*. Um *recipe* é um recurso utilizado para derivar uma sequência de passos para atingir um dado objectivo (o objectivo do *recipe*). Existem algoritmos independentes da aplicação que poderiam ser utilizados para este fim, como o algoritmo de *divide and conquer*, mas o utilizado no CollAgen é algo dependente da aplicação.

Gestor de Diálogos

O processamento de discurso no CollAgen possui três componentes interactivas:

- Uma estrutura linguística que inclui um grupo de declarações em segmentos que é representada por uma linguagem de discurso;
- Um estado atencional que captura a mudança de foco de atenção por parte dos participantes. É representado por uma pilha de segmentos de discurso;
- Uma estrutura intencional que corresponde aos planos parciais de cada um dos participantes, que representada por uma *recipe tree*, é uma reimplementação do algoritmo rgraph

de Lochbaum [18]. Cada nó nesta *recipe tree* corresponde a um objectivo comum. Os filhos do nó são actos que contribuem para o alcançar.

2.8 Sumário

Este capítulo contextualizou o surgimento dos sistemas de diálogo numa panorâmica histórica, e definiu cada um dos componentes usados no desenvolvimento de um sistema de diálogo. As funcionalidades relevantes de um gestor de diálogo descritas neste capítulo servem de base à construção de qualquer gestor, e como tal, também o gestor de diálogos desenvolvido no decorrer deste trabalho segue as mesmas bases. Os actos de fala são muito importantes para uma série de aplicações. No contexto desta tese, representam as intenções do utilizador de um sistema de diálogo. O capítulo apresentou também os diferentes tipos de sistemas existentes, bem como os sistemas desenvolvidos por outros laboratórios. O sistema de diálogos descrito nesta dissertação é baseado em enquadramentos.

Capítulo 3

Gestor de Diálogos

Neste capítulo é feita uma descrição pormenorizada das estratégias seguidas na implementação do gestor de diálogos do L^2F. O capítulo contém duas partes distintas: na primeira são definidos os conceitos de domínio e de dispositivo, e descrito o módulo responsável pela gestão da informação envolvida; na segunda parte apresenta-se em pormenor o funcionamento de cada um dos módulos que compõem o gestor de diálogos.

Um gestor de diálogos é uma componente fundamental de um sistema de diálogos cuja função principal é controlar o fluxo do diálogo. A arquitectura do sistema de diálogos usada no decorrer deste trabalho é apresentada na figura 3.1. O módulo *Gestor de Entradas e Saídas* é responsável pela gestão de informação respeitante às entradas e saídas do gestor de diálogo. Incluem-se neste módulo os sub-módulos de reconhecimento e síntese de fala. O módulo *Gestor de Serviços* é responsável pela gestão de toda a informação relacionada com os domínios e dispositivos acoplados ao sistema.

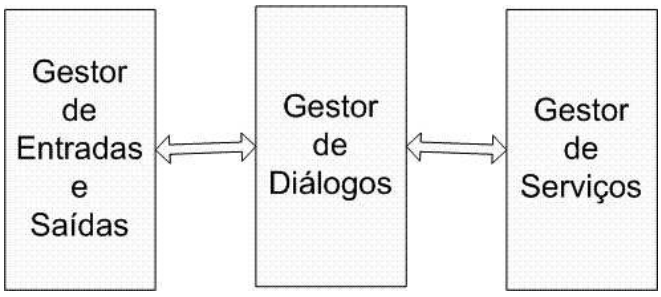

Figura 3.1: Arquitectura do sistema de diálogos.

Os módulos do gestor de diálogos desenvolvidos no âmbito desta tese são (ver figura 3.2): (i) *Reconhecedor da linguagem* (ver secção 3.5); (ii) *Gestor de interpretações* (ver secção 3.6); (iii) *Gestor do contexto* (ver secção 3.7); (iv) *Gestor de tarefas* (ver secção 3.4); (v) *Gestor de comportamentos* (ver secção 3.8); (vi) *Gestor de geração* (ver secção 3.8); e (vii) *Gerador de superfície* (ver secção 3.9).

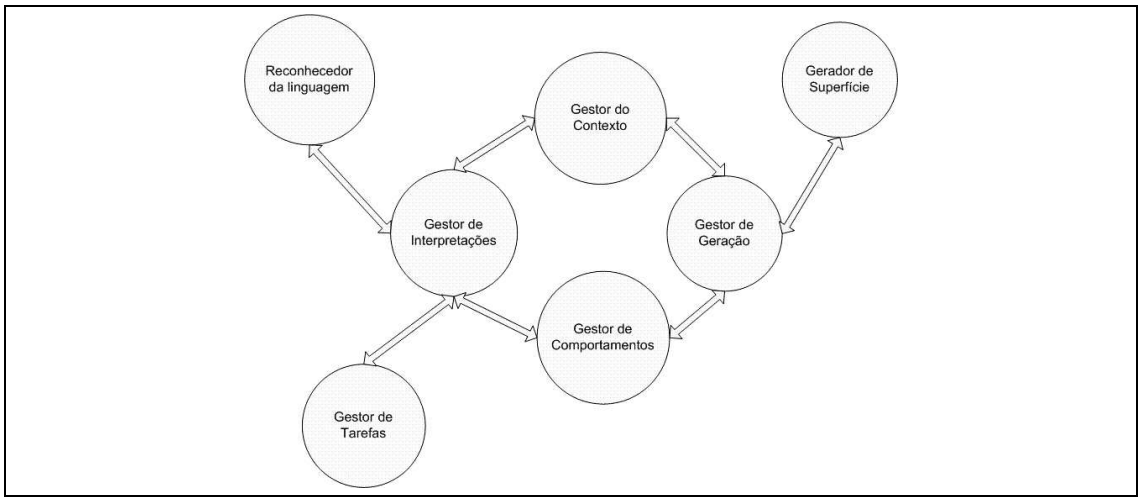

Figura 3.2: Arquitectura do Gestor de Diálogos.

3.1 Domínio

Um domínio representa um conjunto de dispositivos com as mesmas características. É composto por:

- **Nome:** designação do domínio;
- **Enquadramento:** matriz com espaços de informação. Representam um conjunto de conceitos aos quais vão estar associados objectos (palavras ou termos) relativos ao domínio em causa. Estes conceitos permitem identificar quaisquer palavras ou termos que estejam referenciados por esse domínio. Os espaços de informação são preenchidos à medida que o diálogo acontece;
- **Regras Compostas ou de Geração das Frases:** as regras compostas traduzem um conjunto de frases que vão ser usadas pelo sistema na comunicação das suas intenções ao utilizador. Estas frases são baseadas em modelos e são instanciadas em tempo de execução com informação contextual. O uso de modelos permite que se possa usar uma mesma frase de geração para os vários dispositivos que pertençem ao domínio.

O sistema de diálogos foi testado com dois domínios principais: (i) autocarros; e (ii) domótica. O domínio dos autocarros engloba um único dispositivo que representa uma máquina de vender bilhetes. Para que o diálogo de controlo da máquina de vender bilhetes seja possível, foi necessária a definição de um enquadramento como o apresentado na figura 3.3. O domínio da domótica engloba um conjunto de dispositivos relacionados com o interior de uma casa e passíveis de serem controlados. Entre outros, incluem-se a luz, o ar condicionado, e a aparelhagem.

Um espaço de informação tem obrigatoriamente um tipo, um valor de referência e um nome (ver figura 3.3). *Type* representa o tipo do objecto a colocar no espaço de informação e com o qual são

```
<enquadramento>
   <description>
     <enquadramento_name>Autocarros</enquadramento_name>
       <slot type="string" key="1">Action</slot>
       <slot type="string" key="2">Target</slot>
       <slot type="string" key="3">DiaSemana</slot>
       <slot type="string" key="4">CidadeOrigem</slot>
       <slot type="string" key="5">CidadeDestino</slot>
       <slot type="string" key="6">HoraPartida</slot>
       <slot type="string" key="7">HoraChegada</slot>
   </description>
</enquadramento>
```

Figura 3.3: Enquadramento do domínio dos Autocarros.

realizadas as comparações. *Key* é o valor usado para referenciar o espaço de informação. Estas referências são fundamentalmente usadas na descrição dos serviços de um dispositivo (ver figura 3.6).

Quando se define um enquadramento, alguns aspectos importantes devem ser considerados:

- Todos os objectos (palavras ou termos) relevantes para o domínio em questão têm de estar englobados ou devidamente categorizados num dos espaços de informação do enquadramento. O sistema apenas reconhecerá objectos que estejam incluídos num destes espaços de informação. No domínio dos autocarros:

  1. **Action:** representa as acções que podem ser executadas. Ex.: “comprar”, “consultar”;
  2. **Target:** representa os conceitos sobre o qual as acções podem ser executadas. Ex: “bilhete”, “horário”;
  3. **DiaSemana:** representa os dias da semana possíveis. Ex.: “Segunda-feira”, “Terça-feira”;
  4. **CidadeOrigem:** representa as cidades de origem possíveis. Ex.: “Lisboa”, “Porto”;
  5. **CidadeDestino:** representa as cidades de destino possíveis. Ex.: “Coimbra”, “Lisboa”;
  6. **HoraPartida:** representa as horas de partida possíveis. Ex.: “19:00”, ”21:30”;
  7. **HoraChegada:** representa as horas de chegada possíveis. Ex.: “20:00”, “20:30”.

  Os conceitos relevantes ou espaços de informação devem ser decididos com base em estudos prévios realizados sobre diálogos reais no domínio em causa. Para o domínio dos autocarros, os espaços de informação descritos acima permitem que o tipo de frases a utilizar no sistema seja mais complexo. Ex.: “Quero comprar um bilhete de Lisboa para Coimbra, às 19:00 de Segunda-feira”.

- As funcionalidades de cada um dos dispositivos do domínio vão ser disponibilizadas pela combinação lógica dos espaços de informação do enquadramento, bem como dos valores associados ao mesmos. A compra de um bilhete (uma das funcionalidades) só é possível de realizar porque existe um serviço de um dispositivo pertencente ao domínio dos autocarros que diz que se pode comprar um bilhete de uma cidade de origem para uma cidade de destino, a determinadas horas, de um determinado dia da semana (ver secção 3.2).

```
<rule id="3" uttgenerationtype="requestslot"
             uttgenerationslot="cidade">
  <one-of>
    <item>Qual é a cidade? </item>
    <item>Qual é a cidade?, às [$horapartida]? </item>
    <item>Qual é a cidade?, para chegar às [$horachegada]?
          </item>
    <item>Qual é a cidade?, [$#no $diasemana] [$diasemana],
          às [$horachegada]? </item>
  </one-of>
</rule>
```

Figura 3.4: Regra de geração para o espaço de informação “Cidade”.

- A decisão pelo uso de mais ou menos espaços de informação no enquadramento deverá também ser tomada com base na especificidade das frases que se quer na geração. A partilha de espaços de informação por conceitos diferentes obriga ao uso de frases mais abstractas, ao passo que o uso de um espaço de informação para um conceito específico permite o uso de frases mais específicas. Por exemplo, poder-se-ia usar no enquadramento apenas um espaço de informação para a representação de qualquer cidade, origem ou destino. A regra apresentada na figura 3.4 seria usada para pedir ao utilizador uma cidade de origem ou uma cidade de destino. As frases desta regra são mais genéricas, o que torna difícil a sua compreensão, e aumenta a probabilidade de erros no diálogo. A descrição actual do enquadramento do domínio dos Autocarros faz uso de dois espaços de informação, um para cada tipo de cidade possível. São utilizadas duas regras para questionar o utilizador acerca de uma cidade. Uma regra para a cidade de origem (ver figura 3.5) e uma regra para a cidade de destino. Porque existe mais especificidade na construção do texto, o diálogo é mais natural e existe maior facilidade de compreensão por parte do utilizador.

O uso destas frases pelo sistema vai ser analisado em pormenor ainda neste capítulo (ver secção 3.9).

```
<rule id="3" uttgenerationtype="requestslot"
             uttgenerationslot="cidadeorigem">
  <one-of>
    <item>De onde quer partir? </item>
    <item>Qual a cidade de origem? </item>
    <item>De onde deseja partir? </item>
    <item>Quer partir de que cidade? </item>
    <item>Quer partir de que cidade,
          para chegar a [$cidadedestino]? </item>
    <item>Quer partir de que cidade,
          às [$horapartida]? </item>
    <item>Quer partir de que cidade,
          para chegar às [$horachegada]? </item>
    <item>Quer partir de que cidade,
          para chegar a [$cidadedestino], às [$horachegada]?
          </item>
    <item>Quer partir de que cidade,
          [$#no $diasemana] [$diasemana],
          para chegar a [$cidadedestino], às [$horachegada]?
          </item>
  </one-of>
</rule>
```

Figura 3.5: Regra de geração para o espaço de informação *CidadeOrigem*.

3.2 Dispositivo

Um dispositivo é um objecto real cujas funcionalidades se pretendem controlar. A representação de um dispositivo no sistema de diálogos é composta por:

- **Nome do Domínio:** referência para o domínio à qual o dispositivo pertence. Não pode haver um dispositivo que não tenha um domínio associado;
- **Serviços:** definição das regras de execução dos serviços proporcionados pelo dispositivo. A execução de um serviço é determinada através de combinações lógicas para os valores dos espaços de informação;

  As componentes usadas na descrição de um serviço são:

  - **service:** nome do elemento principal de um serviço;
  - **name:** atributo do elemento principal que indica o nome do serviço;
  - **label:** elemento que contém uma descrição mais pormenorizada do serviço;
  - **params:** elemento que contém um conjunto de parâmetros (*params*);
  - **param:** elemento representativo de um parâmetro;

```
<service name="comprar bilhete">
  <label>Este serviço é usado para comprar bilhetes no
         domínio dos autocarros</label>
    <params>
      <param slotkey="1" value="comprar"></param>
      <param slotkey="2" value="bilhete"></param>
      <param slotkey="3"></param>
      <param slotkey="4"></param>
      <param slotkey="5"></param>
      <or>
        <param slotkey="6"></param>
        <param slotkey="7"></param>
      </or>
    </params>
    <execute>
      <name>comprarBilhete</name>
      <success></success>
      <failure></failure>
    </execute>
</service>
```

Figura 3.6: Serviço de compra de bilhetes para o dispositivo da venda de bilhetes de autocarro.

- **execute:** elemento que contém dados relativos à pós-condição do serviço, ou seja, o que irá ser feito caso o serviço seja seleccionado para execução.

Um parâmetro (*param*) é composto por:

- **slotkey:** atributo do elemento principal que serve de referência para um espaço de informação do domínio à qual o dispositivo pertence;
- **value:** atributo opcional do elemento principal. Quando usado, exige um valor específico para o espaço de informação na execução do serviço.

Na descrição de um serviço também podem ser usados os elementos *And* e *Or*. Tanto um como o outro englobam parâmetros[1]. Por exemplo, o serviço "comprar bilhete" (ver figura 3.6), exige que a execução do mesmo só aconteça caso:

1. O espaço de informação com identificador (*slotkey*) igual a "1" (*Action*) esteja preenchido com *comprar*;
2. O espaço de informação com identificador igual a "2" (*Target*) esteja preenchido com *bilhete*;
3. Se forneça valores possíveis para os espaços de informação de identificadores "3" (*DiaSemana*), "4" (*CidadeOrigem*), e "5" (*CidadeDestino*).

[1]Por omissão, o elemento *and* é utilizado para relacionar os parâmetros.

4. Se forneça um de dois tipos de valores possíveis para os espaços de informação de identificador "6" (*HoraPartida*), "7" (*HoraChegada*).

- **Regras Básicas ou de Reconhecimento dos Objectos:** Estas regras são usadas para mapear objectos (palavras ou termos) com os espaços de informação usados na descrição do enquadramento do domínio à qual pertence o dispositivo. Apenas desta forma se pode afirmar que "consultar" é uma *Action*, ou que "Sexta-feira" é um *DiaSemana*. Este mapeamento é fundamental para a execução de um serviço, e consequentemente, para o funcionamento do sistema.

  É util que os valores possíveis para os espaços de informação possam ser extraídos a partir de diferentes fontes. Por exemplo, os valores possíveis para a *CidadeOrigem* são extraídos de uma base de dados com informação horária[2].

  Para exemplificar as regras XML de reconhecimento dos objectos presentes na descrição de um dispositivo, utiliza-se aqui o domínio da Domótica (ver figura 3.7). Cada regra (representada na figura por *rule*) vai indicar um conjunto de valores possíveis para um único espaço de informação. Cada valor vem enquadrado num elemento de nome *item*. Assim, a primeira regra indica quatro valores possíveis para o espaço de informação *Action*: *ligar*, *desligar*, *aumentar* e *diminuir*, ao passo que a segunda regra indica apenas um valor para o espaço de informação *Target*: *luz*[3].

3.3 Gestor de Serviços

Para que haja apenas um ponto central de comunicação do sistema de diálogos com os dispositivos, foi criado um único módulo responsável por isso. O *Gestor de Serviços* disponibiliza toda a informação relacionada com os dispositivos. Para tal, mantém uma descrição de todos estes, controla quais estão acessíveis em cada momento, e em que circunstâncias os serviços fornecidos podem ser executados. Todos os outros módulos, quando necessitam de informação relacionada com um ou mais dispositivos, fazem pedidos a este módulo, que responde de acordo com os parâmetros de entrada fornecidos nos mesmos.

O *Gestor de Serviços* providencia:

- um componente independente do sistema que disponibiliza toda a informação relacionada com os domínios;

- não há mistura entre as partes linguísticas do sistema e a informação sobre os domínios;

[2] O mesmo acontece para os espaços de informação *DiaSemana*, *CidadeDestino*, *HoraPartida* e *HoraChegada*, no domínio dos autocarros.

[3] Apesar de terem o mesmo nome, os espaços de informação *Action* e *Target* deste exemplo são diferentes dos utilizados nos exemplos prévios. Os últimos fazem parte do enquadramento do domínio da Domótica, e não do enquadramento do domínio dos Autocarros.

```
<grammar>
  <simplerules>

    <rule id="action">
      <one-of>
        <item>ligar</item>
        <item>desligar</item>
        <item>aumentar</item>
        <item>diminuir</item>
      </one-of>
    </rule>

    <rule id="target">
      <one-of>
        <item>luz</item>
      </one-of>
    </rule>
  ...
  </simplerules>
</grammar>
```

Figura 3.7: Regras de reconhecimento para o domínio da Domótica.

- porque a informação sobre os domínios está centralizada, é mais fácil definir camadas de raciocínio usando essa informação e extrair conhecimento a partir dela;
- porque houve necessidade de trabalhar com dispositivos de natureza diferente, houve a preocupação de definir uma representação abstracta para ter uma caracterização uniforme dos domínios. Esta descrição uniforme permite um tratamento igual dos domínios;
- um domínio é facilmente introduzido.

3.3.1 Arquitectura

A figura 3.8 mostra a arquitectura do *Gestor de Serviços*. É composta por 5 módulos: (i) *Service Manager Galaxy Server*; (ii) *Device Manager*; (iii) *Access Manager*; (iv) *Object Recognition Manager*; e (v) *Device Proxy*.

- ***Service Manager Galaxy Server*:** define a interface do *Gestor de Serviços*. Torna-se assim o único ponto de comunicação a partir do qual todos os pedidos e respostas relacionados com os dispositivos são feitos;
- ***Device Manager*:** o gestor de dispositivos ou *Device Manager* é o módulo principal do *Gestor de Serviços*. Este método tem os métodos principais a partir do qual o *Gestor de Serviços* funciona, incluíndo aqueles que lidam com os pedidos feitos;

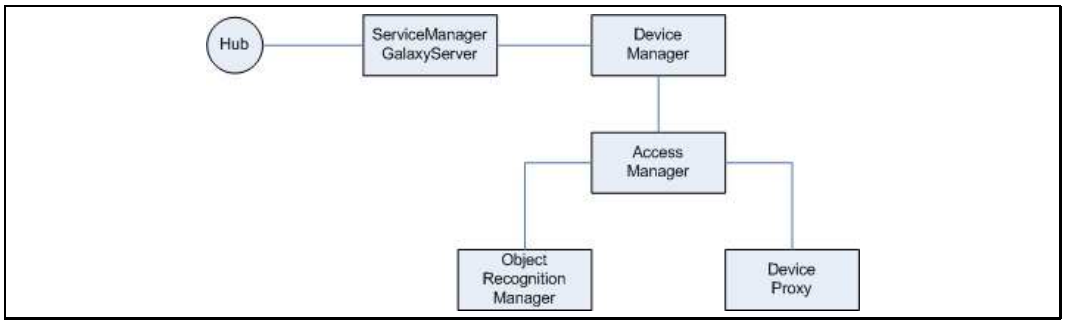

Figura 3.8: Arquitectura do *Gestor de Serviços.*

- ***Access Manager*:** O gestor de acessos ou *Access Manager* actua como um filtro sobre os pedidos feitos ao *Gestor de Serviços*. Este módulo restringe os serviços disponíveis ao utilizador corrente. Para esse fim, é gerido um ficheiro de restrições do dispositivo, e é verificado se o utilizador tem ou não permissões para aceder a dispositivos específicos;
- ***Object Recognition Manager*:** Este módulo é responsável pelo reconhecimento de todos os objectos. Faz a gestão da descrição dos dispositivos activos no sistema;
- ***Device Proxy*:** Este módulo mantém uma ligação com o dispositivo real. Todos os pedidos relativos a um dispositivo são tratados por este módulo.

3.3.2 Interface

Está fora do âmbito desta tese a descrição de todas as funcionalidades disponibilizadas por este módulo. Nesta secção, descrevem-se somente as funcionalidades implementadas na realização desta tese.

- **Reconhecimento dos objectos:** Esta operação recebe como argumento uma lista com os nomes de todos os objectos a serem reconhecidos. Devolve uma lista com a descrição dos objectos identificados. Objectos que pertencem a domínios sobre os quais o utilizador actual não tem acesso não são verificados. Se um objecto não for reconhecido, o nome do objecto é devolvido sem nenhuma descrição. A figura 3.10 contém a descrição XML do resultado do reconhecimento do objecto *PORTO*. Este objecto é, no domínio dos *Autocarros*, reconhecido como sendo uma *cidadeorigem* e uma *cidadedestino*. Um objecto pode fazer parte de mais de um domínio, ou no mesmo domínio, pertencer a mais do que um espaço de informação do enquadramento;

```
<xml>
  <item value='porto' comp='equal' />
</xml>
```

Figura 3.9: XML de entrada para o reconhecimento do objecto *Porto*.

```
<xml>
  <objects>
    <item value='porto'>
      <xml>
        <objecttype devicesid='2 ' domain='Autocarros'>
          <spectype main='cidadeorigem'>
            <modetype mode='word'>cidadeorigem</modetype>
          </spectype>
        </objecttype>
        <objecttype devicesid='2 ' domain='Autocarros'>
          <spectype main='cidadedestino'>
            <modetype mode='word'>cidadedestino</modetype>
          </spectype>
        </objecttype>
      </xml>
    </item>
  </objects>
</xml>
```

Figura 3.10: XML de saída do reconhecimento do objecto *Porto*.

- **Envio de enquadramentos:** Esta operação recebe como argumento uma lista com os nomes de todos os domínios para os quais se deseja obter os enquadramentos. Esta operação é necessária para que se possa processar a informação referente aos serviços de um dispositivo. Enquadramentos que pertencem a domínios sobre os quais o utilizador actual não tem acesso não são verificadas. O resultado desta operação para uma lista vazia é uma descrição de todos os enquadramentos activos no sistema;

- **Envio de serviços disponíveis:** um serviço representa uma possível acção a ser executada pelo utilizador do sistema. Esta operação é invocada usando como argumento uma lista com os nomes de todos os domínios para os quais se deseja obter os serviços disponíveis. Serviços que pertencem a domínios sobre os quais o utilizador actual não tem acesso não são verificadas. O resultado desta operação para uma lista vazia é uma descrição de todas os serviços activos no sistema. Num sistema de diálogos orientado a tarefas, estes serviços são usados para determinar o comportamento do sistema;

- **Envio de serviços indisponíveis:** Esta operação é invocada usando como argumento uma lista com os nomes de todos os domínios para os quais se deseja obter os serviços indisponíveis. Serviços que pertencem a domínios sobre os quais o utilizador actual não tem acesso não são verificadas. O resultado desta operação para uma lista vazia é uma descrição de todas os serviços indisponíveis no sistema;

- **Envio de restrições à execução dos serviços:** esta operação é invocada usando como argumento uma lista com os nomes de todos os domínios para os quais se deseja obter as restrições. Restrições que pertencem a domínios sobre os quais o utilizador actual não tem

acesso não são verificadas. O resultado desta operação para uma lista vazia é uma descrição de todas as restrições activas no sistema. Tal como na operação anterior, estas restrições são usadas para determinar o comportamento do sistema em domínios específicos;

- **Envio de regras compostas ou de geração de frases:** Esta operação é invocada usando como argumento o nome do domínio para o qual se deseja obter as regras de geração. Se o utilizador actual não tem acesso a um determinado domínio, as regras que lhe pertencem não são verificadas. Estas regras são usadas pelo *Gerador de Superfície* para gerar as frases do sistema num domínio em particular;

- **Envio dos valores possíveis de um espaço de informação** esta operação é invocada usando como argumentos: (i) uma lista com os nomes de todos os espaços de informação do enquadramento para os quais se deseja obter os valores possíveis; (ii) os identificadores dos dispositivos para os quais se deseja obter os valores possíveis. Apenas os dispositivos sobre os quais o utilizador actual tem acesso são verificados. Tal como na operação anterior, estas regras são usadas para determinar o comportamento do sistema em domínios específicos;

- **Execução de serviços:** esta operação é invocada usando como argumentos: (i) o identificador do dispositivo sobre o qual se quer ver executado o serviço; (ii) o identificador do serviço a ser executado; e (iii) os parâmetros de execução do serviço.

3.4 *Gestor de Tarefas*

Actualmente, o *Gestor de Tarefas* tem duas funções específicas: (i) criar Actos de Resolução de Problemas (ARP); e (ii) coordenar a execução dos serviços.

3.4.1 Criação dos Actos de Resolução de Problemas (ARP)

Os ARPs constituem um dos meios pela qual são criadas as obrigações de discurso (ver secção 3.6.2) no *Gestor de Interpretações* (ver secção 3.6).

Um Acto de Resolução de Problemas ou ARP contém:

- o nome do enquadramento que o originou;

- a descrição do estado actual do enquadramento;

- as regras de classificação que ainda podem ser utilizadas para tornar o estado actual no estado objectivo. Não tem descrito explicitamente o estado objectivo, mas contém a descrição de um conjunto de operações e regras, que possibilitam tornar o estado actual num que cumpre os objectivos;

- as regras de validação que estejam a tornar o enquadramento inválido;

- um valor probabilístico de reconhecimento (VPREC) que indica se o ARP é algo que faz sentido ou não, baseado no contexto actual do enquadramento;

- um valor probabilístico de resposta (VPRESP) que indica se a acção requisitada deverá ser possível de realizar ou não baseado no contexto actual do enquadramento.

Para criar um ARP, o *Gestor de Tarefas* necessita de 5 diferentes informações relacionadas com um domínio: (i) a interpretação corrente; (ii) o estado actual do diálogo; (iii) o enquadramento; (iv) os serviços; e (v) as possíveis restrições à execução dos serviços. Quando o *Gestor de Interpretaç ões* faz o pedido de cálculo de um ARP, fá-lo enviando a interpretação corrente e o estado actual do diálogo. A partir desta informação, o *Gestor de Tarefas* consulta o *Gestor de Serviços* (ver secção 3.3.2) para obter o enquadramento e os serviços disponíveis para o domínio em questão. As regras de classificação e de validação são obtidas a partir de uma conversão XML dos serviços e das restrições associadas aos mesmos, respectivamente. Esta conversão é impulsionada pela necessidade de facilitar a descrição de um dispositivo. A conversão permite obter as regras que vão ser processadas internamente no *Gestor de Tarefas*. Com o enquadramento, a interpretação corrente e o estado actual do diálogo, o *Gestor de Tarefas* obtém uma descrição do estado actual do enquadramento. As regras de classificação e de validação permitem obter os respectivos valores probabilísticos de reconhecimentos e de resposta.

As regras de classificação e de validação são formadas por composição de outras regras. Todas as regras têm uma probabilidade associada. Esta probabilidade representa o peso de cada regra na formação do ARP, e consequentemente, influencia as decisões do sistema. As regras cujo valor probabilístico seja máximo, ou seja, 1.0, não são consideradas no ARP, pois à excepção do caso em que garantem a execução de um serviço, não ajudam o sistema a decidir qual será a próxima pergunta a ser feita ao utilizador tendo em vista a execução de um serviço. Por outras palavras, a decisão do sistema é tomada com base na informação omissa e não na informação que dispõe de momento. O valor probabilístico associado às regras é dado pela verificação lógica das mesmas. A regra *rule* é usada para formar agrupamentos de regras. Todas as regras têm um valor probabilístico associado por omissão para quando não é possível derivar um valor para elas a partir dos elementos fornecidos. Estes valores por omissão podem ser configurados a partir de uma descrição XML e ditam o comportamento do sistema.

As regras de classificação e de validação são formadas pelas seguintes regras:

- **slot_value:** é usada para comparar o valor actual dado pela interpretação ou pelo estado do diálogo com o valor requerido pelo serviço (ver figura 3.11). Nesta regra, o atributo *required* indica se o valor relacionado com a regra é obrigatório. A obrigatoriedade impõe que o utilizador forneça sempre o valor adequado para a regra. *slotKey* faz referência ao espaço de informação do enquadramento do domínio indicado pelo atributo *domain*, *actualValue* contém o valor actual dado pela interpretação ou pelo estado do diálogo. Os atributos *spectype*, *maintype* e *modetype* complementam a informação sobre o valor dado por *actualValue*,

com valores dados pelo reconhecimento. O atributo *prob* contém o valor da probabilidade associado à regra. Na regra da figura 3.11, o valor do elemento *slot_value* é *consultar* e constitui o valor requerido pelo serviço. A regra *slot_value* é sempre criada a partir do parâmetro de um serviço que requer um valor específico. Assim, o parâmetro de um serviço como o indicado na figura 3.12 dá origem à regra da figura 3.11. A probabilidade desta regra é 0.0 já que o *actualValue* não corresponde a *consultar*;

```
<slot_value required="false" slotKey="1" prob="0.0"
            actualValue="comprar" domain="autocarros"
            spectype="action" maintype=""
            modetype="word">consultar</slot_value>
```

Figura 3.11: Exemplo de uma regra SlotValue.

```
<param slotkey="1" value="consultar"></param>
```

Figura 3.12: Exemplo do parâmetro de um serviço que requer um valor.

- **slot_filled:** é usada para requerer um valor para um espaço de informação específico (ver figura 3.13). Nesta regra, o atributo *required* indica se o valor relacionado com a regra é obrigatório, *actualValue* contém o valor actual dado pela interpretação ou pelo estado do diálogo. No exemplo, nenhum valor foi dado. Os atributos *spectype*, *maintype* e *modetype* complementam a informação sobre o valor dado por *actualValue*, com valores extraídos do reconhecimento de objectos de fala. O atributo *prob* contém o valor da probabilidade associado à regra. Nesta regra, o valor do elemento *slot_filled* é 3, e constitui a referência para o espaço de informação do enquadramento do domínio indicado em *domain*. A regra *slot_filled* é formada quando o parâmetro de um serviço não requer um valor. Assim, o parâmetro de um serviço como o indicado na figura 3.14 dá origem à regra da figura 3.13. A probabilidade desta regra (0.8) é a probabilidade por omissão da regra *slot_filled*[4];

```
<slot_filled required="false" prob="0.8"
             actualValue="" domain="autocarros"
             spectype="diasemana" maintype=""
             modetype="word">3</slot_filled>
```

Figura 3.13: Exemplo de uma regra SlotFilled.

- **And:** tem o valor verdadeiro quando todas as sub-regras forem verdadeiras (ver figura 3.15). O valor probabilístico da regra é dado pela multiplicação da probabilidade de omissão da própria regra (1.0) com as probabilidades das regras interiores;

[4] A regra *slot_value* tem por omissão probabilidade 0.3. Este valor é menor do que o valor para a regra *slot_filled* porque é assumido que os serviços que requerem quaisquer valores são preferíveis aos que requerem valores específicos.

```
<param slotkey="3"></param>
```

Figura 3.14: Exemplo do parâmetro de um serviço que não requer um valor.

```
<and prob="0.0">
  <slot_value required="false" slotKey="1" prob="0.0"
              actualValue="comprar" domain="autocarros"
              spectype="action" maintype=""
              modetype="word">consultar</slot_value>
  <slot_filled required="false" prob="0.8"
               actualValue="" domain="autocarros"
               spectype="diasemana" maintype=""
               modetype="word">3</slot_filled>
</and>
```

Figura 3.15: Exemplo de uma regra And.

- **Or:** tem o valor verdadeiro quando uma das sub-regras for verdadeira (ver figura 3.16). O valor probabilístico da regra é dado pela multiplicação da probabilidade de omissão da própria regra (1.0) com o valor da maior probabilidade que ocorre nas regras interiores;

```
<or prob="0.8">
  <slot_value required="false" slotKey="1" prob="0.0"
              actualValue="comprar" domain="autocarros"
              spectype="action" maintype=""
              modetype="word">consultar</slot_value>
  <slot_filled required="false" prob="0.8"
               actualValue="" domain="autocarros"
               spectype="diasemana" maintype=""
               modetype="word">3</slot_filled>
</or>
```

Figura 3.16: Exemplo de uma regra Or.

- **Equal:** tem o valor verdadeiro quando os valores dados pela interpretação ou pelo estado do diálogo para as sub-regras é igual (ver figura 3.17). O valor probabilístico da regra é dado pela multiplicação da probabilidade de omissão da própria regra (definido como 0.8 neste exemplo) com as probabilidades das regras interiores. Na figura 3.17, a cidade de origem deve ser igual à cidade de destino para a regra ser considerada verdadeira;

- **Ascending:** tem o valor verdadeiro quando a ordem dos valores dados pela interpretação ou pelo estado do diálogo para as regras interiores é ascendente (ver figura 3.18). O valor probabilístico da regra é dado pela multiplicação da probabilidade de omissão da própria regra (1.0) com as probabilidades das regras interiores. Na figura 3.18, é exigido que a hora

```
<equal prob="0.512">
  <slot_filled required="false" prob="0.8"
               actualValue="" domain="autocarros"
               spectype="cidadeorigem" maintype=""
               modetype="word">4</slot_filled>
  <slot_filled required="false" prob="0.8"
               actualValue="" domain="autocarros"
               spectype="cidadedestino" maintype=""
               modetype="word">5</slot_filled>
</equal>
```

Figura 3.17: Exemplo de uma regra Equal.

de chegada seja superior à hora de partida;

```
<ascending prob="0.64">
  <slot_filled required="false" prob="0.8"
               actualValue="" domain="autocarros"
               spectype="horapartida" maintype=""
               modetype="word">6</slot_filled>
  <slot_filled required="false" prob="0.8"
               actualValue="" domain="autocarros"
               spectype="horachegada" maintype=""
               modetype="word">7</slot_filled>
</ascending>
```

Figura 3.18: Exemplo de uma regra Ascending.

- **Not:** se a regra imediatamente inferior tiver valor verdadeiro, a regra *Not* tem o valor falso e assume como valor probabilístico a probabilidade da regra imediatamente inferior (ver figura 3.19). Se a regra imediatamente inferior for falsa, a regra é verdadeira e assume o valor probabilístico de omissão (1.0). Na figura 3.19, é exigido que a cidade de origem e a cidade de destino não sejam iguais. A regra *Not* com a regra *Equal* e a regra *Not* com a regra *Ascending* são usadas no domínio dos Autocarros como regras de validação. Elas expressam que a cidade de origem e a cidade de destino não podem ser iguais e que a hora de partida não pode ser superior à hora de chegada, respectivamente;

- **If:** o valor probabilístico do *If* é igual à multiplicação do valor probabilístico da própria regra com o valor probabilístico do elemento *then* se *test* for considerado verdadeiro (ver figura 3.20). O valor probabilístico do *If* é igual à multiplicação do valor probabilístico da própria regra com o valor probabilístico do elemento *else* se *test* for considerado falso. *test* assume a condição lógica da regra imediatamente inferior. Na figura 3.20, a regra *slot_value* é falsa, logo o *if* assume o valor probabilístico do *else* (1.0);

- **Implic:** a implicação é verdadeira quando o argumento anterior (na figura 3.21,

```
<not prob="0.512">
  <equal prob="0.512">
    <slot_filled required="false" prob="0.8"
                 actualValue="" domain="autocarros"
                 spectype="cidadeorigem" maintype=""
                 modetype="word">4</slot_filled>
    <slot_filled required="false" prob="0.8"
                 actualValue="" domain="autocarros"
                 spectype="cidadedestino" maintype=""
                 modetype="word">5</slot_filled>
  </equal>
</not>
```

Figura 3.19: Exemplo de uma regra Not.

```
<if prob="1.0">
  <test>
    <slot_value required="false" slotKey="1" prob="0.0"
                actualValue="comprar" domain="autocarros"
                spectype="action" maintype=""
                modetype="word">consultar</slot_value>
  </test>
  <then prob="0.8">
    <slot_filled required="false" prob="0.8"
                 actualValue="" domain="autocarros"
                 spectype="cidadeorigem" maintype=""
                 modetype="word">4</slot_filled>
  </then>
  <else prob="1.0"></else>
</if>
```

Figura 3.20: Exemplo de uma regra If.

(*arg_anterior*) é falso, ou o argumento posterior (na figura 3.21, (*arg_anterior*) é verdadeiro. Na figura 3.21, porque *slot_value* é falso, o argumento anterior é verdadeiro e assume a probabilidade da regra interior. O valor probabilístico da implicação é igual ao seu valor de omissão porque o argumento anterior é verdadeiro. Se o argumento anterior fosse verdadeiro, seria ainda avaliado o argumento posterior. Quando a implicação é falsa, o seu valor probabilístico é igual à multiplicação do seu valor de omissão com as probabilidades do argumento anterior e do argumento posterior.

É apresentado de seguida um exemplo do cálculo de um ARP para a frase "Quero comprar um bilhete para o Porto". As figuras referenciadas correspondem a componentes do ARP criado. A figura 3.22 mostra a descrição do estado actual do enquadramento (ver secção 3.7.1) para a frase dita pelo utilizador. Este estado foi totalmente criado a partir da interpretação da frase.

```
<implic prob="1.0">
  <arg_anterior prob="0.0">
    <slot_value required="false" slotKey="1" prob="0.0"
                actualValue="comprar" domain="autocarros"
                spectype="action" maintype=""
                modetype="word">consultar</slot_value>
  </arg_anterior>
  <arg_posterior prob="0.8">
    <slot_filled required="false" prob="0.8"
                 actualValue="" domain="autocarros"
                 spectype="cidadeorigem" maintype=""
                 modetype="word">4</slot_filled>
  </arg_posterior>
</implic>
```

Figura 3.21: Exemplo de uma regra Implic.

```
<actual_state>
  <slot required='false' key='5'
        description='cidadedestino' comp='equal'>porto</slot>
  <slot required='false' key='2'
        description='target' comp='equal'>bilhete</slot>
  <slot required='false' key='1'
        description='action' comp='equal'>comprar</slot>
</actual_state>
```

Figura 3.22: Estado actual do enquadramento criado para a frase "Quero comprar um bilhete para o Porto".

Para o domínio dos autocarros, que corresponde à interpretação da frase dita pelo utilizador, existem dois serviços disponíveis, os únicos descritos no único dispositivo associado ao domínio dos autocarros: (i) o serviço de consulta de horas de partida; e (ii) o serviço de compra de bilhetes. A transformação em regra de classificação do serviço de consulta de horas de partida para a frase "Quero comprar um bilhete para o Porto" encontra-se descrita na figura 3.23.

Para os cinco parâmetros descritos no serviço de consulta de horas de partida (ver figura 3.23, $< params >$), foram criadas na regra de classificação (ver figura 3.23, $< rule >$) quatro regras. As primeiras duas regras *slot_value* foram criadas para os primeiros dois parâmetros do serviço. Repare-se que na descrição destes dois parâmetros já foram atribuídos dois valores, um para cada um dois primeiros espaços de informação do enquadramento de autocarros (ver figura 3.3), *action* e *target*. As outras duas regras *slot_filled* foram criadas para os seguintes dois espaços de informação do enquadramento, o *DiaSemana* e a *CidadeOrigem*. A regra *slot_filled* respeitante ao quinto espaço de informação do enquadramento de autocarros não aparece na regra de classificação, já que a mesma exigia uma qualquer cidade de destino, à qual a interpretação para a frase dita pelo utilizador correspondeu. Esta regra é então retirada do ARP porque a decisão

```
<service name="consultar hora de partida">
  <label>Consultar horas de partida</label>
    <params>
      <param slotkey="1" value="consultar"></param>
      <param slotkey="2" value="horapartida"></param>
      <param slotkey="3"></param>
      <param slotkey="4"></param>
      <param slotkey="5"></param>
    </params>
    <execute>
      <name>consultarHorapartida</name>
      <success></success>
      <failure></failure>
    </execute>
</service>

<rule deviceID='2' prob='0.0'>
  <and deviceID='' prob='0.0'>
    <slot_value required='false' slotKey='1' prob='0.0'
                actualValue='comprar' domain='autocarros'
                spectype='action' maintype=''
                modetype='word'>consultar</slot_value>
    <slot_value required='false' slotKey='2' prob='0.0'
                actualValue='bilhete' domain='Autocarros'
                spectype='target' maintype=''
                modetype='word'>horapartida</slot_value>
    <slot_filled required='false' grauCerteza='0.0'
                grauCertezaActual='0.0' comp='equal' prob='0.8'
                actualValue='' domain='autocarros'
                spectype='diasemana' maintype=''
                modetype='word'>3</slot_filled>
    <slot_filled required='false' grauCerteza='0.0'
                grauCertezaActual='0.0' comp='equal' prob='0.8'
                actualValue='' domain='autocarros'
                spectype='cidadeorigem' maintype=''
                modetype='word'>4</slot_filled>
  </and>
</rule>
```

Figura 3.23: Transformação do serviço de consulta de horas de partida em regra de classificação.

do sistema vai ser tomada apenas com base na informação em falta para a execução dos serviços. Analisando os valores probabilísticos destas regras, as regras *slot_value* assumem probabilidades nulas em função da falta de mapeamento dos valores que a interpretação forneceu com aqueles que o serviço estava à espera. Como a interpretação não forneceu qualquer valor para o *DiaSemana* e a *CidadeOrigem*, as regras *slot_filled* vão manter os valores probabilísticos de omissão. A regra *and* faz a conjunção dos cinco parâmetros fornecidos no serviço e tem probabilidade nula, já que

a multiplicação das probabilidades de cada uma das regras interiores dá 0. A regra rule espelha o elemento *service* do XML, sendo assim o "nó" de nível mais elevado na regra de classificação. A probabilidade desta regra é igual à probabilidade da regra imediata que engloba. Desta forma, o serviço da figura 3.23 não tem qualquer hipótese de ser executado no contexto em que se inclui a interpretação da frase "Quero comprar um bilhete para o Porto". A transformação em regra de classificação do serviço de compra de bilhetes de autocarro para essa frase encontra-se descrita na figura 3.24.

O serviço de compra de bilhetes de autocarro é diferente do serviço de consulta de horas de partida, na medida em que exige que o utilizador forneça uma hora de partida ou uma hora de chegada. A disjunção é feita com o uso do *or* na descrição do serviço a englobar os respectivos parâmetros. Ao contrário da regra anterior, a interpretação da frase dita pelo utilizador corresponde ao esperado para os dois primeiros espaços de informação do serviço de compra de bilhetes. O utilizador pretende "comprar bilhete". O mapeamento exacto da informação exigida pelo serviço com a informação dada pelo utilizador faz com que as mesmas não apareçam no ARP. Também não foi fornecida qualquer hora de partida ou hora de chegada, daí a permanência das regras *slot_filled* englobadas na regra lógica *or*. O valor probabilístico da regra *or* é dado pelo máximo do valor das regras que engloba, daí o valor de 0.8 para a sua probabilidade. O valor probabilístico da regra *and* tem o valor da multiplicação das probabilidades das regras interiores, ou seja, 0.8 * 0.8 * 0.8, o que dá a probabilidade de 0.512 para o *and* e consequentemente, para a *rule*. Como se está especialmente interessado em tratar aqueles serviços que ofereçam uma maior probabilidade de execução, esta última regra é naturalmente aquela que mais se adequa a esse fim.

A descrição do dipositivo de venda de bilhetes de autocarro conta actualmente com duas restrições ao preenchimento dos espaços de informação do enquadramento do domínio a que o dispositivo pertence. As restrições são as apresentadas na figura 3.25. Os espaços de informação 4 e 5 são respectivamente, referências para os espaços de informação *CidadeOrigem* e *CidadeDestino* no enquadramento. Assim, esta restrição diz que a cidade de origem não pode ser igual à cidade de destino (*not equal*). Os espaços de informação 6 e 7 referem-se aos espaços de informação *HoraPartida* e *HoraChegada* respectivamente, e afirma que a hora de partida não pode ser maior que a hora de chegada (*ascending*). A descrição aqui apresentada é, tal como para os serviços descritos anteriormente, convertida em regras de validação a serem utilizadas para processamento pelo *Gestor de Tarefas*. Como no contexto criado pela única interpretação de "Quero comprar um bilhete para o Porto" não se evidencia qualquer uma destas restrições, as duas regras de validação não surgem no ARP, e por isso, não são contabilizadas.

Para completar a descrição das componentes no ARP, só falta dar significado aos valores probabilísticos de reconhecimento (VPREC) e de resposta (VPRESP).

O VPREC é um indicador quantificável probabilístico da proximidade à execução de um serviço de qualquer dispositivo activo no sistema. O seu valor será tanto mais alto quanto maior for a proximidade. Quando existe um serviço que cumpra todas as condições de execução, o valor do VPREC é igual a 1. Quando nenhum serviço cumpre a mínima condição, o seu valor é igual a 0.

```
<service name="comprar bilhete">
  <label>Consultar horários dos autocarros</label>
  <params>
    <param slotkey="1" value="comprar"></param>
    <param slotkey="2" value="bilhete"></param>
    <param slotkey="3"></param>
    <param slotkey="4"></param>
    <param slotkey="5"></param>
    <or>
      <param slotkey="6"></param>
      <param slotkey="7"></param>
    </or>
  </params>
    <execute>
      <name>comprarBilhete</name>
      <success></success>
      <failure></failure>
    </execute>
</service>

<rule deviceID='2' prob='0.512'>
  <and deviceID='' prob='0.512'>
    <slot_filled required='false' grauCerteza='0.0'
                 grauCertezaActual='0.0' comp='equal' prob='0.8'
                 actualValue='' domain='autocarros'
                 spectype='diasemana' maintype=''
                 modetype='word'>3</slot_filled>
    <slot_filled required='false' grauCerteza='0.0'
                 grauCertezaActual='0.0' comp='equal' prob='0.8'
                 actualValue='' domain='autocarros'
                 spectype='cidadeorigem' maintype=''
                 modetype='word'>4</slot_filled>
    <or deviceID='' prob='0.8'>
      <slot_filled required='false' grauCerteza='0.0'
                   grauCertezaActual='0.0' comp='equal' prob='0.8'
                   actualValue='' domain='autocarros'
                   spectype='horapartida' maintype=''
                   modetype='word'>6</slot_filled>
      <slot_filled required='false' grauCerteza='0.0'
                   grauCertezaActual='0.0' comp='equal' prob='0.8'
                   actualValue='' domain='autocarros'
                   spectype='horachegada' maintype=''
                   modetype='word'>7</slot_filled>
    </or>
  </and>
</rule>
```

Figura 3.24: Transformação do serviço de compra de bilhetes em regra de classificação.

```
<validation>
  <not>
    <equal>
      <params>
        <param slotkey="4"></param>
        <param slotkey="5"></param>
      </params>
    </equal>
  </not>
</validation>

<validation>
  <ascending>
    <params>
      <param slotkey="6"></param>
      <param slotkey="7"></param>
    </params>
  </ascending>
</validation>
```

Figura 3.25: Descrição das restrições para o dispositivo da venda de bilhetes de autocarro.

Para as regras de classificação apresentadas nas figuras 3.23 e 3.24, o VPREC é igual ao valor da regra com maior probabilidade, ou seja, 0.512.

O VPRESP é um indicador quantificável probabilístico da validação da informação providenciada pelas interpretações das afirmações do utilizador até ao momento. O VPRESP toma o valor 1 quando a informação é coerente, e nenhuma restrição se verifica, o que acontece nos exemplos supramencionados. Toma o valor 0 quando todas as restrições se verificam na íntegra. Como se vai poder verificar, ambos os valores vão ser importantes na construção das obrigações de discurso (ver secção 3.6.2).

3.4.2 Coordenação da execução dos serviços

Quando todas as pré-condições à execução de um serviço são verificadas, o *Gestor de Interpretações* (ver secção 3.6) elabora um pedido ao *Gestor de Tarefas* para a obtenção dos resultados da execução. Este pedido é feito com dois argumentos: (i) o domínio; e (ii) o estado do diálogo. Com estes dois elementos, o *Gestor de Tarefas* consegue obter informação sobre o serviço a executar. Para o pedido de execução de um serviço (ver 3.3.2), o *Gestor de Tarefas* reúne a seguinte informação: (i) o identificador do dispositivo; (ii) o identificador do serviço; e (iii) os parâmetros de execução do serviço. Ao passo que os dois primeiros elementos são obtidos facilmente através da identificação do serviço a executar, o terceiro é mais difícil de obter. Actualmente, a obtenção dos parâmetros é efectuada subtraíndo informação implícita ao serviço, do conjunto de todos os parâmetros exigidos à execução do mesmo. Por informação implícita ao

serviço entende-se aquela informação que a identificação do dispositivo e do serviço já fornece.

Para o serviço da compra de bilhetes apresentado na figura 3.24, os primeiros dois espaços de informação, respectivamente, *Action* e *Target*, não são considerados parâmetros de execução do serviço, já que identificam o serviço a executar. Por outras palavras, estes dois parâmetros constituem informação desnecessária quando o serviço para execução já está seleccionado. O conjunto dos parâmetros exigidos à execução do serviço de compra de bilhetes é constituído pelos cinco restantes parâmetros do serviço (ver figura 3.3). Para os domínios empregues na execução do trabalho nesta tese (Autocarros e Domótica), os parâmetros de todos os serviços são dados pela retirada dos dois primeiros espaços de informação do enquadramento do respectivo domínio, mas tal pode não ser suficiente para outros domínios. A determinação dos parâmetros deve ser sempre efectuada com algum cuidado, dado o enquadramento do domínio a que o dispositivo pertençe e o serviço pretendido.

3.5 Reconhecedor da Linguagem

O *Reconhecedor da Linguagem* é o módulo responsável pela transformação do texto em actos de fala (ver secção 2.5) para serem utilizados como entrada no *Gestor de Interpretações*. Esta transformação deve ser feita com base em informação linguística. Definida uma estrutura para os actos de fala a empregar no gestor, decidiu-se numa primeira versão que os mesmos seriam obtidos por mapeamento quase directo entre a frase e o acto de fala préviamente construído. Esta versão para a construção dos actos de fala foi então construída de forma muito primitiva, com mapeamentos quase directos entre a frase proferida pelo utilizador e o acto de fala pretendido para essa mesma frase. Um acto de fala contém um identificador e a informação da frase a que está associado. Contém também um conjunto de propriedades:

- **Type:** contém a performativa encontrada para a frase dita pelo utilizador, ou seja, explicita o tipo de acto de fala associado à frase;
- **Who:** diz quem é o autor da frase;
- **Action:** diz qual é a acção associada à frase;
- **What:** contém o complemento directo da frase. Pode conter mais do que um objecto;
- **FromWhere:** contém um ou mais objectos da frase classificados como origens;
- **ToWhere:** contém um ou mais objectos da frase classificados como destinos;
- **BeginTime:** contém um ou mais objectos da frase classificados como tempos de início;
- **EndTime:** contém um ou mais objectos da frase classificados como tempos finais.

Nas propriedades, os objectos são indicados através do elemento de nome *item*. Existem dois tipos de atributos associados aos objectos: (i) o critério de comparação; e (ii) o valor. O valor de comparação pode assumir os seguintes valores: (i) *greater*; (ii) *lesser* e (iii) *equal*. Mais características poderiam vir associadas a um acto de fala, dependendo da quantidade de informação que o reconhecedor da linguagem providencia.

A figura 3.26 mostra o acto de fala associado à frase "Quero comprar um bilhete para o Porto". Repare-se que as palavras "Quero" e "para" são utilizadas para indicar, respectivamente, o tipo do acto de fala ($< type > Request < /type >$) e a propriedade ($< towhere >< item > porto < /item >< /towhere >$).

```
<sa id="42" value="quero comprar um bilhete para o porto">
  <type>Request</type>
  <who>user</who>
  <action>comprar</action>
  <what>
    <item>bilhete</item>
  </what>
  <towhere>
    <item>porto</item>
  </towhere>
</sa>
```

Figura 3.26: Saída do Language Parser para a frase "Quero comprar um bilhete para o Porto".

3.6 Gestor de Interpretações

O *Gestor de Interpretações* é o módulo principal na componente de interpretação do gestor de diálogos. Este módulo recebe um conjunto de actos de fala como entrada e produz dois tipos de informação fundamentais no sistema: (i) interpretações; e (ii) obrigações do discurso (*discourse obligations*). As interpretações são as possíveis interpretações do que foi dito pelo utilizador, enquanto que as obrigações do discurso são representativas das intenções do próprio sistema na resolução das tarefas impostas pelas interpretações previamente determinadas. Este módulo troca informação com o *Gestor de Serviços*, o *Gestor de Tarefas*, o *Gestor do Contexto* e o *Gestor de Comportamentos*.

3.6.1 Interpretações

O processo de interpretação da frase proferida pelo utilizador está subdividida em duas fases distintas. A primeira ocorre no *Reconhecedor da Linguagem* (ver secção 3.5), e consiste na associação de um conjunto de actos de fala à frase. A segunda fase ocorre no *Gestor de Interpretações*, e faz a fusão da informação resultante da primeira fase, com informação do domínio e do contexto.

Uma interpretação é então definida como uma instância das possíveis combinações derivadas: (i) do acto de fala; (ii) do reconhecimento do objectos contidos no acto de fala; e (iii) do contexto actual.

Para que seja possível a realização do reconhecimento dos objectos contidos no acto de fala, é estabelecida uma comunicação com o *Gestor de Serviços* (ver secção 3.3) e utilizada a funcionalidade descrita em 3.3.2. No pedido, é enviado como argumento uma lista com todos os objectos contidos nos actos de fala, enquanto que na resposta é obtida uma descrição do "significado" desses mesmos objectos para os dispositivos activos no sistema. O número de interpretações criadas pelo *Gestor de Interpretações* nesta fase é dado pelo número de combinações dos significados possíveis para todos os objectos contidos no acto de fala. Desta forma, para a frase "Quero comprar um bilhete para o Porto" vão ser criadas duas interpretações (ver figura 3.27). Enquanto que *comprar* e *bilhete* tem apenas um significado no domínio dos autocarros, *Porto* tem dois possíveis significados: (i) *cidadeorigem* e (ii) *cidadedestino* [5].

Naturalmente, nunca existirá uma concordância entre todos os possíveis significados dos objectos em qualquer domínio, e as propriedades da interpretação onde os mesmos sejam inseridos. Um objecto, neste caso, *Porto*, que é considerado ser uma *cidadeorigem* não pode ficar inserida na propriedade de uma interpretação *Towhere* (interpretação da esquerda da figura 3.27). Repare-se que a semântica dada ao objecto *Porto* entra em conflito com a semântica da propriedade da interpretação. Dito informalmente, nunca se quer ir para uma "origem". Torna-se assim necessário eliminar interpretações inválidas. Para esse efeito, foram criadas regras que ditam as combinações que não são válidas nestas interpretações (ver figura 3.28). Após a aplicação destas regras, obtém-se apenas uma única interpretação para a frase "Quero comprar um bilhete para o Porto" (interpretação da direita da figura 3.27).

O processo de criação de interpretações não termina com o reconhecimento dos objectos. Após aplicação da informação dominiana sobre os actos de fala, é ainda necessário usar informação contextual para desambiguar as possíveis interpretações. A desambiguação das interpretações criadas são feitas no *Gestor do Contexto* (ver secção 3.7), como resultado do mapeamento entre as interpretações actuais e as perguntas que poderão estar pendentes no *Gestor do Contexto*.

Para exemplificar (ainda no domínio dos Autocarros), supõe-se que o utilizador afirma "Porto" como resposta a uma pergunta do sistema "Para onde deseja viajar?". Após a criação das duas interpretações, descritas na figura 3.29, é necessário que se faça a correspondência entre as interpretações criadas e aquilo que o sistema está à espera que o utilizador diga. Assim, como resultado da desambiguação de interpretações feita no *Gestor do Contexto*, está-se interessado somente na interpretação de "Porto" como *cidadedestino* (ver figura 3.30), a qual corresponde à pergunta feita previamente pelo sistema.

Após aplicar informação do domínio e contextual sobre os actos de fala, que resultaram na determinação de uma ou mais interpretações para a frase dita pelo utilizador, é necessário ac-

[5] Assume-se neste exemplo que o domínio dos Autocarros é o único domínio activo no sistema que faz referência aos objectos contidos no acto de fala, e que o utilizador corrente tem acesso ao mesmo.

```
<interpretations>
  <interpretation>
    <property type='type'>
                  Request</property>
    <property type='who'>
                  User</property>
    <property type='What'>
      <item>
        <name>comprar</name>
        <domain>autocarros</domain>
        <device></device>
        <maintype>action</maintype>
        <modetype>word</modetype>
        <spectype>action</spectype>
        <value>0.0</value>
        <comp>equal</comp>
      </item>
      <item>
        <name>bilhete</name>
        <domain>autocarros</domain>
        <device></device>
        <maintype>target</maintype>
        <modetype>word</modetype>
        <spectype>target</spectype>
        <value>0.0</value>
        <comp>equal</comp>
      </item>
    </property>
    <property type='Towhere'>
      <item>
        <name>porto</name>
        <domain>autocarros</domain>
        <device></device>
        <maintype>cidadeorigem</maintype>
        <modetype>word</modetype>
        <spectype>cidadeorigem</spectype>
        <value>0.0</value>
        <comp>equal</comp>
      </item>
    </property>
  </interpretation>
</interpretations>
```

```
<interpretations>
  <interpretation>
    <property type='type'>
                  Request</property>
    <property type='who'>
                  User</property>
    <property type='What'>
      <item>
        <name>comprar</name>
        <domain>autocarros</domain>
        <device></device>
        <maintype>action</maintype>
        <modetype>word</modetype>
        <spectype>action</spectype>
        <value>0.0</value>
        <comp>equal</comp>
      </item>
      <item>
        <name>bilhete</name>
        <domain>autocarros</domain>
        <device></device>
        <maintype>target</maintype>
        <modetype>word</modetype>
        <spectype>target</spectype>
        <value>0.0</value>
        <comp>equal</comp>
      </item>
    </property>
    <property type='Towhere'>
      <item>
        <name>porto</name>
        <domain>autocarros</domain>
        <device></device>
        <maintype>cidadedestino</maintype>
        <modetype>word</modetype>
        <spectype>cidadedestino</spectype>
        <value>0.0</value>
        <comp>equal</comp>
      </item>
    </property>
  </interpretation>
</interpretations>
```

Figura 3.27: Interpretações criadas a partir do reconhecimento dos objectos na frase *"Quero comprar um bilhete para o Porto". A diferença entre as duas reside no spectype da propriedade Towhere.*

```
<contradictions>
  <fromwhere>
    <destino></destino>
  </fromwhere>
  <towhere>
    <origem></origem>
  </towhere>
  <begintime>
    <chegada></chegada>
    <duracao></duracao>
    <fim></fim>
  </begintime>
  <endtime>
    <partida></partida>
    <inicio></inicio>
    <duracao></duracao>
  </endtime>
</contradictions>
```

Figura 3.28: Exemplo das regras usadas para eliminar interpretações que não sejam válidas.

```
<interpretation>
  <property type = 'type'>affirm</property>
  <property type = 'who'>user</property>
  <property type='What'>
    <item>
      <name>porto</name>
      <domain>autocarros</domain>
      <device></device>
      <maintype>cidadeorigem</maintype>
      <modetype>word</modetype>
      <spectype>cidadeorigem</spectype>
      <value>0.0</value>
      <comp>equal</comp>
    </item>
  </property>
</interpretation>
```

Figura 3.29: Interpretação de "Porto" como *cidadeorigem*

tualizar duas informações essenciais ao funcionamento do sistema e em particular, à componente de interpretação: (i) o domínio; e (ii) o estado do diálogo.

No gestor, mais do que um diálogo pode acontecer ao mesmo tempo e em domínios diferentes. Para que isso seja possível, é essencial que após a determinação das interpretações correntes, o domínio seja novamente estabelecido. Para o efeito, três diferentes verificações são aplicadas,

```
<interpretation>
  <property type = 'type'>affirm</property>
  <property type = 'who'>user</property>
  <property type='What'>
    <item>
      <name>porto</name>
      <domain>autocarros</domain>
      <device></device>
      <maintype>cidadedestino</maintype>
      <modetype>word</modetype>
      <spectype>cidadedestino</spectype>
      <value>0.0</value>
      <comp>equal</comp>
    </item>
  </property>
</interpretation>
```

Figura 3.30: Interpretação de "Porto" como *cidadedestino*

pela ordem que se segue, e com resultados que podem auxiliar à desambiguação de interpretações:

1. Verifica-se se as interpretações fazem referência a apenas um único domínio. Caso esta condição seja verdadeira, mantêm-se as interpretações já estabelecidas até ao momento, o domínio actual é dado pelo domínio das interpretações e as duas verificações seguintes não são realizadas;

2. Verifica-se se existe alguma interpretação que faça referência a um único domínio. É dada preferência às interpretações cujos objectos pertençam todos ao mesmo domínio, o que leva à eliminação de todas as interpretações com objectos pertencentes a domínios diferentes. Se no final deste processo se obtiver mais do que uma interpretação com referência a um único domínio, é realizada novamente a primeira verificação, agora com a certeza de se ter menos interpretações para desambiguar. Caso se obtenha como resultado uma única interpretação, o domínio actual passa a ser o domínio dessa mesma interpretação. Em qualquer dos casos, a terçeira verificação não é realizada;

3. Verifica-se a existência de palavras chave de um domínio nas interpretações obtidas até ao momento. Esta acaba por ser uma verificação de recurso que raramente é utilizada. Aqui são também eliminadas as interpretações que não contêm qualquer palavra-chave. Se no final deste processo, se obtiver mais do que uma interpretação, não é possível determinar o domínio actual. Caso contrário, o domínio actual passa a ser o domínio dessa mesma interpretação.

Existem em geral, duas situações em que as verificações anteriores não são suficientes para determinar o novo domínio:

- As interpretações de uma frase não fazem referência a um domínio em particular. Neste caso, o domínio corrente será dado pelo domínio determinado em interpretações prévias. A resposta do sistema a este tipo de interpretações não necessita da informação de qualquer domínio. Por exemplo, quando o utilizador diz "Olá.", e o sistema responde com "Olá.";
- As interpretações de uma frase fazem referência a mais do que um domínio, caso em que o sistema deverá realizar uma desambiguação de domínio (ver secção 3.6.2), que consiste em perguntar directamente ao utilizador o domínio que o mesmo deseja tratar.

Após a determinação das interpretações, o estado do diálogo é actualizado. O estado do diálogo consiste na identificação daquilo que já foi dito pelo utilizador em cada um dos domínios activos no sistema e é mantido no *Discourse Context* (ver secção 3.7). Após esta actualização, torna-se necessário determinar a intenção do sistema no sentido de resolver os novos problemas que se lhe deparam. Este tópico será tratado na próxima secção.

3.6.2 Obrigações do Discurso

Uma obrigação do discurso é uma intenção do próprio sistema. No contexto de um diálogo, a criação e a gestão das obrigações do discurso [14], como aquelas criadas por questões, promessas, e até silêncios [16], é um desafio para um sistema de diálogos. A atitude de obrigação no sistema constitui um auxílio indispensável a atitudes de crença, objectivos e intenções [15], e mais importante, pode ser usado para estabelecer a ligação de uma pergunta do sistema com uma resposta dada pelo utilizador, tal como se tem usado no decorrer desta dissertação. As obrigações de discurso são criadas no *Gestor de Interpretações* analisando: (i) as interpretações correntes; (ii) o contexto actual; e (iii) os actos de resolução de problemas (ARPs) associados às interpretações criadas. Existem no sistema obrigações de discurso que são independentes do domínio e outras que são dependentes. A distinção entre estes dois tipos é feita com base na informação que é usada para a criação dos mesmos. Uma interpretação que é criada usando apenas a interpretação corrente e o contexto actual é considerada independente do domínio, ao passo que uma interpretação criada a partir dos ARPs é dependente do domínio. Para que este último tipo de obrigações seja criado, é necessário que o *Gestor de Interpretações* elabore um pedido ao *Gestor de Tarefas* (ver secção 3.4 para que este devolva os ARPs associado às interpretações correntes (ver secção 3.4.1).

São descritas de seguida as obrigações de discurso independentes do domínio:

- **Cumprimento:** criada quando o utilizador diz algo que é interpretado como um cumprimento. A resposta do sistema a um cumprimento é um cumprimento (ver figura 3.31);

```
UTILIZADOR: Olá.
SISTEMA: Olá.
```

Figura 3.31: Exemplo da obrigação de discurso "Cumprimento".

- **Apresentação:** pode ser colocada no início de um diálogo para apresentar o sistema ao novo utilizador (ver figura 3.32);

```
UTILIZADOR: Olá.
SISTEMA: Olá.
SISTEMA: Sou o Ambrósio. Que posso fazer por si?
```

Figura 3.32: Exemplo da obrigação de discurso "Apresentação".

- **Ausência de confirmação:** criada para informar o utilizador de que o sistema não espera nenhuma resposta do tipo confirmação (ver figura 3.33). Tal como para o "Sim", o "Não" tem exactamente o mesmo efeito.

```
UTILIZADOR: Sim
SISTEMA:  Não existe nada por confirmar!
```

Figura 3.33: Exemplo da obrigação de discurso "Ausência de confirmação".

- **Desambiguação de domínio:** criada quando existe a necessidade de desambiguar com o utilizador o domínio corrente (ver figura 3.34).

```
SISTEMA: Em que domínio pretende estabelecer o diálogo ?
```

Figura 3.34: Exemplo da obrigação de discurso "Desambiguação de domínio".

Obrigações dependentes do domínio:

- **Desambiguação de espaço de informação:** criada quando um objecto pode "encaixar" no mínimo em dois espaços de informação de um enquadramento. Para que a sequência apresentada na figura 3.35 aconteça, é necessário que: (i) *Lisboa* seja reconhecida como uma *CidadeOrigem*, e uma *CidadeDestino* no enquadramento do domínio dos Autocarros; (ii) nenhuma informação no contexto actual forneça dados que auxiliem a desambiguação a efectuar;

```
UTILIZADOR: Lisboa.
SISTEMA: Lisboa é cidade de origem ou é cidade de destino ?
```

Figura 3.35: Exemplo da obrigação de discurso "Desambiguação de espaço de informação".

- **Pedido de valor para um espaço de informação:** criada quando o sistema pretende questionar o utilizador acerca de determinado espaço de informação (ver figura 3.36);

```
SISTEMA: A que horas quer partir com destino a Porto?
```

Figura 3.36: Exemplo da obrigação de discurso "Pedido de valor para um espaço de informação".

Para que a obrigação de discurso "Pedido de valor para um espaço de informação" seja criada, as seguintes condições devem ser verificadas: (1) o valor probabilístico de resposta ou

VPRESP (ver secção 3.4.1) deve ser igual a 1; (2) o valor probabilístico de reconhecimento ou VPREC (ver secção 3.4.1) deve ser diferente de 0; e (3) o VPREC deve ser menor do que 1. O primeiro ponto é exigido porque as incoerências do discurso devem ser resolvidas assim que são detectadas. Se o segundo ponto não se verificar, o diálogo precisará de uma clarificação, já que nenhum serviço está disponível para o que o utilizador pretende. Se o VPREC for 1, tal como foi dito na secção 3.4.1, o sistema não necessita de mais informação para executar um determinado serviço, e o *Gestor de Interpretações* pede ao *Gestor de Tarefas* para que o serviço seja executado (ver secção 3.4.2). Quando os três pontos referidos são verificados, existe um processo de escolha da pergunta a ser feita ao utilizador. Esta pergunta tem como objectivo não apenas preencher um espaço de informação do enquadramento do domínio corrente, mas o espaço de informação que for mais útil na resolução das tarefas. A análise do ARP é essencial para atingir esse mesmo objectivo: (i) extracção das melhores regras; e (ii) extracção das regras relevantes. As melhores regras são aquelas que obtiveram um maior valor probabilístico de reconhecimento ou VPREC. Pode acontecer existir mais do que uma regra com o mesmo valor. De entre estas regras, são extraídas as regras relevantes, aquelas que vão ser usadas para determinar o espaço de informação sobre o qual irá recair a próxima pergunta a ser feita ao utilizador.

Extracção das regras relevantes

Se houver um único serviço ou regra com melhor classificação, são consideradas como regras relevantes todas as regras atómicas interiores a essa regra (referente a um espaço de informação). Dado que não existe possibilidade de escolha para perguntas entre espaços de informação de duas regras diferentes, qualquer uma dessas regras interiores tem a mesma probabilidade de ser a escolhida. Por exemplo, se apenas a primeira regra da figura 3.37 constasse no ARP actual, qualquer uma das duas regras *slot_value* poderia ser escolhida como referência para a pergunta a ser feita. Nesse caso seria feita uma pergunta sobre o espaço de informação *target*, ou sobre o espaço de informação *zone* no enquadramento do domínio da domótica, com `50%` de hipóteses para cada. Como existem três regras de classificação no ARP actual com igual probabilidade de serem escolhidas (0.09), é construída uma árvore para averiguar qual dos dois espaços de informação envolvidos nas três regras garante a obtenção de um objectivo mais rapidamente.

A figura 3.38 descreve a árvore criada para a selecção das regras relevantes. Cada nó da árvore é um estado. A raiz da árvore, ou estado inicial, é representativa da informação actual do enquadramento para o domínio em causa (ainda não é contabilizada qualquer regra de classificação). Os sucessores de um estado são dados pelas combinações possíveis de espaços de informação e valores existentes nas regras de classificação. Assim, como sucessores para o estado inicial, obtêm-se os cinco estados apresentados na figura 3.38. Repare-se que no nível seguinte, o estado que contém o espaço de informação *zone* pode tomar dois valores distintos (*quarto* ou *cozinha*), quando o *target* toma o valor de *luz*, enquanto que para os restantes estados, apenas um valor pode ser atribuído (de acordo com as regras de

```
<rule deviceID='3' prob='0.09'>
  <and deviceID='' prob='0.09'>
    <slot_value required='false' slotKey='2' prob='0.3'
                actualValue='' domain='Domotica'
                spectype='target' maintype=''
                modetype='word'>luz</slot_value>
    <slot_value required='false' slotKey='3' prob='0.3'
                actualValue='' domain='Domotica'
                spectype='zone' maintype=''
                modetype='word'>quarto</slot_value>
  </and>
</rule>

<rule deviceID='3' prob='0.09'>
  <and deviceID='' prob='0.09'>
    <slot_value required='false' slotKey='2' prob='0.3'
                actualValue='' domain='Domotica'
                spectype='target' maintype=''
                modetype='word'>luz</slot_value>
    <slot_value required='false' slotKey='3' prob='0.3'
                actualValue='' domain='Domotica'
                spectype='zone' maintype=''
                modetype='word'>cozinha</slot_value>
  </and>
</rule>

<rule deviceID='3' prob='0.09'>
  <and deviceID='' prob='0.09'>
    <slot_value required='false' slotKey='2' prob='0.3'
                actualValue='' domain='Domotica'
                spectype='target' maintype=''
                modetype='word'>tv</slot_value>
    <slot_value required='false' slotKey='3' prob='0.3'
                actualValue='' domain='Domotica'
                spectype='zone' maintype=''
                modetype='word'>sala</slot_value>
  </and>
</rule>
```

Figura 3.37: Exemplo de regras de classificação disponíveis para execução no domínio da Domótica.

classificação). As folhas da árvore ou estados finais são estados em que já não será possível introduzir mais informação relevante das regras de classificação.

Os valores apresentados na figura 3.38 são atribuídos da base para o topo. Os valores dos estados finais são sempre equivalentes a 1, enquanto que para os estados superiores, os seus valores são dados pela soma dos valores dos estados filhos. A ideia desta árvore é

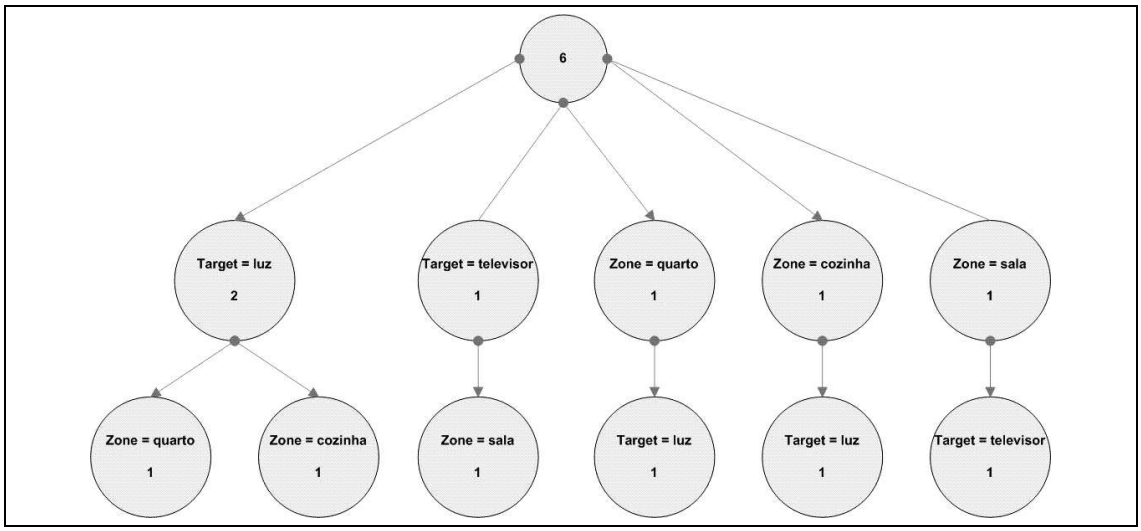

Figura 3.38: Árvore construída para as regras de classificação da figura 3.37 na selecção dos regras relevantes.

contabilizar todos os possíveis estados na decisão da regra mais relevante. Desta forma, se o sistema perguntar pelo valor do espaço de informação *target*, o utilizador poderá responder *luz* ou *televisor*. Se o utilizador responder com "luz", o sistema será levado a perguntar pelo valor do espaço de informação *zone*, justificado pelos possíveis valores que este espaço de informação pode tomar. Se o utilizador responder com "televisor", o valor para o espaço de informação *zone* é automaticamente inferido. Se se contabilizarem as diferentes e possíveis resoluções para o problema, verifica-se que se o sistema perguntar por um valor para o espaço de informação *zone*, a restante informação é imediatamente inferida, qualquer que seja a resposta dada pelo utilizador. Para as regras de classificação apresentadas na figura 3.37, é mais adequado perguntar por um valor do espaço de informação *zone* do que por um valor do espaço de informação *target*. Isto é dado, na prática, pela selecção no primeiro nível da árvore, do menor dos maiores valores atribuídos a cada um dos espaços de informação presentes. Enquanto que para o espaço de informação *target*, o maior valor é 2, para o espaço de informação *zone* o maior valor é 1.

- **Confirmação:** criada quando o sistema infere um ou mais valores possíveis para execução de um serviço. A figura 3.39 ilustra o caso em que existem apenas dois dispositivos disponíveis para controlo. Ambas as luzes da mesa e da sala estão inicialmente desligadas. No exemplo, depois de ligar a luz da mesa, o utilizador pretende ligar possivelmente uma outra luz. Porque a luz da sala é a única disponível a ser ligada, o sistema constrói uma pergunta de confirmação. Se o utilizador responder com "Não" à pergunta de confirmação, o serviço não é executado, e o diálogo continua com uma pergunta do sistema do género "O que deseja então fazer ?";

- **Validação do estado:** Para que esta obrigação de discurso seja criada, as seguintes condições devem ser verificadas: (1) o valor probabilístico de reconhecimento ou VPREC deve ser igual a zero; (ii) existe um serviço não disponível no estado actual que serve os

```
UTILIZADOR: Liga a luz da mesa
SISTEMA: A luz da mesa foi ligada!
UTILIZADOR: Liga a luz
SISTEMA: Quer ligar a luz da sala ?
UTILIZADOR: Sim
SISTEMA: A luz da sala foi ligada!
```

Figura 3.39: Exemplo da obrigação de discurso "Confirmação".

intentos do utilizador. Na figura 3.40, quando o utilizador procura ligar pela segunda vez o mesmo dispositivo, o sistema verifica que a luz da mesa já se encontra ligada, e informa o utilizador dessa mesma situação;

```
UTILIZADOR: Liga a luz da mesa
SISTEMA: A luz da mesa foi ligada!
UTILIZADOR: Liga a luz da mesa
SISTEMA: A luz da mesa já se encontra ligada.
```

Figura 3.40: Exemplo da obrigação de discurso "Validação do estado".

- **Validação do discurso:** criada quando o discurso actual é considerado inválido e requer uma correcção por parte do utilizador (ver figura 3.41);

```
UTILIZADOR: Quero comprar um bilhete de Lisboa para Lisboa
SISTEMA: A cidade de origem e a cidade de destino não podem
         ser iguais. Por favor, indique uma nova cidade de
         origem.
```

Figura 3.41: Exemplo da obrigação de discurso "Validação do Discurso".

Para que esta obrigação de discurso seja criada, as seguintes condições devem ser verificadas: (1) o valor probabilístico de reconhecimento ou VPREC (ver secção 3.4.1) deve ser diferente de 0; (2) o valor probabilístico de resposta ou VPRESP deve ser menor do que 1. O VPRESP é menor do que 1 quando está a ser infrigida uma regra de validação. As regras de validação de um dispositivo são resolvidas assim que são detectadas.

- **Apresentação de resultados:** criada quando o sistema pretende apresentar ao utilizador os resultados da execução de determinado serviço (ver figura 3.42).

```
SISTEMA: Tem o seguinte bilhete disponível: Partida de
         lisboa, com destino a porto às 08:30, e
         chegada prevista às 12:25.
```

Figura 3.42: Exemplo da obrigação de discurso "Apresentação de Resultados".

Para que a obrigação de discurso "Apresentação de resultados" seja criada, o valor probabilístico de reconhecimento ou VPREC (ver secção 3.4.1) deve ser igual a 1. Caso seja menor do que 1, a

obrigação pode ser usada caso se verifiquem as seguintes três condições: (i) existe uma única melhor regra de classificação no ARP; (ii) todas as regras atómicas interiores são do tipo *slot_value*, ou seja, são esperados valores específicos para as regras; e (iii) nenhuma das regras atómicas interiores tem o atributo *required* a *true* (ver figura 3.37), ou seja, nenhuma exige que o utilizador forneça um valor para a regra, em vez de ser inferido. Apenas quando estes três pontos são satisfeitos é possível inferir a informação necessária para executar um serviço.

Existem ainda obrigações de discurso que podem ser independentes ou dependentes do domínio:

- **Clarificação (independente do domínio):** criada quando o sistema não consegue fazer uso das interpretações criadas para decidir a próxima acção. Isto acontece, por exemplo, quando nenhum dos objectos nas interpretações é reconhecido como pertencendo a um domínio activo no sistema (ver figura 3.43).

```
UTILIZADOR: quero um gelado
SISTEMA: Não percebi!
```

Figura 3.43: Exemplo da obrigação de discurso "Clarificação" independente do domínio.

- **Clarificação (dependente do domínio):** criada quando o sistema não consegue executar o pedido feito pelo utilizador. Esta obrigação é gerada sempre que: (i) o valor probabilístico de reconhecimento ou VPREC é igual a 0; (ii) não existe nenhum serviço num estado que não o actual que possa servir os intentos do utilizador.

```
UTILIZADOR: liga os estores
SISTEMA: Uma acção do tipo ligar não pode ser realizada sobre os estores .
UTILIZADOR: abre a luz
SISTEMA: Uma acção do tipo abrir não pode ser realizada sobre a luz .
```

Figura 3.44: Exemplo da obrigação de discurso "Clarificação" dependente do domínio.

3.7 Gestor do Contexto

O *Gestor do Contexto* possui toda a informação necessária para controlar o comportamento comunicacional do sistema. As principais funções do módulo incluem manter e gerir toda a informação do discurso, a história do diálogo, tarefas e registos de domínio. Este módulo troca informação com o *Gestor de Interpretações* e com o *Gerador de Superfície*.

3.7.1 Interface

- **Inicialização do *Gestor do Contexto*:** esta operação é usada para inicializar todas as variáveis implicadas na manutenção e gestão de toda a informação;

- **Actualização do domínio corrente:** esta operação é usada para actualizar o domínio corrente no *Gestor do Contexto*. O domínio corrente é dado pelo nome do domínio sobre o qual decorre o actual diálogo;

- **Envio do domínio corrente:** esta operação devolve o domínio actual;

- **Actualização do estado do diálogo:** o estado do diálogo consiste na indicação, para cada um dos domínio activos no sistema, dos valores associados aos espaços de informação dos enquadramentos nos respectivos domínios. A figura 3.22 mostra o estado do diálogo imediatamente após o utilizador proferir "Quero comprar um bilhete para o Porto.". A actualização do estado do diálogo pode ser feita com o envio de um novo estado do diálogo, ou apenas com o envio de informação parcial. É possível adicionar ou remover um valor de um espaço de informação de um enquadramento;

- **Envio do estado do diálogo:** esta operação devolve os estados dos diálogos em cada um dos domínios recebidos como argumento;

- **Actualização das interpretações:** todas as interpretações ocorridas no decorrer do diálogo são mantidas no *Gestor do Contexto*. Esta operação faz a inserção de novas interpretações no *Gestor do Contexto*;

- **Envio das interpretações:** esta operação devolve as interpretações ocorridas até ao momento no decorrer do diálogo;

- **Actualização das obrigações de discurso:** todas as obrigações de discurso criadas no *Gestor de Interpretações* são mantidas no *Gestor do Contexto*. Esta operação faz a inserção de novas obrigações de discurso no *Gestor do Contexto*;

- **Envio das obrigações de discurso:** esta operação devolve as obrigações de discurso ocorridas até ao momento no decorrer do diálogo;

- **Activação das obrigações de discurso:** tal como foi dito na secção 3.6.2, as obrigações de discurso são usadas para fazer o mapeamento das interpretações com algo que foi dito préviamente pelo sistema. Como nem todas as obrigações criadas poderão ser seleccionadas para execução futura, e algumas até em momentos distintos (ver secção 3.8), existe a necessidade de activar apenas aquelas que foram escolhidas para execução. Esta operação é usada pelo *Gerador de Superfície* com esse mesmo fim, e tem como argumento os identificadores das obrigações de discurso que foram escolhidas para execução;

- **Mapeamento das interpretações com as obrigações de discurso:** esta operação permite ao sistema interpretar as respostas do utilizador em função do contexto. Recebe como argumento as interpretações criadas até ao momento pelo *Gestor de Interpretações* e devolve um de três resultados possíveis: (i) falso; (ii) verdadeiro ; (iii) as interpretações que mapeiam com as obrigações de discurso. Esta operação verifica se as interpretações mapeiam com alguma das obrigações de discurso que estejam activas. Se não houver obrigações activas,

o resultado é "falso". Caso haja mapeamento, a obrigação de discurso com a qual foi realizado o mapeamento é removida da lista de obrigações activas. Existem neste momento, quatro obrigações de discurso com a qual é procurado fazer o mapeamento: (i) cumprimento (ver figura 3.31); (ii) desambiguação de domínio (ver figura 3.34); (iii) desambiguação de espaço de informação (ver figura 3.35); e (iv) pedido de valor para um espaço de informação (ver figura 3.36).

- **Cumprimento:** o mapeamento com esta obrigação de discurso é realizada se houver uma única interpretação cujo tipo (dado pela propriedade *type* da interpretação) seja *cumprimento*;
- **Desambiguação de domínio:** o mapeamento com esta obrigação de discurso é realizada se houver uma única interpretação cujo objecto seja um dos domínios da obrigação a desambiguar;
- **Desambiguação de espaço de informação:** o mapeamento com esta obrigação de discurso é realizada se houver uma única interpretação cujo tipo seja "afirmação", e cujo objecto esteja relacionado com o tipo de slot relacionado com a desambiguação. Na figura 3.45, *Lisboa* foi reconhecida como sendo do tipo *cidadeorigem* e do tipo *cidadedestino*, o que originou a pergunta. A resposta dada pelo utilizador *cidade de origem* contém objectos, neste caso dois, *cidade* e *origem*, que se relacionam com um dos tipos dos espaços de informação da obrigação, e por isso, determinam o espaço a preencher;

```
UTILIZADOR: Lisboa.
SISTEMA: Lisboa é cidade de origem ou é cidade de destino?
UTILIZADOR: cidade de origem !
```

Figura 3.45: Exemplo de desambiguação de espaço de informação.

- **Pedido de valor para um espaço de informação:** o mapeamento com esta obrigação do discurso é realizada se houver uma interpretação cujo tipo seja "afirmação" ou "pedido", e um dos objectos dessa interpretação seja reconhecido como tendo o domínio e o tipo de espaço de informação igual ao requisitado na obrigação de discurso e pedido pelo sistema. Na sequência da pergunta do sistema apresentada na figura 3.46, o utilizador responde com *Porto*. Neste caso, *Porto* é reconhecido como um objecto que pertence ao domínio dos Autocarros, e que pode tanto preencher um espaço de informação relativo a uma cidade de origem, como um espaço de informação relativo a uma cidade de destino. Das duas interpretações criadas (ver secção 3.6.1), a que contém o objecto *Porto* como *cidadedestino* ajusta-se à pergunta feita pelo sistema, e corresponde à interpretação resultado do mapeamento.

```
SISTEMA: Para onde deseja viajar?
UTILIZADOR: Porto.
```

Figura 3.46: Exemplo de pedido de valor para um espaço de informação.

3.8 Gestor de Comportamentos e Gestor de Geração

O *Gestor de Comportamentos* é responsável pelo comportamento de todo o sistema. Escolhe entre as obrigações do discurso qual é aquela que é melhor aplicável num determinado momento. Recebe uma lista de obrigações de discurso do *Gestor de Interpretações* e devolve uma lista daquelas que serão as obrigações prioritárias a executar no momento. No final, o *Gestor de Comportamentos* envia as obrigações do discurso para o *Gerador de Geração*.

O *Gerador de Geração* recebe as obrigações de discurso seleccionadas para execução (enviadas pelo *Gestor de Comportamentos*) e pede ao *Gestor do Contexto* que active as obrigações de discurso correspondentes. É com as obrigações de discurso activadas que o sistema procura fazer o mapeamento das interpretações (ver secção 3.7). No final, *Gestor de Geração* envia as obrigações de discurso que recebeu como entrada para o *Surface Manager* (ver secção 3.9).

3.9 Gerador de Superfície

O *Gerador de Superfície* é um componente importante no sistema de diálogo. Tem de transformar em língua natural as intenções do sistema para que haja melhor comunicação com o utilizador. Existem dois tipos de frases a serem geradas para o utilizador: (i) frases independentes do domínio; e (ii) frases dependentes do domínio. As frases independentes do domínio estão descritas num ficheiro próprio, e são carregadas para o sistema juntamente com o lançamento do módulo. As frases dependentes fazem parte da descrição de um domínio, e como tal, estão colocadas à disposição a partir da interface do *Gestor de Serviços* (ver secção 3.3.2). A comunicação com o *Gestor de Serviços* para obtenção das frases de um domínio é efectuada apenas na primeira vez em que tal se verifique necessário.

Na descrição de um domínio (ver secção 3.1), as frases a serem usadas na geração aparecem incluídas em regras. Existe uma regra de geração para cada obrigação de discurso e espaço de informação possível. As frases são baseadas em modelos, o que significa que a partir delas poderão ser aplicados um conjunto de substituições com informação relevante para a geração da frase pretendida. Os valores de substituição são fornecidos na obrigação de discurso. Do conjunto de frases fornecidas, é seleccionada para geração não apenas a frase cuja obrigação de discurso fornecer os elementos requeridos, como também aquela ou aquelas que fornecerem mais informação ao utilizador. O objectivo desta estratégia passa por confirmar os elementos já fornecidos pelo utilizador (verificação implícita - ver secção 2.4.6). Para exemplificar, se nenhuma informação for fornecida na obrigação de discurso, qualquer uma das duas primeiras frases da figura 3.47 pode ser seleccionada para geração. Caso a obrigação contenha apenas informação sobre a hora de partida, a terceira frase será a seleccionada.

Os elementos requeridos e a serem substituídos numa frase aparecem entre parênteses rectos e com o símbolo do `$` na primeira posição. O primeiro elemento entre parênteses rectos corresponde

```
<rule id="4" uttgenerationtype="requestslot"
             uttgenerationslot="cidadedestino">
 <one-of>
  <item>Qual é a cidade de destino? </item>
  <item>Para onde deseja ir? </item>
  <item>Para onde deseja viajar, às [$horapartida]? </item>
  <item>Para onde deseja viajar, para chegar às
        [$horachegada]? </item>
  <item>Para onde deseja viajar, com partida de
        [$cidadeorigem]? </item>
  <item>Para onde deseja viajar, com hora de chegada às
        [$horachegada]? </item>
  <item>Para onde deseja viajar, com partida às
        [$horapartida] e cidade de
        origem [$cidadeorigem]?</item>
  <item>Para onde deseja viajar, [$#no $diasemana]
        [$diasemana], com partida às [$horapartida] e
        cidade de origem [$cidadeorigem]?</item>
 </one-of>
</rule>
```

Figura 3.47: Exemplo de frases a serem utilizadas no domínio dos autocarros.

```
<word value='no'>
  <form gen='m' card='s' value='no'/>
  <form gen='m' card='p' value='nos'/>
  <form gen='f' card='s' value='na'/>
  <form gen='f' card='p' value='nas'/>
</word>

<word value='segunda-feira'>
  <form gen='f' card='s' value='segunda-feira'/>
  <form gen='f' card='p' value='segundas-feiras'/>
</word>
```

Figura 3.48: Parte da descrição com as substituições a efectuar para as palavras.

à palavra a substituir, enquanto que os restantes elementos correspondem ao tempo em que o primeiro elemento vai aparecer na frase gerada. A informação temporal de todas as palavras a serem usadas na geração é dada num ficheiro XML. A figura 3.48 apresenta parte do ficheiro XML usado, com informação acerca do elemento *no* e do elemento *segunda-feira*. Todos os elementos que na segunda posição tenham o símbolo de cardinal são elementos cujo valor de substituição vai ser fornecido apenas com base na informação do ficheiro. Para todos os outros elementos, a substituição vai ser efectuada não só com base no ficheiro XML, mas também com a informação providenciada na obrigação de discurso.

A regra da figura 3.47 é utilizada para a obrigação de discurso *requestslot* (pedido de valor para um espaço de informação) sobre o espaço de informação *cidadedestino*. O atributo *uttgenerationtype* é usado para especificar o tipo de obrigação, enquanto que *uttgenerationslot* é utilizado para especificar o espaço de informação em causa. Um elemento *item* contém a frase a gerar, enquanto que o elemento *one-of* contém todas as frases que é possível utilizar. Apenas uma delas será utilizada na geração.

Para exemplificar, repare-se na última do conjunto de frases fornecido na figura 3.47 para a pergunta a gerar acerca da cidade de destino. Suponha-se que a obrigação de discurso actual fornece "segunda-feira" como valor para *diasemana*, "19:00" para *horapartida*, e "Lisboa" para *cidadeorigem*. A frase diz que a palavra *no* deve aparecer com o tempo do valor a substituir *diasemana*. Como o dia da semana pretendido é "segunda-feira", e segundo o ficheiro de descrição XML, é uma palavra feminina singular, a palavra *no* deve aparecer também no feminino e singular, o que corresponde a obter como resultado a palavra "na". Após o mesmo cálculo para os restantes elementos (estes não necessitam de informação temporal), obtém-se a seguinte frase como saída do módulo: "Para onde deseja viajar, na segunda-feira, com partida às 19:00 e cidade de origem Lisboa?".

Dependendo da forma como a frase é construída, a informação temporal para uma determinada palavra pode ser dada explicitamente na frase a processar. Para a frase da figura 3.49, usada no domínio da Domótica, qualquer que seja a acção (*action*) dada na obrigação de discurso, ela será gerada no particípio passado.

```
<item>[$#o $target] [$target] [$#ir $target][$action pp $target]
      para [$#o $direction] [$direction]!</item>
```

Figura 3.49: Exemplo de frase utilizada no domínio da Domótica.

Quando é fornecido mais do que um elemento para caracterizar o tempo, como é o caso em `[$action pp $target]`, é feita a reunião da informação temporal necessária para gerar *action* correctamente. A prioridade dos items temporais é feita da esquerda para a direita, pelo que se *target*, no exemplo, aparecer noutro tempo, é sempre considerado o tempo do primeiro, ou seja, de *pp*. A frase gerada a partir da modelo apresentada na figura 3.49 poderá ser: "Os estores foram rodados para a esquerda!".

3.10 Integração do Gestor de Diálogos com Informação Linguística

A integração dos módulos de informação linguística com o gestor de diálogos ocorre através da construção dos actos de fala. Com esta integração, os actos de fala deixam de ser construídos manualmente, para passarem a ser construídos a partir do resultado do processamento dos vários módulos de informação linguística. A nova construção dos actos de fala obrigou a efec-

tuar mudanças no comportamento dos módulos, essencialmente no *Gestor de Interpretações* (ver secção 3.6). Alguns fluxos de informação tiveram de ser redefinidos, e o conceito de interpretação (ver secção 3.6.1) dada no *Gestor de Interpretações* deixou de existir para estar englobado na definição de acto de fala. Em suma, a interpretação que o sistema faz da frase proferida pelo utilizador engloba-se na definição de acto de fala, e os requisitos necessários à construção das interpretações passam a ser os requisitos de construção dos actos de fala. Funções como o reconhecimento dos objectos ou o mapeamento das interpretações com as obrigações de discurso existentes no *Gestor do Contexto* (ver secção 3.7) passam agora a ser usadas na construção dos actos de fala. Estas funções, associadas ao processo de descoberta da performativa dos actos de fala (ver secção 2.5), fazem parte dos processos necessários à construção dos actos de fala (ver figura 3.50). Ao módulo responsável pela construção dos actos de fala denominou-se *"Speech Act Finder (SAF)"*. Este módulo vem substituir o *Reconhecedor da Linguagem* (ver secção 3.5).

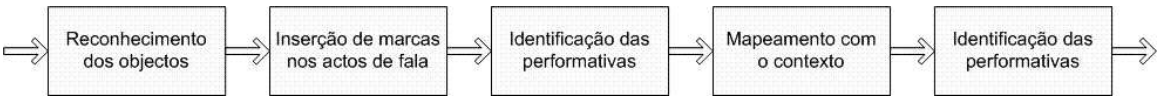

Figura 3.50: Processos do *Speech Act Finder*.

Enquanto que a entrada do módulo *Reconhecedor da Linguagem* é idêntica à saída do reconhecedor, ou seja, é uma sequência de palavras sem nenhuma informação associada, a entrada do *SAF* é constituída por uma forma primitiva de actos de fala alcançada através do processamento dos módulos de informação linguística. A figura 3.51 apresenta os módulos de informação linguística.

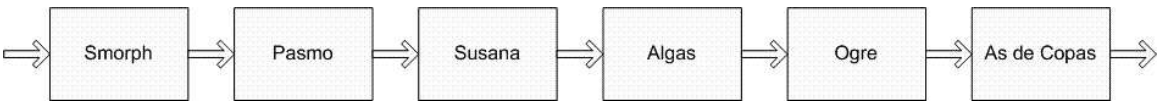

Figura 3.51: Módulos de informação linguística.

A saída do reconhecedor é agora a entrada do módulo *Smorph*, enquanto que a saída do módulo *As de Copas* é agora a entrada do módulo *SAF*. O *Smorph* é uma ferramenta que permite fazer a análise morfológica de um texto ou de uma cadeia de caracteres e que disponibiliza uma interface para facilitar a geração dos dicionários binários a serem utilizados na análise. O *Pasmo* é uma ferramenta de pós-análise morfológica, enquanto que o *Susana* faz a análise sintáctica do texto, que usando uma gramática de superfície, agrupa os constituintes das frases. Os outros três módulos de processamento de informação linguística são responsáveis por efectuar uma análise semântica do texto. O *Algas* é constituído por um algoritmo de pré-análise semântica, e é responsável por conectar pedaços de texto e os elementos dentro dos mesmos, tomando como entrada propriedades de flechagem e informação que limita o espaço de procura. O *Ogre* gera um grafo a partir das estruturas produzidas pelo *Algas*, e o *As de Copas* usa regras semânticas hierarquicamente organizadas para construir representações semânticas. Uma destas representações semânticas produzidas pelo *As de Copas* constitui a entrada do módulo SAF.

O reconhecimento dos objectos (ver figura 3.50) é aplicado sobre a representação semântica dos actos de fala produzido pelo *As de Copas*. O reconhecimento elaborado nesta fase é em tudo

idêntico ao elaborado previamente no *Gestor de Interpretações* e descrito em 3.6.1. Em nenhuma fase da construção dos actos de fala no *SAF* são eliminados candidatos à interpretação correcta do que foi proferido pelo utilizador. Em vez da remoção, o *SAF* marca os actos de fala que revelem incongruências na sua estrutura. Esta fase está assinalada na figura 3.50 com "Inserção de marcas nos actos de fala". A remoção dos candidatos acontece apenas no *Gestor de Interpretações*. As marcas são colocadas a partir de uma descrição XML apresentada em parte na figura 3.52. Existem actualmente dois tipos de regras principais: (i) *domainallowed*; e (ii) *domaincontradiction*. Qualquer um dos tipos especifica uma relação entre uma propriedade de um acto de fala (*saproperty*) e informação proveniente do reconhecimento do objecto (*spectype*). Por exemplo, a segunda regra da figura 3.52 afirma que os objectos classificados como *target* podem ser incluídos na propriedade *what* do acto de fala. No entanto, a terçeira regra da mesma figura afirma que uma *cidadeorigem* não pode ser incluída numa propriedade *towhere* de um acto de fala. A razão pela qual existem estes dois tipos está relacionada com a especificidade das respostas que o sistema fornece para o exterior, podendo ser diferente em cada caso.

A figura 3.53 descreve uma amostra de um dos dois actos de fala criados para a frase "Quero um bilhete para o Porto"[6]. Procurou-se especificar onde ocorrem as marcas no acto de fala. Para isso, foi criado o atributo *marks* na propriedade dos actos de fala. O acto de fala também contém informação de todas as marcas que ocorrem em todas as suas propriedades.

Passadas as duas primeiras fases, é tempo de procurar identificar as performativas dos actos de fala (ver secção 2.5). Esta identificação é elaborada em duas fases, antes e depois de ser realizado o "mapeamento com o contexto", tal como descrito na figura 3.50. Descreve-se aqui um pequeno exemplo da necessidade desta separação, mas poder-se-iam dar muitos outros. Suponha-se que o utilizador diz "sim". Sem qualquer contexto, esta frase não passa de uma afirmação, porventura estranha, mas a resposta do sistema de diálogos deverá ser produzida de acordo com essa mesma afirmação. Por outro lado, se "sim" for resposta a uma pergunta do sistema do tipo "Quer ir para o Porto?", "sim" deixa de ser apenas uma afirmação para passar a ser também uma confirmação. A identificação da performativa de um acto de fala é dependente do contexto em que se insere o discurso.

A escolha da performativa é efectuada com base em heurísticas e em dicionários. As heurísticas são constituídas por um conjunto de primitivas, tal como a apresentada na figura 3.54. *weight* traduz o peso a atribuir à heurística caso a função em *name* com o argumento *openings* seja verificada. Neste caso, será verificado se o acto de fala contém alguma das palavras que aparecem no dicionário de nome *openings* (ver figura 3.55). Se isso acontecer, será atribuída à primitiva "Cumprimento" o valor 1.0. Contabilizadas todas as heurísticas e primitivas do sistema, ficará associado ao acto de fala uma lista de performativas com um peso atribuído a cada uma. Quanto maior for este peso, maior será a probabilidade da performativa a que está o peso associado ser a mais correcta para o acto de fala.

[6]Tal como foi descrito na secção 3.6.1 são criados dois actos de fala porque "Porto" é reconhecido como sendo uma cidade de origem e uma cidade de destino.

```
<speechactrules>

  <rule>
    <type>domainallowed</type>
    <description>The specttypes that are allowed with
                 the saproperty</description>
    <saproperty>action</saproperty>
    <spectype>action</spectype>
  </rule>
  <rule>
    <type>domainallowed</type>
    <description>The specttypes that are allowed
                 with the saproperty</description>
    <saproperty>what</saproperty>
    <spectype>target</spectype>
  </rule>
  ...
   <rule>
    <type>domaincontradiction</type>
    <description>Cidade de origem nao pode ser
                 towhere</description>
    <saproperty>towhere</saproperty>
    <spectype>cidadeorigem</spectype>
  </rule>
  <rule>
    <type>domaincontradiction</type>
    <description>Cidade de destino nao pode ser
                 fromwhere</description>
    <saproperty>fromwhere</saproperty>
    <spectype>cidadedestino</spectype>
  </rule>
```

Figura 3.52: Exemplo de algumas regras usadas na marcação dos actos de fala.

Após a realização da primeira fase de identificação das performativas associadas a um acto de fala, é necessário efectuar o mapeamento de todos os actos de fala construídos até ao momento com o contexto. Este mapeamento é idêntico ao descrito na secção 3.6.1. O processo principal encontra-se descrito em 3.7. Como já foi referido, após este mapeamento ocorre a segunda fase da identificação das performativas. No final das cinco etapas apresentadas na figura 3.50, é enviado para o *Gestor de Interpretações* todos os actos de fala identificados até ao momento. As funcionalidades neste último módulo incluem agora duas fases importantes: (i) análise das marcas inseridas nos actos de fala e das performativas associadas aos mesmos; (ii) construção das obrigações de discurso (ver secção 3.6.2).

```
<SpeechActs>
 <speechAct phrase = 'Quero um bilhete para o Porto'
            marks = 'DomainNotAllowed DomainContradiction '>
 ...
  <property type='toWhere'
            marks = 'DomainNotAllowed DomainContradiction '>
   <item>
    <name>porto</name>
    <domain>autocarros</domain>
    <device></device>
    <maintype>cidadeorigem</maintype>
    <modetype>word</modetype>
    <spectype>cidadeorigem</spectype>
    <value>0.0</value>
    <comp></comp>
   </item>
  </property>
  ...
 </speechAct>
 ...
</SpeechActs>
```

Figura 3.53: Exemplo de marcação na criação dos actos de fala.

```
<?xml version="1.0" encoding="UTF-8"?>
<heuristic id="Cumprimento">
  <primitive>
    <weight>1.0</weight>
    <name>ContainsWords</name>
    <argument>openings</argument>
  </primitive>
</heuristic>
```

Figura 3.54: Exemplo de heurística associada à performativa "Cumprimento".

```
bom dia
ola
boa noite
boa tarde
passou bem
tudo bem
como esta
viva
```

Figura 3.55: Amostra do dicionário associado à performativa "Cumprimento".

3.11 Sumário

Este capítulo apresentou uma descrição pormenorizada das estratégias seguidas na implementação do gestor de diálogos. Definiu os conceitos de domínio e de dispositivo, bem como o módulo responsável pela gestão da informação envolvida, para depois apresentar em pormenor o funcionamento de cada um dos módulos que compõem o gestor de diálogos. Para auxiliar na explicação, foram usados exemplos dos dois domínios construídos na tese: (i) Domínio da Domótica; e (ii) Domínio dos Autocarros. No final do capítulo descreveu-se uma nova versão do módulo *Reconhecedor da linguagem* para a construção dos actos de fala.

Capítulo 4

Resultados e Avaliação

Este capítulo começa por descrever a interface do sistema de diálogos, para depois apresentar exemplos de diálogos produzidos em dois domínios distintos: (i) Domótica; e (ii) Autocarros. Para cada exemplo são explicadas as principais intervenções do sistema no decorrer do diálogo. Estas intervenções surgem na sequência das estratégias descritas no capítulo 3. Neste capítulo é também feita uma avaliação do sistema de diálogo. São descritas as características do sistema e dos diálogos produzidos.

4.1 Interface do Sistema de Diálogos

A figura 4.1 mostra a interface usada no sistema de diálogos. Esta interface permite:

- Visualizar as mensagens trocadas entre os módulos do sistema;
- Controlar os parâmetros de áudio do reconhecedor de fala;
- Controlar a disponibilidade dos dispositivos (públicos, privados, ou restrictos a determinados utilizadores);
- Definir os utilizadores do sistema;
- Disponibilizar informação adicional sobre o sistema;
- Definir uma "casa virtual" que engloba um conjunto de dispositivos de acordo com a sua localização.

Em cima e à esquerda na figura 4.1 aparece a imagem da face representativa do sistema, também desenvolvida no Laboratório. A introdução via teclado dos pedidos no sistema aparece na caixa de texto abaixo de "O que pretende?", ao passo que na caixa abaixo de "Diálogo" aparece uma transcrição de tudo o que é dito para o sistema e o próprio gera como resposta.

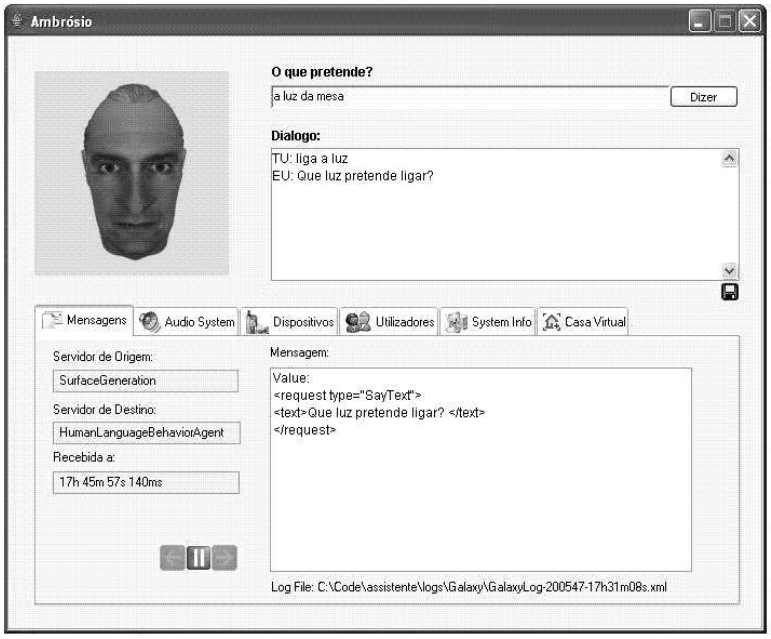

Figura 4.1: Interface do sistema de diálogos.

4.2 Apresentação de Diálogos

Nesta secção apresentam-se alguns dos diálogos produzidos no domínio da Domótica e no domínio dos Autocarros. Os exemplos foram retirados directamente da interface do sistema de diálogos, mas para efeitos de apresentação, a primeira letra relativa a cada frase do utilizador foi tornada maiúscula.

4.2.1 Diálogos no domínio da Domótica

No domínio da Domótica, controlam-se cinco dispositivos: (i) a luz da sala; (ii) a luz da mesa; (iii) o ar condicionado; (iv) os estores; e (v) a aparelhagem. A descrição do domínio e dos dispositivos é fornecida no apêndice A desta dissertação. A sequência apresentada nos diálogos seguintes é relevante para a compreensão dos mesmos. No estado inicial, a luz da sala, a luz da mesa, o ar condicionado, e a aparelhagem encontram-se desligadas. Os estores encontram-se inicialmente fechados.

```
(1)  UTILIZADOR: Olá
(2)  SISTEMA: Olá
```

O "Olá" é interpretado como um cumprimento. A resposta do sistema a um cumprimento consiste num cumprimento semelhante.

```
(3)  UTILIZADOR: Liga
(4)  SISTEMA: O que deseja ligar?
```

A pergunta do sistema é justificada pelo facto de haver vários dispositivos que o utilizador possa "ligar".

```
(5)  UTILIZADOR: A luz
(6)  SISTEMA: Que luz pretende ligar?
```

Podem ser ligadas duas luzes: (i) a luz da sala; e (ii) a luz da mesa. A pergunta em (6) é feita com o objectivo de resolver a ambiguidade.

```
(7)  UTILIZADOR: A luz da mesa
(8)  SISTEMA: A luz da mesa foi ligada!
```

Agora que todas as ambiguidades foram resolvidas e o sistema finalmente interpretou completamente o pedido do utilizador, o pedido foi executado e a luz da mesa foi ligada. O estado associado ao dispositivo "luz da mesa" passa assim de "desligado" para "ligado", influenciando assim os futuros pedidos.

```
(9)  UTILIZADOR: Liga a luz
(10) SISTEMA: Quer ligar a luz da sala?
(11) UTILIZADOR: Sim
(12) SISTEMA: A luz da sala foi ligada!
```

Não existe agora qualquer ambiguidade associada ao novo pedido do utilizador, já que a única luz que é possível "ligar" é a "luz da sala". Porque o utilizador não afirmou directamente que queria ligar a luz da sala, o sistema gera uma pergunta de confirmação. Para além da informação previamente fornecida pelo utilizador, esta pergunta contém também o(s) elemento(s) que necessitam de uma confirmação ("luz da sala"). O utilizador responde à confirmação com "Sim" e o serviço é imediatamente executado.

```
(13) UTILIZADOR: Não
(14) SISTEMA: Não existe nada por confirmar!
```

O sistema não espera nenhuma confirmação do utilizador, e informa o utilizador dessa condição.

```
(15) UTILIZADOR: Desliga a luz
(16) SISTEMA: Que luz pretende desligar?
(17) UTILIZADOR: A luz da mesa
(18) SISTEMA: A luz da mesa foi desligada!
```

Porque estavam agora as duas luzes ligadas, o sistema foi obrigado a perguntar ao utilizador qual das duas luzes o utilizador queria desligar.

```
(19) UTILIZADOR: Liga
(20) SISTEMA: O que deseja ligar?
(21) UTILIZADOR: Desejo ligar o ar condicionado
(22) SISTEMA: O ar condicionado foi ligado!
(23) UTILIZADOR: Aumenta a temperatura do ar condicionado
(24) SISTEMA: A temperatura foi aumentada!
```

Face aos dispositivos e serviços disponíveis, só é possível aumentar a temperatura do ar condicionado. Se o pedido em (23) fosse realizado antes do pedido traduzido no ponto (19) e (21), o sistema responderia que o estado actual não lhe permitia executar essa mesma acção, porque o ar condicionado estaria "desligado".

```
(25) UTILIZADOR: Liga a aparelhagem
(26) SISTEMA: A aparelhagem foi ligada!
(27) UTILIZADOR: Aumenta o volume
(28) SISTEMA: Quer aumentar o volume da aparelhagem?
(29) UTILIZADOR: Não
(30) SISTEMA: O que deseja então fazer?
```

```
(31) UTILIZADOR: A maçã está boa
(32) SISTEMA: Não consegui entender. Pode repetir o que disse,
              se faz favor?
```

Num domínio como o da Domótica, não é reconhecido qualquer termo como o expressado em (31). O sistema não foi capaz de reconhecer qualquer dos objectos expressos na frase face aos dispositivos e serviços disponíveis, pelo que o sistema indica explicitamente na resposta o facto de não ter entendido essa mesma expressão do utilizador.

```
(33) UTILIZADOR: Fecha os estores
```

```
(34) SISTEMA: Os estores já se encontram fechados.
(35) UTILIZADOR: Abre os estores
(36) SISTEMA: Os estores foram abertos!
```

Em (33), o utilizador pede para fechar os estores. Não existe nenhum serviço que sirva o intento no estado actual, mas existe num estado indisponível, ou seja, quando os estores estão "abertos". O sistema informa o utilizador do estado actual do dispositivo.

```
(37) UTILIZADOR: Desliga
(38) SISTEMA: O que deseja desligar?
(39) UTILIZADOR: Diminui
(40) SISTEMA: O que deseja diminuir?
(41) UTILIZADOR: A temperatura do ar condicionado
(42) SISTEMA: A temperatura foi diminuída!
```

É possível, assim como para qualquer outro conceito, refazer intenções. Neste pequeno exemplo, o utilizador começou por dizer que queria "desligar", acabando por emendar com a acção "diminuir".

```
(43) UTILIZADOR: Fecha os estores
(44) SISTEMA: Os estores foram fechados!
(45) UTILIZADOR: Desliga o ar condicionado
(46) SISTEMA: O ar condicionado foi desligado!
(47) UTILIZADOR: Desliga
(48) SISTEMA: O que deseja desligar?
(49) UTILIZADOR: A luz da sala
(50) SISTEMA: A luz da sala foi desligada!
(51) UTILIZADOR: Desliga a luz da sala
(52) SISTEMA: A luz da sala já se encontra desligada.
(53) UTILIZADOR: Desliga a aparelhagem
(54) SISTEMA: A aparelhagem foi desligada!
```

O bloco de diálogo acima coloca todos os dispositivos no seu estado inicial.

```
(55) UTILIZADOR: Desliga
(56) SISTEMA: Não consigo desligar o que quer que seja no estado actual.

Porque os dispositivos estão todos desligados, não existe nenhum que seja possível ap

(57) UTILIZADOR: Liga os estores
(58) SISTEMA: Uma acção do tipo ligar não pode ser realizada sobre os estores.
```

```
(59) UTILIZADOR: Abre a luz
(60) SISTEMA: Uma acção do tipo abrir não pode ser realizada sobre a luz.
```

Não existem serviços com as características indicadas em (57) ou em (59). O sistema informa o utilizador dessa mesma condição.

4.2.2 Diálogos no domínio dos Autocarros

No domínio dos Autocarros, controla-se um dispositivo de venda de bilhetes. A descrição do domínio e do dispositivo é dada no apêndice B desta dissertação.

```
(1)  UTILIZADOR: Quero comprar um bilhete de Lisboa para o Porto
                 com partida às 11:00
(2)  SISTEMA: Em que dia da semana deseja partir com destino a Porto?
(3)  UTILIZADOR: No Domingo
(4)  SISTEMA: Peço desculpa, mas não existe bilhete disponível a essa
              hora.
```

Em (1) o utilizador fornece de imediato 5 informações essenciais à compra de um bilhete no domínio dos Autocarros: (i) *action* (comprar); (ii) *Target* (bilhete); (iii) *CidadeOrigem* (Lisboa); (iv) *CidadeDestino* (Porto); e (v) *HoraPartida* (11:00). Esta informação corresponde a 83% dos espaços de informação do enquadramento do domínio. Para "comprar um bilhete" o serviço exige que se forneça também o dia da semana pretendido para a viagem, o que justifica a pergunta feita pelo sistema em (2). Após uma consulta à base de dados, o sistema não foi capaz de encontrar qualquer horário que se aproximasse ao horário pretendido pelo utilizador.

No exemplo acima, a primeira frase do sistema incluí a sequência de caracteres "a Porto?". Correctamente, a sequência apresentada seria "ao Porto?", mas para isso ter-se-ia de determinar a característica da palavra "Porto" que obriga ao uso de "ao". Como a frase de geração de uma pergunta para o dia da semana contém a letra "a" antes da cidade (ver apêndice B), todas as frases geradas, independentemente da cidade envolvida, vão fazer uso desta letra.

```
(1)  UTILIZADOR: Quero comprar um bilhete de Lisboa para Coimbra
                 com partida às 07:30
(2)  SISTEMA: Em que dia da semana deseja partir com destino
              a Coimbra?
(3)  UTILIZADOR: No sábado
(4)  SISTEMA: A viagem com partida de Lisboa, e destino a Coimbra,
              pode ficar marcada para as 08:30, com chegada às 12:25.
```

A diferença deste bloco de diálogo para o anterior surge nos resultados. Desta vez, foi encontrado na base de dados um horário próximo do pretendido pelo utilizador. Na resposta, o sistema volta a confirmar os parâmetros fornecidos pelo utilizador.

```
(1)  UTILIZADOR: Quero comprar um bilhete
(2)  SISTEMA: A que horas quer partir?
(3)  UTILIZADOR: A partir das 12:00
(4)  SISTEMA: Quer partir de que cidade, às 12:00?
(5)  UTILIZADOR: Quero partir de Lisboa
(6)  SISTEMA: Em que dia da semana deseja partir de Lisboa?
(7)  UTILIZADOR: No Sábado
(8)  SISTEMA: Para onde deseja viajar, no sábado, com partida
              às 12:00 e cidade de origem Lisboa?
(9)  UTILIZADOR: Para Coimbra
(10) SISTEMA: A viagem com partida de Lisboa, e destino a Coimbra,
              pode ficar marcada para as 15:45, com chegada às 19:55.
              A viagem com partida de Lisboa, e destino a Coimbra,
              pode ficar marcada para as 17:00, com chegada às 20:55.
```

À medida que o diálogo se vai desenrolando, o sistema procura confirmar os valores que já foram fornecidos pelo utilizador e que são necessários à compra de um bilhete. Por exemplo, no ponto (4) o sistema confirma a hora de partida, no ponto (6) o sistema confirma a cidade de origem, e no ponto (8) o sistema confirma o dia da semana, a hora e a cidade de partida. Mais do que uma solução pode ser apresentada ao utilizador, tal como acontece em (10).

```
(1)  UTILIZADOR: Lisboa
(2)  SISTEMA: Lisboa é cidade de origem ou é cidade de destino?
(3)  UTILIZADOR: Cidade de origem
(4)  SISTEMA: A que horas quer chegar, partindo de Lisboa?
(5)  UTILIZADOR: Quero chegar antes das 20:00
(6)  SISTEMA: Para onde deseja viajar, com chegada às 20:00 e
              cidade de origem Lisboa?
(7)  UTILIZADOR: Para o Porto
(8)  SISTEMA: Em que dia da semana deseja partir com destino a Porto?
(9)  UTILIZADOR: Sexta-feira
(10) SISTEMA: O que pretende fazer?
(11) UTILIZADOR: Comprar bilhete
(12) SISTEMA: Peço desculpa, mas não existe bilhete disponível a
              essa hora.
```

Apesar de não ser muito convencional, no ponto (1) o utilizador começa por proferir “Lisboa”, sem qualquer diálogo prévio, e portanto, sem qualquer contexto que apoie o que foi dito. Porque “Lisboa”, no domínio

dos Autocarros, tanto pode corresponder a uma cidade de origem ou a uma cidade de destino, o sistema procura imediatamente resolver em (2) a ambiguidade. Porque mais de um serviço está disponível no dispositivo de venda de bilhetes, em (10) o sistema pergunta ao utilizador o que este pretende realizar com a informação fornecida anteriormente.

```
(1)  UTILIZADOR: Quero consultar horas de partida
(2)  SISTEMA: Para onde deseja ir?
(3)  UTILIZADOR: Para o Porto
(4)  SISTEMA: Em que dia da semana deseja partir com destino a Porto?
(5)  UTILIZADOR: No Sábado
(6)  SISTEMA: Quer partir de que cidade, para chegar a Porto?
(7)  UTILIZADOR: De Lisboa
(8)  SISTEMA: Tem as seguintes horas de partida disponíveis: 08:30.
```

Este novo bloco de diálogo é um exemplo de um outro serviço que está disponível no dispositivo de venda de bilhetes. Neste caso, o utilizador pretende consultar as horas de partida de "sábado", com origem em "Lisboa" e destino "Porto". De acordo com a informação fornecida na base de dados, existe apenas uma hora de partida disponível.

No exemplo apresentado acima, a resposta do sistema faz menção à pluralidade do número de horas de partida, quando diz: "Tem as seguintes horas...". Esta frase incorre em erro quando existe apenas uma única hora disponível, como é o caso. Isto acontece porque o sistema tem uma única frase disponível para responder com as horas de partida, e esta frase está elaborada no plural (ver apêndice). Poderia-se arranjar uma frase em alternativa que não fizesse referência à cardinalidade da resposta, mas isso seria apenas adiar o problema. A solução ideal passa por gerar uma resposta diferente para um e outro caso, o que requer algo mais elaborado.

```
(1)  UTILIZADOR: Quero comprar um bilhete
(2)  SISTEMA: Quer partir de que cidade?
(3)  UTILIZADOR: Quero partir de Lisboa
(4)  SISTEMA: A que horas quer chegar, partindo de Lisboa?
(5)  UTILIZADOR: Quero chegar às 20:00
(6)  SISTEMA: Para onde deseja viajar, com partida às 20:00 e
              cidade de origem Lisboa?
(7)  UTILIZADOR: Desejo viajar para Lisboa
(8)  SISTEMA: A cidade de origem e a cidade de destino não podem ser
              iguais. Por favor, indique uma nova cidade de destino.
(9)  SISTEMA: Para onde deseja viajar, com partida às 20:00 e
              cidade de origem Lisboa?
(10) UTILIZADOR: Para Coimbra
(11) SISTEMA: Em que dia da semana deseja partir com destino a Coimbra?
(12) UTILIZADOR: Na Sexta-feira
(13) SISTEMA: Peço desculpa, mas não existe bilhete disponível a
```

```
              essa hora.
```

No ponto (3) e (7), o utilizador forneceu uma informação contraditória, já que só é possível efectuar viagens entre cidades. Como tal, não é possível que a cidade de origem e a cidade de destino sejam iguais ("Lisboa"). Imediatamente após a verificação da contradição, o sistema informa o utilizador da razão pela qual considerou haver uma contradição (ponto 8), e em (9) repete a pergunta para o utilizador cuja resposta esteve na base da contradição (ponto 9). O sistema procura resolver sempre primeiro contradições que existam no discurso, e só depois continua o diálogo.

```
(1)  UTILIZADOR: Quero comprar um bilhete de Lisboa para Lisboa
(2)  SISTEMA: A cidade de origem e a cidade de destino não podem
             ser iguais. Por favor, indique uma nova cidade de destino.
(3)  SISTEMA: Para onde deseja viajar, com partida de Lisboa?
(4)  UTILIZADOR: Para Coimbra
(5)  SISTEMA: Em que dia da semana deseja partir com destino a Coimbra?
(6)  UTILIZADOR: na Terça-feira
(7)  SISTEMA: A que horas quer partir com partida de Lisboa e
             destino a Coimbra?
(8)  UTILIZADOR: às 11:00
(9)  SISTEMA: A viagem com partida de Lisboa, e destino a Coimbra,
             pode ficar marcada para as 17:00, com chegada às 20:55.
```

Não é necessário que seja fornecida informação passo a passo para que possam ser detectadas contradições pelo sistema. Em (1), o utilizador fornece a contradição na mesma frase. A tentativa de resolução da contradição pode ser verificada em (2) e em (3).

4.2.3 Diálogos obtidos com ambos os domínios em paralelo

É possível realizar no sistema diálogos sobre mais do que um domínio ao mesmo tempo. Porque é realizado sobre dois contextos diferentes, o utilizador pode, por exemplo, comprar um bilhete de autocarros ao mesmo tempo que interage com os dispositivos da Domótica:

```
(1)  UTILIZADOR: Quero comprar um bilhete
(2)  SISTEMA: Quer partir de que cidade?
(3)  UTILIZADOR: Liga
(4)  SISTEMA: O que deseja ligar?
(5)  UTILIZADOR: Quero partir de Lisboa
(6)  SISTEMA: Qual é a hora de partida de Lisboa?
(7)  UTILIZADOR: A luz
(8)  SISTEMA: Que luz pretende ligar?
(9)  UTILIZADOR: Às 17:00
```

```
(10) SISTEMA: Em que dia da semana deseja partir de Lisboa?
(11) UTILIZADOR: Sábado
(12) SISTEMA: Para onde deseja viajar, no Sábado, com partida às 17:00 e
             cidade de origem Lisboa?
(13) UTILIZADOR: Luz da sala
(14) SISTEMA: A luz da sala foi ligada!
(15) UTILIZADOR: Pretendo viajar para Coimbra
(16) SISTEMA: A viagem com partida de Lisboa, e destino a Coimbra,
             pode ficar marcada para as 18:45, com chegada às 22:55.
```

Pela visualização do bloco de diálogo acima, pode-se constatar que é possível separar a informação contextual de dois domínios diferentes, e assim, efectuar mais do que uma acção em paralelo. No decorrer do diálogo é sempre mantida a informação que foi previamente fornecida em cada um dos domínios. A resposta do sistema é sempre realizada de acordo com a última interpretação efectuada, pelo que se uma afirmação do utilizador for efectuada no domínio dos Autocarros, o sistema responderá de seguida com algo respeitante ao mesmo domínio. Isso não obriga a que o utilizador responda a uma pergunta do sistema relacionada com um determinado domínio. Nos pares de pontos (4)-(5) e em (6)-(7), bem como noutros casos à frente, fica patente esse facto.

```
(1)  UTILIZADOR: Liga a aparelhagem
(2)  SISTEMA: A aparelhagem foi ligada!
(3)  UTILIZADOR: Aumenta o volume
(4)  SISTEMA: O volume foi aumentado!
(5)  UTILIZADOR: Quero comprar um bilhete
(6)  SISTEMA: A que horas quer partir?
(7)  UTILIZADOR: Desliga
(8)  SISTEMA: O que deseja desligar?
(9)  UTILIZADOR: às 11:00
(10) SISTEMA: Quer partir de que cidade, às 11:00?
(11) UTILIZADOR: A luz
(12) SISTEMA: A luz da sala foi desligada!
(13) UTILIZADOR: Quero partir de Lisboa
(14) SISTEMA: Para onde deseja viajar, com partida às 11:00 e
             cidade de origem Lisboa?
(15) UTILIZADOR: Lisboa
(16) SISTEMA: A cidade de origem e a cidade de destino não podem ser iguais.
             Por favor, indique uma nova cidade de destino.
(17) SISTEMA: Para onde deseja viajar, com partida às 23:00 e
             cidade de origem Lisboa?
(18) UTILIZADOR: Porto
(19) SISTEMA: Em que dia da semana deseja partir com destino a porto?
(20) UTILIZADOR: Sexta-feira
(21) SISTEMA: Peço desculpa, mas não existe bilhete disponível a essa hora.
(22) UTILIZADOR: Desliga
(23) SISTEMA: A aparelhagem foi desligada!
```

```
(24) UTILIZADOR: Desliga
(25) SISTEMA: A aparelhagem já se encontra desligada.
```

4.3 Avaliação

4.3.1 Características do sistema

O sistema de diálogos apresentado nesta dissertação é baseado em enquadramentos (ver secção 2.6.2). Como tal, beneficia das vantagens de um sistema deste tipo. Neste tipo de sistemas são feitas questões ao utilizador que permitem preencher espaços de informação. O fluxo não é determinado, tal como acontece nos sistemas baseados em estados finitos (ver secção 2.6.1). A acção tomada pelo sistema é baseada na interpretação actual da frase proferida pelo utilizador, pelo estado actual do diálogo e pelo conjunto de serviços disponíveis em cada momento, de acordo com os vários domínios activos no sistema. A separação da informação do domínio dos restantes módulos do sistema (ver secção 3.3) permite obter um sistema em que é possível inserir, remover, e actualizar informação sobre domínios mais facilmente.

O preenchimento múltiplo de espaços de informação permite ao sistema processar as informações extra dadas pelo utilizador, quer ao nível das respostas que ao nível das correcções. Deste modo, o tempo de transacção do diálogo pode ser reduzido, resultando num diálogo mais eficiente e natural. Neste tipo de sistemas, o utilizador não controla o diálogo. Na realidade, esta característica é resultado da própria estratégia de um sistema baseado em enquadramentos, já que o diálogo é à partida orientado para satisfazer o utilizador e não ambos os participantes, tal como acontece nos sistemas baseados em agentes (ver secção 2.6.3).

A informação do sistema que contribui para a determinação da próxima acção é ainda limitada. O sistema não providencia um leque alargado de respostas para diferentes conhecimentos de vários utilizadores, e à excepção dos serviços pertencentes aos vários domínios que possam estar activos no sistema, não se ajusta a outras possíveis configurações que possam surgir da mudança dinâmica do mundo. Também não envolve negociação, nem planeamento ou outro tipo de interacção colaborativa. O sistema usa confirmações e negações nos seus diálogos, mas ainda de forma limitada. É necessário enriquecer os actos de fala para que, por exemplo, o significado da frase “Quero ligar a luz” seja o mesmo de “Não, quero ligar a luz”.

O sistema de diálogo apresentado permite obter um diálogo em mais do que um domínio ao mesmo tempo. A independência contextual de cada um dos domínios presentes é uma das vantagens da estrutura usada na construção do sistema, mas o problema de selecção do domínio actual, apesar das abordagens referidas, continua a ser tema de investigação. A performance actual do sistema neste âmbito, face às abordagens seguidas para a sua resolução, depende na realidade do grau de

diferenças conceituais[1] entre os domínios.

4.3.2 Características do diálogo

Nesta secção são feitas considerações sobre cada uma das características de diálogo enunciadas na introdução desta dissertação (ver secção 1.3), com respeito aos diálogos obtidos no sistema.

- ***Turn Taking* e Iniciativa Mútua**: O *Turn Taking* efectuado no sistema de diálogos é realizado de forma muito simples. No início do diálogo, o utilizador despoleta uma sequência de acções, caracterizadas pelos constantes pedidos ao utilizador da informação necessária à execução de uma tarefa. O diálogo termina quando a tarefa é executada. O tempo e a duração com que cada participante actua não é considerada no sistema actual. O utilizador pode dar nas suas respostas mais informação daquela que é requisitada, mas o sistema constrói a resposta com vista à obtenção de apenas uma informação de cada vez;

- **Contexto do Diálogo e Elipses:** A formação da interpretação corrente envolve mapeamento das interpretações formadas até ao momento com obrigações do discurso no *Discourse Context* (ver secção 3.7). A resposta dada pelo sistema é seleccionada não só a partir das interpretações correntes, como também das contribuições dadas previamente pelo utilizador e armazenadas no *Discourse Context*;

- **Resolução de Referências:** No sistema de diálogos não é feita qualquer resolução de referências. Desta forma, nem o sistema utiliza referências nas respostas que dá, nem faz qualquer interpretação de referências utilizadas pelo utilizador nas frases que profere.

- **Pares adjacentes e inserções:** O utilizador pode corrigir informação que tenha fornecido previamente, apesar desta introdução de informação ser feita da mesma forma que para a informação original (ver figura 4.2). A disponibilidade de informação adicional para o utilizador requer a descrição dos serviços respectivos no domínio em causa;

```
(1)  UTILIZADOR: Quero ir para o Porto.
(2)  SISTEMA: Quer partir de onde para chegar a Porto?
(3)  UTILIZADOR: Quero um bilhete de Lisboa para Coimbra.
(4)  SISTEMA: A que horas quer partir com destino a Coimbra?
(5)  ...
```

Figura 4.2: Exemplo de diálogo produzido pelo sistema para os pares adjacentes e inserções.

- **Interpretação e Correcção:** Para que o participante de um diálogo tenha a certeza de que a sua contribuição é compreendida, ou que o próprio participante compreendeu uma

[1] Por diferenças conceituais entenda-se os termos, conceitos, e relações entre os conceitos dados em cada um dos domínios. A interpretação do sistema depende em larga escala deste conhecimento, o que torna extremamente relevante o seu papel na determinação do domínio actual. As ontologias poderiam aqui fornecer uma enorme contribuição.

contribuição prévia, o sistema faz uso de confirmação implícita nas respostas que dá. Sempre que possível, o sistema procura nas suas respostas fornecer ao utilizador o máximo de informação que já tiver sido providenciada pelo mesmo (ver figura 4.3). Esta funcionalidade é concretizada através da estratégia de selecção de respostas do *Surface Generation* (ver secção 3.9);

```
(1)  UTILIZADOR: Quero ir para Coimbra.
(2)  SISTEMA: Quer partir de onde para chegar a Coimbra?
(3)  UTILIZADOR: Quero partir de Lisboa.
(4)  SISTEMA: A que horas quer partir com partida de Lisboa e destino
             a Coimbra?
(5)  ...
```

Figura 4.3: Exemplo de diálogo produzido pelo sistema para a Interpretação e Correcção.

4.4 Sumário

Este capítulo apresentou exemplos de diálogos produzidos em dois domínios: (i) Domótica; e (ii) Autocarros. Para cada exemplo foram explicadas as principais intervenções do sistema no decorrer do diálogo. Foi também feita, na medida do possível, uma avaliação do sistema de diálogo. Foram descritas as características do sistema e dos diálogos produzidos.

Capítulo 5

Trabalho Futuro e Conclusões

Um gestor de diálogos é um componente fundamental de um sistema de diálogos. No decorrer desta dissertação, procurou-se sempre definir perfeitamente cada um dos módulos do gestor, para que cada um deles desempenhasse uma tarefa específica. Os domínios construídos para o sistema não resultaram apenas de ideias para demonstração, mas também de oportunidades aplicacionais concretas que surgiram ao longo do tempo em que o trabalho foi realizado.

Neste trabalho, ficam patentes duas experiências cuja avaliação se espera serem frutíferas no futuro: a primeira resulta da integração do gestor de diálogos com todo um sistema. Este sistema resulta da cooperação com outros colegas de trabalho do Laboratório. A segunda está relacionada com a introdução de informação linguística no sistema de diálogos, cuja experiência permitirá realçar as melhorias necessárias a cada um dos módulos envolvidos.

Na primeira secção deste capítulo é feito um resumo do trabalho elaborado nesta tese. Na secção seguinte é apresentada a situação actual do gestor de diálogos. A secção de trabalho futuro traça linhas orientadoras daquele que se considera ser o trabalho a desenvolver, para na secção seguinte elaborar-se um perfil do sistema de diálogos ideal. A última secção deste capítulo tece os comentários finais a este trabalho.

5.1 Contribuição

Enumeram-se de seguida as principais contribuições:

1. Introdução dos módulos prévios do gestor de diálogo na arquitectura *Galaxy*;
2. Introdução das obrigações de discurso (ver secção 3.6.2) no gestor de diálogo:
   - Reestruturação do *Gestor de interpretações* (ver secção 3.6):
     (a) Actualização do domínio;

(b) Criação das interpretações (com o auxílio do *Gestor do contexto*);

(c) Alteração das regras no *Gestor de tarefas*;

(d) Alteração do raciocínio sobre a informação do Acto de Resolução do Problema;

(e) Criação das obrigações de discurso.

- Construção do *Gestor do contexto* (ver seccção 3.7):
  (a) Mapeamento das obrigações de discurso com a frase dita pelo utilizador;
  (b) Criação do estado do diálogo.
- Construção do *Gestor de comportamentos* (ver secção 3.8):
  (a) Definição do módulo com base nas obrigações de discurso.
- Construção do *Gestor de geração* (ver secção 3.8):
  (a) Envio de informação ao *Gestor do contexto* sobre quais as obrigações de discurso seleccionadas.
- Construção do *Gerador de superfície* (ver secção 3.9):
  (a) Geração das frases com base na informação proveniente das obrigações de discurso.

3. Alteração da descrição de um domínio:

   (a) Separação da descrição de um domínio e de um dispositivo;

   (b) Introdução das regras de reconhecimento dos objectos;

   (c) Introdução das frases a serem geradas pelo sistema.

4. Reestruturação da comunicação com o *Gestor de serviços*:

   (a) Introdução dos pedido de enquadramentos e serviços ao *Gestor de serviços* (assegurar sincronização);

   (b) Introdução do pedido de reconhecimento dos objectos para o *Gestor de interpretaç ões*;

   (c) Introdução do pedido de frases para o *Gerador de superfície*.

5. Construção da representação do domínio da Domótica e do domínio dos Autocarros:

   (a) Aplicação do domínio da Domótica na casa do futuro.

6. Construção de um novo *Reconhecedor da linguagem*:

   (a) Integração do gestor de diálogos com a informação linguística providenciada pelos módulos: (i) Smorph; (ii) Pasmo; (iii) Susana; (iv) Ogre; (v) Algas; e (vi) AsdeCopas;

   (b) Integração do gestor de diálogos com a informação providenciada pelas heurísticas usadas na escolha da performativa dos actos de fala.

5.2 Situação Actual

Tal como foi descrito nesta dissertação, o gestor de diálogos tem definidas as suas fronteiras de acordo com aquilo que se considerou ser a melhor forma de estruturar os fluxos e as estruturas de informação. A divisão do gestor em módulos específicos ligados à interpretação, ao comportamento e à geração, garante uma divisão de funcionalidades à semelhança do que acontece com o cérebro humano (ver secção 5.4). O funcionamento do sistema baseia-se fundamentalmente no conceito de obrigações de discurso, onde se representam as intenções do sistema. Toda a informação relacionada com os domínios de aplicação dos diálogos são obtidas a partir do *Gestor de Serviços*. Tentou-se minimizar assim as dependências do sistema de diálogos para com os domínios, que podem, desta forma, ser introduzidos facilmente no sistema.

A informação de entrada no gestor de diálogos continua a ser deficitária. Os actos de fala são construídos idealmente a partir de informação linguística, mas para as demonstrações que actualmente efectuamos, esta informação ainda não é utilizada. Neste aspecto, o trabalho a realizar envolve a integração de outras ferramentas linguísticas já desenvolvidas no Laboratório. O módulo de comportamento é outro dos módulos que necessita de ser mais trabalhado. Para se possa usufruir das capacidades deste módulo, será útil usar o sistema com outros domínios para além dos descritos nesta dissertação.

5.3 Trabalho Futuro

É impossível descrever todo o trabalho que ainda há pela frente. Deste modo, fica aqui apenas um resumo do trabalho a realizar:

- **Construção de actos de fala:** todos os módulos de informação linguística, que incluem, entre outras, funcionalidades de análise morfológica, sintáctica e semântica, terão de ser combinados de forma a contribuirem para a produção dos actos de fala. O *Gestor de Interpretações* e o *Gestor do Contexto* devem ser adaptados às novas exigências.

- **Adaptação dos módulos de informação linguística para o tratamento de informação dinâmica:** porque a compreensão da intenção do utilizador está intrinsecamente relacionada com os domínios acoplados ao sistema, os módulos de informação linguística devem ser capazes de tratar as mudanças relacionadas com os domínios.

- **Desenvolvimento da informação contextual:** tal como a informação do domínio de aplicação, também a informação do contexto contribui para a descoberta da intenção do utilizador. É assim fundamental que se perceba que informação contextual contribui para desambiguar o resultado dos módulos de informação linguística. Esta informação contextual deverá ser actualizada para que possa contabilizar todas e quaisquer referências que sejam usadas nos diálogos, como os pronomes.

- **Desenvolvimento do raciocínio e de comportamento:** deverá ser introduzida no sistema capacidade de planeamento, o que permitirá obter outro tipo de diálogos. Associada a esta nova funcionalidade será necessariamente construído um novo domínio, para que se possam retirar os respectivos dividendos da nova funcionalidade.

5.4 O Sistema Ideal

O sistema de diálogos ideal será forçosamente aquele que tiver uma performance parecida com a dos humanos. Esse é o objectivo de todos os que trabalham nesta ou em áreas relacionadas. É com aquilo que os humanos produzem que se fazem os treinos, os testes, as comparações e as avaliações dos sistemas. Tal como é feito actualmente em ciências ligadas a áreas da neurologia, tenta-se também na área específica dos sistemas de diálogo, diferenciar tanto quanto possível, as diferentes áreas de actividade que possam estar envolvidas em todos os processos cognitivos. Procura-se então modelar computacionalmente as várias zonas do cérebro que se crêem serem responsáveis pelas mais variadas actividades. O cérebro racional (neopálio ou neocórtex) é uma de três unidades constitutivas do cérebro humano e é responsável pelas tarefas intelectuais que desempenhamos. Associada a esta unidade estarão zonas específicas de camadas celulares responsáveis pelo tratamento especializado ao nível sensorial, ao nível da interpretação, ao nível do raciocínio, ao nível do comportamento, e finalmente, ao nível dos actuadores. É nestas cinco camadas que se concentram actualmente as estruturas de grande parte dos sistemas de diálogo existentes. A estrutura do gestor desenvolvido no âmbito desta tese, à semelhança da estrutura também usada no sistema TRIPS (ver secção 2.7.2), procura separar essas mesmas camadas[1].

A camada interpretativa ideal destes sistemas tem a missão fundamental de interpretar toda e qualquer informação que lhe chegue a partir do nível sensorial. Para tal, esta camada lida não apenas com texto, mas também com outros tipos de informação, como as características da voz, os gestos, e as emoções do locutor. Toda esta actividade de interpretação implica necessariamente combinação com informação proveniente do contexto e com o historial do diálogo em que os participantes estão envolvidos. Na camada de raciocínio do sistema são tomadas todas as decisões. Para a decisão é tida em conta não apenas a interpretação actual, mas também as crenças actuais do sistema. As emoções têm também um papel fundamental nesta camada. Recentes investigações de António Damásio [2] comprovam que as emoções têm um papel indispensável na tomada de decisão. Ao nível do comportamento é decidido de que forma as decisões tomadas na camada de raciocínio vão ser exteriorizadas. Existe diferença entre muitas decisões que tomamos constantemente e a forma como elas são transmitidas para fora. As emoções têm também nesta camada um papel preponderante.

A complexidade inerente a este tipo de sistemas é enorme. Até hoje não se conseguiu perceber

[1] O nível dos sensores e dos actuadores não é considerado num gestor de diálogos.

[2] António Damásio é professor de Neurologia na Universidade do Iowa - USA, e é actualmente considerado um dos mais conceituados neurocientistas do mundo.

como funciona o cérebro humano, e creio que dificilmente se conseguirá alguma vez perceber como é que realmente ele funciona. O sonho de muitos investigadores nesta área consiste em modelar computacionalmente um sistema que tem a eficácia e a eficiência de um sistema com uma evolução de cerca de cinco mil milhões de anos. A minha opinião é de que o sistema de diálogos ideal não será obtido apenas com base em modelos computacionais, mas com uma sinergia entre os sistemas biológicos e computacionais. Ao longo da história da humanidade assistiram-se a revoluções de extrema importância, nomeadamente nas áreas quântica, informática e biológica, mas a investigação decorreu em larga escala isoladamente em cada uma delas. Na minha opinião, os futuros investigadores terão de aprender a lidar não apenas com uma destas áreas, mas com todas elas, de forma a poderem usufruir das vantagens de cada uma.

5.5 Observações Finais

Porque o gestor de diálogos é um dos primeiros a ser desenvolvido no Laboratório, creio que nesta tese fica acima de tudo a experiência tida na construção do mesmo. A complexidade inerente a este tipo de sistemas, bem como a integração do mesmo com outros módulos previamente construídos tornaram difícil a tarefa. Para concretizar a mesma, foi fundamental a contribuição de todos os que estiveram envolvidos no desenvolvimento do sistema de diálogos. Pelo que se disse na secção anterior, é ilógico afirmar que um sistema de diálogos esteja alguma vez terminado, a menos que se considerem aplicações muito específicas. Existirá sempre muito trabalho por realizar nesta área, mas espera-se que o resultado desta tese sirva de base para uma continuação do trabalho, e que de alguma forma tenha contribuído para aumentar o conhecimento actual no laboratório acerca dos gestores e dos sistemas de diálogo.

Bibliografia

[1] Audimus, Documentação da Audimus API, L^2F, INESC-ID.

[2] Allen J., Ferguson G., Stent A., *An Architecture for more realistic conversational systems*, University of Rochester, 2001.

[3] Woodruff A., Aoki P. M., *An Introduction to Conversational Analysis*, Palo Alto Research Center, 2000.

[4] Allen, J., *Natural Language Understanding*, Benjamin/Cummings, Menlo Park CA, second edition, 1995.

[5] Picard R., *Affective Computing*, MIT Press, 1997.

[6] Allen J., Byron D., Dzikovska M., Ferguson G., Galescu L., Stent A., *Towards Conversational Human-Computer Interaction*, Department of Computer Science, University of Rochester, AI Magazine, 2001.

[7] Graesser A. C., Lehn K. V., Rose C. P., Jordan P. W., Harter D., *Intelligent tutoring systems with conversational dialogues*, AI Magazine, Winter, 2001.

[8] Hill I., *Natural language versus computer language*, In M. Sime and M. Coombs (Eds.) Designing for Human-Computer Communication. Academic Press, 1983.

[9] Mayhew D., *Principles and Guidelines in Software User Interface Design*, Prentice-Hall, 1992.

[10] Shneiderman B., *Designing the User Interface: Strategies for Effective Human-Computer Interaction (Second Edition)*, Addison-Wesley, 1992.

[11] Véronis J., *Error in natural language dialogue between man and machine*, International Journal of Man-Machine Studies 35, 187-217, 1991.

[12] Austin J. A., *How to Do Things with Words*, Harvard University Press, 1996.

[13] Searle J. R., *A taxonomy of illocutionary acts*, Language, Mind and Knowledge, Minnesota Studies in the Philosophy of Science, pages 344-369, 1975.

[14] Kreutel J., Matheson C., *Modelling Questions and Assertions in Dialogue Using Obligations*, In Proceedings of Amselogue 99, the 3 Workshop on the Semantics and Pragmatics of Dialogue. University of Amsterdam, 1999.

[15] Traum D. R., Allen J., *Discourse Obligations in Dialogue Processing*, In Proceedings of the 32 Annual Meeting of the Association for Computational Linguistics, pages 1-8, 1994.

[16] Jameson A., Weis T., *How to juggle Discourse Obligations*, In R. Meyer-Klabunde & C. von Stutterheim (Eds.), Proceedings des Symposiums "Konzeptuelles und semantisches Wissen in der Sprachproduktion" [Proceedings of the symposium "Conceptual and Semantic Knowledge in Language Production"] (pp. 171-185). Heidelberg/Mannhei Germany: Reports from the Collaborative Research Center 245, "Language and Situation", 1996.

[17] Tsovaltzi D., Walter S., Burchardt A., *Dialogue*, MiLCA, Computerlinguistik, Universität des Saarlandes, Saarbrücken, Germany, Dezember 2003.

[18] Lochbaum G., Grosz, B.J. and Sidner, C.L., *Models of plans to support communication: An initial report*, In Proc. AAAI-90, pp. 485-490, 1990.

[19] Allen J., Ferguson G., Miller B. W., Ringger E. K., and Zollo T. S., *Dialogue systems: From theory to practice in TRAINS-96*, In Robert Dale, Hermann Moisl, and Harold Somers, eds., Handbook of Natural Language Processing, Marcel Dekker, New York, pp. 347-376, July 2000.

[20] Rich C., Sidner C., Lesh N., *COLLAGEN: Applying Collaborative Discourse Theory to Human-Computer Interaction*, Mitsubishi Electric Research Laboratories, November 2000.

[21] Seneff S., Polifroni J., *Dialogue Management in the Mercury Flight Reservation System*, Spoken Language Systems Group, Laboratory for Computer Science, Massachusetts Institute of Technology, 2000.

[22] Seneff S., Polifroni J., *Formal and Natural Language Generation in the mercury conversational system*, Spoken Language Systems Group, Laboratory for Computer Science, Massachusetts Institute of Technology, 2000.

[23] Rudnicky A.I., Thayer E., Constantinides P., Tchou C., Shern R., Lenzo K., Xu W., Oh A., *Creating Natural Dialogs in the Carnegie Mellon Communicator system*, Carnegie Mellon University, 2000.

[24] Rudnicky A.I., Bennet C., Black A., Chotomongcol A., Lenzo K., Oh A., Singk R., *Task and Domain Specific Modelling in the Carnegie Mellon Communicator System*, School of Computer Science, Carnegie Mellon University, 2000.

[25] http://www.scism.sbu.ac.uk/inmandw/tutorials/nlp/pragmatics/pragmatics.html.

[26] http://www.arts.monash.edu.au/ling/`speech_acts_allan.shtml`.

[27] http://www.sls.lcs.mit.edu/sls/whatwedo/applications/mercury.html.

[28] http://www.cs.rochester.edu/research/cisd/projects/trips.

[29] http://www.speech.cs.cmu.edu/Communicator.

[30] http://www.merl.com/projects/collagen.

[31] http://communicator.sourceforge.net.

[32] http://www.ach.org.

Apêndice A

Domínio da Domótica

A.1 Descrição do Domínio

```
<?xml version="1.0" encoding="UTF-8"?>
  <domain name="Domotica">
  <!-- Descrição do dominio -->
  <description>Esta e a descricao do dominio da domotica</description>

  <frame frameType="static">
    <description>
      <frame_name>Domotica</frame_name>
      <slot type="string" required="true" KEY="1">Action</slot>
      <slot type="string" key="2">Target</slot>
      <slot type="string" key="3">Identification</slot>
      <slot type="string" key="4">Division</slot>
      <slot type="string" key="5">Direction</slot>
      <slot type="string" key="6">State</slot>
     </description>
  </frame>

  <grammar>

  <composedrules>

   <rule id="1" uttgenerationtype="requestslot" uttgenerationslot="action">
    <one-of>
      <item>O que deseja fazer?</item>
      <item>Pode: [$values]. O que deseja fazer?</item>
      <item><Pode: [$values]. O que deseja fazer com [$#o $target] [$target]?</item>
      <item><Pode: [$values]. O que deseja fazer com [$#o $target] [$target]
            [$#do $identification] [$identification]?</item>
    </one-of>
   </rule>

   <rule id="2" uttgenerationtype="requestslot" uttgenerationslot="target">
    <one-of>
      <item>Em que dispositivo pretende realizar a sua acção?</item>
      <item>O que deseja [$action]?</item>
      <item>O que deseja [$action] [$#no $identification] [$identification]?</item>
```

```
 </one-of>
</rule>

<rule id="3" uttgenerationtype="requestslot" uttgenerationslot="identification">
 <one-of>
   <item>Onde pretende actuar?</item>
   <item>O que pretende [$action]?</item>
   <item>O que pretende [$action]? [$values]?</item>
   <item>Que [$target] pretende [$action]?</item>
   <item>Que [$target] pretende [$action], [$values]? </item>
   <item>O que pretende [$action]? [$#o $target] [$target] da [$values]?</item>
 </one-of>
</rule>

<rule id="4" uttgenerationtype="requestslot" uttgenerationslot="division">
 <one-of>
   <item>Em que divisão pretende actuar?</item>
   <item>Em que divisão pretende [$action]?</item>
   <item>Em que divisão [$#do $identification] [$identification] pretende [$action]?</item>
   <item>Em que divisão [$#do $identification] [$identification] pretende [$action]
         [$#o $target] [$target]?</item>
 </one-of>
</rule>

<rule id="5" uttgenerationtype="requestslot" uttgenerationslot="direction">
 <one-of>
   <item>Qual é a direcção?</item>
   <item>Qual é a direcção em que pretende [$action]?</item>
   <item>Em que direcção pretende [$action] [$#o $target] [$target]?</item>
 </one-of>
</rule>

<rule id="6" uttgenerationtype="requestslot" uttgenerationslot="state">
 <one-of>
   <item>Qual é o novo estado?</item>
   <item>Pretende [$action] [$#o $target] para que estado?</item>
   <item>Para que estado pretende [$#o $target]?</item>
 </one-of>
</rule>

<rule id="7" uttgenerationtype="showresults" uttgenerationresult="success">
 <one-of>
   <item>O serviço foi executado com sucesso!</item>
   <item>A sua ordem foi cumprida!</item>
   <item>[$#o $target] [$target] [$#ir $target] [$action $target pp]!</item>
   <item>[$action 1p_pi] [$#o $target] [$target] com sucesso!</item>
   <item>[$action 1p_pi] [$#o $target] [$target] [$#do $identification]
         [$identification] com sucesso!</item>
   <item>[$#o $target] [$target] [$#do $identification] [$identification] foi
         [$action $target pp]!</item>
   <item>[$#o $target] [$target] [$#ir $target] [$action pp $target] para
         [$#o $direction] [$direction]!</item>
 </one-of>
</rule>

<rule id="8" uttgenerationtype="showresults" uttgenerationresult="failure">
 <one-of>
   <item>Aconteceu um erro na execução do serviço.</item>
   <item>Aconteceu um erro e não podemos [$action] o que pretende.</item>
```

```
    <item>Aconteceu um erro e não podemos [$action] [$#o $target] [$target] que pretende.</item>
    <item>Aconteceu um erro e não podemos [$action] [$#o $target] [$target] [$#do $identification]
          [$identification] que pretende.</item>
  </one-of>
 </rule>

 <rule id="9" uttgenerationtype="clarify" uttgenerationslot="">
  <one-of>
    <item>Tenha atenção com aquilo que pede.</item>
    <item>Não consigo [$action] o que pretende.</item>
    <item>Uma acção do tipo [$action] não pode ser realizada sobre [$#o $target]
          [$target].</item>
    <item>Uma acção do tipo [$action] não pode ser realizada sobre [$#o $target] [$target]
          [$#do $identification] [$identification].</item>
    <item>Uma acção do tipo [$action] não pode ser realizada sobre [$#o $target] [$target]
          [$#do $identification] [$identification] [$#do $division] [$division].</item>
  </one-of>
 </rule>

 <rule id="10" uttgenerationtype="validatestate" uttgenerationslot="">
  <one-of>
    <item>O estado actual não permite executar o que pretende.</item>
    <item>O estado actual não permite realizar a acção [$action].</item>
    <item>Não consigo [$action] o que quer que seja no estado actual.</item>
    <item>[$#o $target] [$target] [$#do $identification] [$identification] [$#do $division]
          [$division] já se [$#encontra $target] [$state $target].</item>
    <item>[$#o $target] [$target] já se [$#encontra $target] [$action $target].</item>
    <item>[$#o $target] [$target] [$#do $identification] [$identification] já se
          [$#encontra $target] [$action $target].</item>
    <item>[$#o $target] [$target] [$#do $identification] [$identification] [$#do $division]
          [$division] já se [$#encontra $target] [$action $target].</item>
  </one-of>
 </rule>

<rule id="11" uttgenerationtype="confirm" uttgenerationresult="">
  <one-of>
    <item>Tem a certeza?</item>
    <!-- Algumas frases próprias para os estores -->
    <item>Quer [$action] [$#o $target] [$target] para [$#o $direction] [$direction]?</item>
    <item>Quer [$action] [$#o $target] [$target] [$#do $identification] [$identification]
          para [$#o $direction] [$direction]?</item>
    <item>Quer [$action] [$#o $target] [$target] [$#do $identification] [$identification]
          [$#do $division] [$division] para [$#o $direction] [$direction]?</item>
    <!-- Para todas as restantes frases -->
    <item>Quer [$action] [$#o $target] [$target] ?</item>
    <item>Quer [$action] [$#o $target] [$target] [$#do $identification] [$identification]?</item>
    <item>Quer [$action] [$#o $target] [$target] [$#do $identification] [$identification]
          [$#do $division] [$division]?</item>
  </one-of>
 </rule>

 <rule id="12" uttgenerationtype="continuedialogue" uttgenerationresult="">
  <one-of>
    <item>O que deseja então fazer?</item>
  </one-of>
 </rule>

 <rule id="13" uttgenerationtype="informaboutactionexecution" uttgenerationresult="">
  <one-of>
```

```
      <item>O seu pedido está a ser executado. Aguarde um instante !</item>
    </one-of>
   </rule>

  </composedrules>

 </grammar>

</domain>
```

A.2 Descrição dos Dispositivos

A.2.1 Luz da sala

```
<?xml version="1.0" encoding="UTF-8" ?>
<device name="Luz da Sala" domain="Domotica">
  <!-- Descrição das funcionalidades do dispositivo -->
  <description>Este e o controlador da luz da sala de
               demonstracoes</description>

  <state name="desligada">

    <service name="ligar_luz">
      <label>Ligar a luz da sala</label>
      <params>
        <param slotkey="1" value="ligar"></param>
        <param slotkey="2" value="luz"></param>
        <param slotkey="3" value="sala"></param>
        <param slotkey="4" value="quarto"></param>
      </params>
      <execute>
        <name>liga</name>
        <success>ligada</success>
        <failure/>
      </execute>
    </service>

  </state>

  <state name="ligada">

    <service name="desligar_luz">
      <label>Desligar a luz da sala</label>
      <params>
        <param slotkey="1" value="desligar"></param>
        <param slotkey="2" value="luz"></param>
        <param slotkey="3" value="sala"></param>
        <param slotkey="4" value="quarto"></param>
      </params>
      <execute>
        <name>desliga</name>
        <success>desligada</success>
        <failure/>
```

```
      </execute>
    </service>

  </state>

  <grammar>

   <simplerules>

    <rule id="action">
      <one-of>
        <item>ligar</item>
        <item>desligar</item>
      </one-of>
    </rule>

    <rule id="target">
      <one-of>
        <item>luz</item>
      </one-of>
    </rule>

    <rule id="identification">
      <one-of>
        <item>sala</item>
      </one-of>
    </rule>

    <rule id="division">
      <one-of>
        <item>quarto</item>
      </one-of>
    </rule>

  </simplerules>

 </grammar>

</device>
```

A.2.2 Luz da mesa

```
<?xml version="1.0" encoding="UTF-8"?>
  <device name="Luz da Mesa" domain="Domotica">
  <!-- Descrição das funcionalidades do dispositivo -->
  <description>Este e o controlador da luz da mesa da
               sala de demonstracoes</description>

  <state name="desligada">

    <service name="ligar luz">
      <label>Ligar a luz da mesa</label>
      <params>
        <param slotkey="1" value="ligar"></param>
        <param slotkey="2" value="luz"></param>
```

```
        <param slotkey="3" value="mesa"></param>
        <param slotkey="4" value="quarto"></param>
      </params>
      <execute>
        <name>liga</name>
        <success>ligada</success>
        <failure/>
      </execute>\label{fig:descricaodispositivoluzsala}
    </service>

</state>

<state name="ligada">

    <service name="desligar luz">
      <label>Desligar a luz da mesa</label>
      <params>
        <param slotkey="1" value="desligar"></param>
        <param slotkey="2" value="luz"></param>
        <param slotkey="3" value="mesa"></param>
        <param slotkey="4" value="quarto"></param>
      </params>
      <execute>
        <name>desliga</name>
        <success>desligada</success>
        <failure/>
      </execute>
    </service>

    <service name="aumentar luz">
      <label>Aumentar a intensidade da luz da mesa</label>
      <params>
        <param slotkey="1" value="aumentar"></param>
        <param slotkey="2" value="luz"></param>
        <param slotkey="3" value="mesa"></param>
        <param slotkey="4" value="quarto"></param>
      </params>
      <execute>
        <name>dimUp</name>
        <success/>
        <failure/>
      </execute>
    </service>

    <service name="diminuir luz">
      <label>Diminuir a intensidade da luz da mesa</label>
      <params>
        <param slotkey="1" value="diminuir"></param>
        <param slotkey="2" value="luz"></param>
        <param slotkey="3" value="mesa"></param>
        <param slotkey="4" value="quarto"></param>
      </params>
      <execute>
        <name>dimDown</name>
        <success/>
        <failure/>
      </execute>
    </service>
```

```
  </state>

 <grammar>

   <simplerules>

    <rule id="action">
      <one-of>
        <item>ligar</item>
        <item>desligar</item>
        <item>aumentar</item>
         <item>diminuir</item>
      </one-of>
    </rule>

    <rule id="target">
      <one-of>
        <item>luz</item>
      </one-of>
    </rule>

    <rule id="identification">
      <one-of>
        <item>mesa</item>
      </one-of>
    </rule>

    <rule id="division">
      <one-of>
        <item>quarto</item>
      </one-of>
    </rule>

  </simplerules>

 </grammar>

</device>
```

A.2.3 Ar condicionado

```
<?xml version="1.0" encoding="UTF-8" ?>
<device name="Controlador de Ar Condicionado" domain="Domotica">
  <!-- Descrição das funcionalidades do dispositivo -->
  <description>Este e controlador do sistema de ar condicionado</description>

  <state name="desligado">

    <service name="ligar arcondicionado">
      <label>Ligar o Ar Condicionado</label>
      <params>
        <param slotkey="1" value="ligar"></param>
        <param slotkey="2" value="arcondicionado"></param>
      </params>
      <execute>
```

```
        <name>liga</name>
        <success>ligado</success>
        <failure/>
      </execute>
    </service>

</state>

<state name="ligado">

    <service name="desligar arcondicionado">
      <label>Desligar o Ar Condicionado</label>
      <params>
        <param slotkey="1" value="desligar"></param>
        <param slotkey="2" value="arcondicionado"></param>
      </params>
      <execute>
        <name>desliga</name>
        <success>desligado</success>
        <failure/>
      </execute>
    </service>

    <service name="aumentar temperatura">
      <label>Aumentar a temperatura</label>
      <params>
        <param slotkey="1" value="aumentar"></param>
        <param slotkey="2" value="temperatura"></param>
      </params>
      <execute>
        <name>aumenta</name>
        <success/>
        <failure/>
      </execute>
    </service>

    <service name="diminuir temperatura">
      <label>Diminuir a temperatura</label>
      <params>
        <param slotkey="1" value="diminuir"></param>
        <param slotkey="2" value="temperatura"></param>
      </params>
      <execute>
        <name>diminui</name>
        <success/>
        <failure/>
      </execute>
    </service>

    <service name="alterar direccao">
      <label>Alterar a direcção do ar</label>
      <params>
        <param slotkey="1" value="alterar"></param>
        <param slotkey="2" value="direccao"></param>
      </params>
      <execute>
        <name>sweepOn</name>
        <success/>
        <failure/>
```

```
      </execute>
    </service>

    <service name="parar direccao">
      <label>Parar alteração da direcção do ar condicionado</label>
      <params>
        <param slotkey="1" value="parar"></param>
        <param slotkey="2" value="direccao"></param>
      </params>
      <execute>
        <name>sweepOff</name>
        <success/>
        <failure/>
      </execute>
    </service>

  </state>

  <grammar>

   <simplerules>

    <rule id="action">
      <one-of>
        <item>ligar</item>
         <item>desligar</item>
         <item>aumentar</item>
        <item>diminuir</item>
        <item>alterar</item>
        <item>parar</item>
      </one-of>
    </rule>

    <rule id="target">
      <one-of>
        <item>arcondicionado</item>
        <item>temperatura</item>
        <item>direccao</item>
      </one-of>
    </rule>

  </simplerules>

 </grammar>

</device>
```

A.2.4 Estores

```
<?xml version="1.0" encoding="UTF-8" ?>
  <device name="Controlador de Estores" domain="Domotica">
  <!-- Descrição das funcionalidades do dispositivo -->
  <description>Este e o controlador dos estores</description>

  <state name="fechado">
```

```
  <service name="abrir estores">
    <label>Abrir os estores</label>
    <params>
      <param slotkey="1" value="abrir"></param>
      <param slotkey="2" value="estores"></param>
    </params>
    <execute>
      <name>abrir</name>
      <success>aberto</success>
      <failure/>
    </execute>
  </service>

</state>

<state name="aberto">

  <service name="fechar estores">
    <label>Fechar os estores</label>
    <params>
      <param slotkey="1" value="fechar"></param>
      <param slotkey="2" value="estores"></param>
    </params>
    <execute>
      <name>fechar</name>
      <success>fechado</success>
      <failure/>
    </execute>
  </service>

  <service name="rodar os estores para a esquerda ou para a direita">
    <label>Rodar a esquerda ou direita</label>
    <params>
      <param slotkey="1" value="rodar"></param>
      <param slotkey="2" value="estores"></param>
       <param slotkey="5"></param>
    </params>
    <execute>
      <name>roda</name>
      <success/>
      <failure/>
    </execute>
  </service>

</state>

<grammar>

 <simplerules>

  <rule id="action">
    <one-of>
      <item>abrir</item>
       <item>fechar</item>
      <item>rodar</item>
    </one-of>
  </rule>
```

```
    <rule id="target">
      <one-of>
        <item>estores</item>
      </one-of>
    </rule>

    <rule id="direction">
      <one-of>
        <item>esquerda</item>
         <item>direita</item>
      </one-of>
    </rule>

  </simplerules>

 </grammar>

</device>
```

A.2.5 Aparelhagem

```
<?xml version="1.0" encoding="UTF-8"?>
<device name="Controlador de Aparelhagem" domain = "Domotica">
  <!-- Descrição das funcionalidades do dispositivo -->
  <description>Este e controlador do sistema HIFI</description>

  <state name="desligado">

    <!-- Ligar a aparelhagem -->
    <service name="ligar aparelhagem">
      <label>Ligar a aparelhagem</label>
      <params>
        <param slotkey="1" value="ligar"></param>
        <param slotkey="2" value="aparelhagem"></param>
      </params>
     <execute>
        <name>liga</name>
        <success>ligado</success>
        <failure/>
      </execute>
    </service>

  </state>

  <state name="ligado">

    <!-- Servico de Desligar a Aparelhagem -->
    <service name="desligar aparelhagem">
      <label>Desligar a aparelhagem</label>
      <params>
        <param slotkey="1" value="desligar"></param>
        <param slotkey="2" value="aparelhagem"></param>
      </params>
      <execute>
        <name>desliga</name>
```

```
    <success>desligado</success>
    <failure/>
  </execute>
</service>

<!-- Ligar o Radio -->
<service name="ligar radio">
  <label>Ligar o Radio</label>
  <params>
    <param slotkey="1" value="ligar"></param>
    <param slotkey="2" value="radio"></param>
  </params>
  <execute>
    <name>estacaoRadio</name>
    <success/>
    <failure/>
  </execute>
</service>

<!-- Aumentar o volume da aparelhagem -->
<service name="aumentar volume">
  <label>Aumentar volume</label>
  <params>
    <param slotkey="1" value="aumentar"></param>
    <param slotkey="2" value="volume"></param>
  </params>
  <execute>
    <name>volumeUp</name>
    <success/>
    <failure/>
  </execute>
</service>

<!-- Diminuir o volume da aparelhagem -->
<service name="diminuir volume">
  <label>Diminuir volume</label>
  <params>
    <param slotkey="1" value="diminuir"></param>
    <param slotkey="2" value="volume"></param>
  </params>
  <execute>
        <name>volumeDown</name>
        <success/>
        <failure/>
    </execute>
</service>

<!-- Tocar o CD -->
<service name="tocar cd">
  <label>Tocar CD</label>
  <params>
    <param slotkey="1" value="tocar"></param>
    <param slotkey="2" value="cd"></param>
  </params>
    <execute>
        <name>tocaCd</name>
        <sucess/>
        <failure/>
    </execute>
```

```
</service>

<!-- Tocar o proximo CD -->
<service name="proximo cd">
  <label>Proximo CD</label>
  <params>
    <param slotkey="1" value="proximo"></param>
    <param slotkey="2" value="cd"></param>
  </params>
    <execute>
        <name>proximoCd</name>
        <sucess/>
        <failure/>
    </execute>
</service>

<!-- Proxima musica -->
<service name="avancar musica">
  <label>Proxima Musica</label>
  <params>
    <param slotkey="1" value="avancar"></param>
    <param slotkey="2" value="musica"></param>
  </params>
    <execute>
        <name>faixaSeguinte</name>
        <success/>
        <failure/>
    </execute>
</service>

<!-- Musica anterior -->
<service name="recuar musica">
  <label>Musica Anterior</label>
  <params>
    <param slotkey="1" value="recuar"></param>
    <param slotkey="2" value="musica"></param>
  </params>
    <execute>
        <name>faixaAnterior</name>
        <success/>
        <failure/>
    </execute>
</service>

<!-- Tirar o som da aparelhagem -->
<service name="tirar som">
  <label>Tirar Som</label>
  <params>
    <param slotkey="1" value="tirar"></param>
    <param slotkey="2" value="som"></param>
  </params>
    <execute>
        <name>calar</name>
        <success/>
        <failure/>
    </execute>
</service>

<!-- Ouvir o som da aparelhagem -->
```

```
  <service name="ouvir som">
    <label>Ouvir Som</label>
    <params>
      <param slotkey="1" value="ouvir"></param>
      <param slotkey="2" value="som"></param>
    </params>
      <execute>
          <name>ouvir</name>
          <success/>
          <failure/>
      </execute>
  </service>

  <!-- Parar o CD -->
  <service name="parar cd">
    <label>Parar CD</label>
    <params>
      <param slotkey="1" value="parar"></param>
      <param slotkey="2" value="cd"></param>
    </params>
      <execute>
          <name>pararCd</name>
          <success/>
          <failure/>
      </execute>
  </service>

  <!-- Ouvir cassete -->
  <service name="ouvir cassete">
    <label>Ouvir Cassete</label>
    <params>
      <param slotkey="1" value="ouvir"></param>
      <param slotkey="2" value="cassete"></param>
    </params>
      <execute>
          <name>tocaCassete</name>
          <success/>
          <failure/>
      </execute>
  </service>

</state>

<grammar>

 <simplerules>

  <rule id="action">
    <one-of>
       <item>ligar</item>
       <item>desligar</item>
       <item>aumentar</item>
       <item>diminuir</item>
       <item>tocar</item>
       <item>proximo</item>
       <item>avancar</item>
       <item>recuar</item>
       <item>tirar</item>
       <item>ouvir</item>
```

```
        <item>parar</item>
        </one-of>
    </rule>

    <rule id="target">
      <one-of>
        <item>aparelhagem</item>
        <item>radio</item>
        <item>volume</item>
        <item>cd</item>
        <item>musica</item>
        <item>som</item>
        <item>cassete</item>
      </one-of>
    </rule>

  </simplerules>

 </grammar>

</device>
```

Apêndice B

Domínio dos Autocarros

B.1 Descrição do Domínio

```
<?xml version="1.0" encoding="UTF-8"?>
  <domain name="Autocarros">
  <!-- Descrição do dominio -->
  <description>Esta e a descricao do dominio dos Autocarros</description>

  <frame frameType="static">
    <description>
             <frame_name>Autocarros</frame_name>
             <slot type="string" key="1">Action</slot>
             <slot type="string" key="2">Target</slot>
             <slot type="string" key="3">DiaSemana</slot>
             <slot type="string" key="4">CidadeOrigem</slot>
             <slot type="string" key="5">CidadeDestino</slot>
             <slot type="string" key="6">HoraPartida</slot>
             <slot type="string" key="7">HoraChegada</slot>
    </description>
  </frame>

 <grammar>

 <composedrules>

   <rule id="1" uttgenerationtype="requestslot" uttgenerationslot="action">
    <one-of>
      <item>O que deseja fazer?</item>
      <item>Pode [$values]. O que deseja fazer?</item>
      <item>Pode [$values]. O que deseja fazer com o [$target]?</item>
      <item>Pode [$values]. O que deseja fazer com o [$target] de [$cidadeorigem]?</item>
      <item>Pode [$values]. O que deseja fazer com o [$target] de [$cidadedestino]?</item>
      <item>Pode [$values]. O que deseja fazer com o [$target] de [$cidadeorigem],
            para [$cidadedestino]?</item>
      <item>Pode [$values]. O que deseja fazer com o [$target] de [$cidadeorigem],
            para [$cidadedestino], às [$horapartida]?</item>
      <item>Pode [$values]. O que deseja fazer com o [$target] de [$cidadeorigem],
            para [$cidadedestino], que chega às [$horachegada]?</item>
```

```
    <item>Pode [$values]. O que deseja fazer com o [$target] de [$cidadeorigem],
          para [$cidadedestino], que chega às [$horachegada], [$#no $diasemana]
          [$diasemana]?</item>
  </one-of>
</rule>

<rule id="2" uttgenerationtype="requestslot" uttgenerationslot="target">
  <one-of>
    <item>O que pretende fazer?</item>
    <item>O que deseja [$action]?</item>
    <item>O que pretende [$action]?</item>
  </one-of>
</rule>

 <rule id="3" uttgenerationtype="requestslot" uttgenerationslot="cidadeorigem">
 <one-of>
    <item>De onde quer partir? </item>
    <item>Qual a cidade de origem? </item>
    <item>De onde deseja partir? </item>
    <item>Quer partir de que cidade? </item>
    <item>Quer partir de que cidade, para chegar a [$cidadedestino]? </item>
    <item>Quer partir de que cidade, às [$horapartida]? </item>
    <item>Quer partir de que cidade, para chegar às [$horachegada]? </item>
    <item>Quer partir de que cidade, para chegar a [$cidadedestino],
          às [$horachegada]? </item>
    <item>Quer partir de que cidade, [$#no $diasemana] [$diasemana],
          para chegar a [$cidadedestino], às [$horachegada]? </item>
  </one-of>
</rule>

 <rule id="4" uttgenerationtype="requestslot" uttgenerationslot="cidadedestino">
 <one-of>
    <item>Para onde quer ir? </item>
    <item>Qual é a cidade de destino? </item>
    <item>Qual é o teu destino? </item>
    <item>Para onde quer viajar? </item>
    <item>Para onde deseja ir? </item>
    <item>Para onde deseja viajar, às [$horapartida]? </item>
    <item>Para onde deseja viajar, para chegar às [$horachegada]? </item>
    <item>Para onde deseja viajar, com partida de [$cidadeorigem]? </item>
    <item>Para onde deseja viajar, com hora de chegada às [$horachegada]? </item>
    <item>Para onde deseja viajar, com partida às [$horapartida] e cidade de origem
          [$cidadeorigem]?</item>
    <item>Para onde deseja viajar, [$#no $diasemana] [$diasemana], com partida
          às [$horapartida] e cidade de origem [$cidadeorigem]?</item>
  </one-of>
</rule>

 <rule id="5" uttgenerationtype="requestslot" uttgenerationslot="horapartida">
 <one-of>
    <item>Qual é a hora de partida? </item>
    <item>A que horas quer partir? </item>
    <item>Qual é a hora de partida de [$cidadeorigem]? </item>
    <item>A que horas quer partir com destino a [$cidadedestino]? </item>
    <item>A que horas quer partir com partida de [$cidadeorigem] e destino
          a [$cidadedestino]? </item>
    <item>A que horas quer partir [$#no $diasemana] [$diasemana], para chegar
          às [$horachegada] a [$cidadedestino]? </item>
  </one-of>
```

```
</rule>

<rule id="6" uttgenerationtype="requestslot" uttgenerationslot="horachegada">
 <one-of>
   <item>Qual é a hora de chegada? </item>
   <item>A que horas quer chegar? </item>
   <item>A que horas quer chegar, partindo de [$cidadeorigem]? </item>
   <item>Qual é a hora de chegada a [$cidadedestino]? </item>
   <item>A que horas quer chegar, partindo de [$cidadeorigem] e chegando
         a [$cidadedestino]? </item>
   <item>A que horas quer chegar [$#no $diasemana] [$diasemana], partindo
         às [$horapartida] de [$cidadeorigem]? </item>
 </one-of>
</rule>

<rule id="7" uttgenerationtype="requestslot" uttgenerationslot="diasemana">
 <one-of>
   <item>Em que dia da semana deseja partir? </item>
   <item>Em que dia da semana deseja partir de [$cidadeorigem]? </item>
   <item>Em que dia da semana deseja partir com destino a [$cidadedestino]? </item>
 </one-of>
</rule>

<rule id="8" uttgenerationtype="slotdesambiguation" uttgenerationdesambiguation="cidadeorigem">
  <one-of>
    <item>[$name] é cidade de origem ou é cidade de destino ?</item>
  </one-of>
</rule>

 <rule id="9" uttgenerationtype="slotdesambiguation" uttgenerationdesambiguation="cidadedestino">
  <one-of>
    <item>[$name] é cidade de origem ou é cidade de destino ?</item>
  </one-of>
</rule>

<rule id="11" uttgenerationtype="slotdesambiguation" uttgenerationdesambiguation="horapartida">
  <one-of>
    <item>[$name] é hora de partida ou é hora de chegada ?</item>
  </one-of>
</rule>

 <rule id="12" uttgenerationtype="slotdesambiguation" uttgenerationdesambiguation="horachegada">
  <one-of>
  <item>[$name] é hora de partida ou é hora de chegada ?</item>
  </one-of>
</rule>

<rule id="13" uttgenerationtype="validatediscourse" uttgenerationdiscourse="cidadeorigem">
  <one-of>
    <item>A cidade de origem e a cidade de destino não podem ser iguais. Por favor, indique
          uma nova cidade de origem.</item>
  </one-of>
</rule>

<rule id="14" uttgenerationtype="validatediscourse" uttgenerationdiscourse="cidadedestino">
  <one-of>
    <item>A cidade de origem e a cidade de destino não podem ser iguais. Por favor, indique
          uma nova cidade de destino.</item>
  </one-of>
```

```
  </rule>

  <rule id="15" uttgenerationtype="validatediscourse" uttgenerationdiscourse="horapartida">
    <one-of>
      <item>A hora de partida e a hora de chegada não podem ser iguais. Por favor, indique
            uma nova hora de partida.</item>
    </one-of>
  </rule>

  <rule id="16" uttgenerationtype="validatediscourse" uttgenerationdiscourse="horachegada">
    <one-of>
      <item>A hora de partida e a hora de chegada não podem ser iguais. Por favor, indique
            uma nova hora de chegada.</item>
    </one-of>
  </rule>

  <rule id="17" uttgenerationtype="showresults" uttgenerationresult="success">
    <one-of>
      <item>Tem as seguintes horas de partida disponíveis: [$horapartida]</item>
      <item>Tem as seguintes horas de chegada disponíveis: [$horachegada]</item>
      <item>A viagem com partida de [$cidadeorigem], e destino a [$cidadedestino],
            pode ficar marcada para as [$horapartida], com chegada às
            [$horachegada].</item>
    </one-of>
  </rule>

  <rule id="18" uttgenerationtype="showresults" uttgenerationresult="failure">
   <one-of>
     <item>Aconteceu um erro na execução do serviço.</item>
     <item>Ocorreu um problema e não poderemos [$action] o que pretende.</item>
     <item>Ocorreu um problema e não poderemos [$action] o [$target] que pretende.</item>
   </one-of>
  </rule>

  <rule id="19" uttgenerationtype="showresults" uttgenerationresult="unavailable">
   <one-of>
     <item>Peço desculpa, mas não existe [$target] disponível a essa hora.</item>
   </one-of>
  </rule>

  <rule id="20" uttgenerationtype="clarify" uttgenerationslot="">
   <one-of>
     <item>Não existem serviços disponíveis para o que pretende.</item>
   </one-of>
  </rule>

 </composedrules>

</grammar>

</domain>
```

B.2 Descrição dos Dispositivos

B.2.1 Máquina de vender bilhetes de Autocarro

```
<?xml version="1.0" encoding="UTF-8" ?>
<device name="Controlador dos Autocarros" domain="Autocarros">
        <!-- Descrição das funcionalidades do dispositivo -->
        <description>Este controlador gere a base de dados dos Autocarros.</description>

        <state name="on-line">
                <service name="consultar hora de partida">
                        <label>Consultar horas de partida</label>
                        <params>
                                <param slotkey="1" value="consultar"></param>
                                <param slotkey="2" value="horapartida"></param>
                                <param slotkey="3"></param>
                                <param slotkey="4"></param>
                                <param slotkey="5"></param>
                        </params>
                        <execute>
                                <name>consultarHorapartida</name>
                                <success></success>
                                <failure></failure>
                        </execute>
                </service>

                <service name="consultar hora de chegada">
                        <label>Consultar horas de chegada</label>
                        <params>
                                <param slotkey="1" value="consultar"></param>
                                <param slotkey="2" value="horachegada"></param>
                                <param slotkey="3"></param>
                                <param slotkey="4"></param>
                                <param slotkey="5"></param>
                        </params>
                        <execute>
                                <name>consultarHorachegada</name>
                                <success></success>
                                <failure></failure>
                        </execute>
                </service>

                <service name="consultar horario">
                        <label>Consultar horários dos autocarros</label>
                        <params>
                          <param slotkey="1" value="consultar"></param>
                          <param slotkey="2" value="horário"></param>
                          <param slotkey="3"></param>
                          <param slotkey="4"></param>
                          <param slotkey="5"></param>
                        </params>
                        <execute>
                                <name>consultarHorario</name>
                                <success></success>
                                <failure></failure>
                        </execute>
```

```
                </service>

                <service name="comprar bilhete">
                        <label>Consultar horários dos autocarros</label>
                        <params>
                          <param slotkey="1" value="comprar"></param>
                          <param slotkey="2" value="bilhete"></param>
                          <param slotkey="3"></param>
                          <param slotkey="4"></param>
                          <param slotkey="5"></param>
                          <or1>
                            <param slotkey="6"></param>
                            <param slotkey="7"></param>
                          </or1>
                        </params>
                        <execute>
                                <name>comprarBilhete</name>
                                <success></success>
                                <failure></failure>
                        </execute>
                </service>

        <validation>
                <not>
                        <equal>
                                <params>
                                        <param slotkey="4"></param>
                                        <param slotkey="5"></param>
                                </params>
                        </equal>
                </not>
        </validation>
        <validation>
                <ascending>
                        <params>
                                <param slotkey="6"></param>
                                <param slotkey="7"></param>
                        </params>
                </ascending>
        </validation>

        </state>

</device>
```