



Francisco Marcelo
Gomes Quinderé

Comunicação Humano-Robô
através de Linguagem Falada



**Francisco Marcelo
Gomes Quinderé**

**Comunicação Humano-Robô
através de Linguagem Falada**

Tese apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Doutor em Engenharia Electrotécnica, realizada sob a orientação científica do Doutor Luís Filipe de Seabra Lopes, Professor Associado do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro e do Doutor António Joaquim da Silva Teixeira, Professor Auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

para Natália

o júri

presidente

Doutor Fernando Manuel Bico Marques

Professor Catedrático da Universidade de Aveiro

Doutora Maria Beatriz Alves Sousa Santos

Professora Associada com Agregação da Universidade de Aveiro

Doutor Nuno João Neves Mamede

Professor Associado do Instituto Superior Técnico da Universidade Técnica de Lisboa

Doutor Luís Filipe de Seabra Lopes

Professor Associado da Universidade de Aveiro (Orientador)

Doutor Carlos Jorge da Conceição Teixeira

Professor Auxiliar da Universidade de Lisboa

Doutor Alexandre José Malheiro Bernardino

Professor Auxiliar do Instituto Superior Técnico da Universidade Técnica de Lisboa

Doutor António Joaquim da Silva Teixeira

Professor Auxiliar da Universidade de Aveiro (Coorientador)

agradecimentos

Gostaria de agradecer ao Luís e ao Teixeira pela orientação e apoio, que tornaram possível a realização deste trabalho.

Também gostaria de agradecer ao meu compadre Gabriel, pela amizade e pela indicação deste programa de doutorado.

Agradeço a Rogê pela recepção calorosa, que amenizou bastante as dificuldades dos primeiros meses dessa jornada. Também sou muito grato ao Vítor, João, Antónia, Meire, Celso, Francesca, Mariana, Bartira, Jamille e Fabiane pelos bons momentos compartilhados em Aveiro.

Aos companheiros de laboratório Mário, QingHua, Filipe, Pedro do Mar e Pimentas, agradeço pelo companheirismo e ajuda nos trabalhos.

Devo um agradecimento especial ao amigo Aneesh por toda a força.

Não posso deixar de agradecer a Paim e Mainha pelo incansável apoio em todos os momentos. A Gustavo, Mário Henrique, Ana Luíza, Neidinha, Tominhas e toda minha família pelo suporte.

Finalmente, agradeço a Natália, por tudo.

palavras-chave

interação humano-robô, linguagem falada, gestão de diálogo, representação de conhecimento, avaliação de sistemas de diálogo

resumo

Nos últimos anos, as tecnologias que dão suporte à robótica avançaram expressivamente. É possível encontrar robôs de serviço nos mais variados campos. O próximo passo é o desenvolvimento de robôs inteligentes, com capacidade de comunicação em linguagem falada e de realizar trabalhos úteis em interação/cooperação com humanos.

Torna-se necessário, então, encontrar um modo de interagir eficientemente com esses robôs, e com agentes inteligentes de maneira geral, que permita a transmissão de conhecimento em ambos os sentidos. Partiremos da hipótese de que é possível desenvolver um sistema de diálogo baseado em linguagem natural falada que resolva esse problema. Assim, o objetivo principal deste trabalho é a definição, implementação e avaliação de um sistema de diálogo utilizável na interação baseada em linguagem natural falada entre humanos e agentes inteligentes.

Ao longo deste texto, mostraremos os principais aspectos da comunicação por linguagem falada, tanto entre os humanos, como também entre humanos e máquinas. Apresentaremos as principais categorias de sistemas de diálogo, com exemplos de alguns sistemas implementados, assim como ferramentas para desenvolvimento e algumas técnicas de avaliação.

A seguir, entre outros aspectos, desenvolveremos os seguintes: a evolução levada a efeito na arquitetura computacional do Carl, robô utilizado neste trabalho; o módulo de aquisição e gestão de conhecimento, desenvolvido para dar suporte à interação; e o novo gestor de diálogo, baseado na abordagem de "Estado da Informação", também concebido e implementado no âmbito desta tese.

Por fim, uma avaliação experimental envolvendo a realização de diversas tarefas de interação com vários participantes voluntários demonstrou ser possível interagir com o robô e realizar as tarefas solicitadas. Este trabalho experimental incluiu avaliação parcial de funcionalidades, avaliação global do sistema de diálogo e avaliação de usabilidade.

keywords

human-robot interaction, spoken language, dialogue management, knowledge representation, dialogue systems evaluation

abstract

In recent years, robotics-related technologies have reached a remarkable level of maturity. Service robots can be found in various fields. The next step is the development of intelligent robots, capable of communicating in spoken language and doing useful work in interaction / cooperation with humans.

It is then necessary to find a way to efficiently interact with these robots, and with intelligent agents in general, enabling the transmission of knowledge in both directions. We will assume that one can develop a spoken language dialogue system to solve this problem. Therefore, the main goal of this work is the design, implementation and evaluation of a dialogue system that can be used on spoken language interaction between humans and intelligent agents.

Throughout this document, we present and discuss the main aspects related to spoken language communication, among humans as well as between humans and machines. We present the main dialogue system categories, with examples of some implemented systems, development tools and a few evaluation techniques.

Then, we describe the developed dialog system and its integration in a real robot, including the following aspects: the evolution in the computational architecture of Carl, the robot used in this work; the knowledge acquisition and management module, developed to support the interaction; and the new dialogue manager, based on the "Information State" approach, also designed and implemented within this thesis work.

Finally, an experimental evaluation involving the completion of several interaction tasks involving several volunteers proved to be possible to interact with the robot and perform the requested tasks. The evaluation includes a partial evaluation of features, an overall evaluation of the dialogue system and a usability evaluation.

Conteúdo

Conteúdo	i
Lista de Figuras	vii
Lista de Tabelas	ix
Acrónimos	xi
1 Introdução	1
1.1 Motivação	1
1.2 Enquadramento: o Projeto CARL	2
1.2.1 Infância	2
1.2.2 Tornando-se Útil: <i>Hors d'Oeuvres anyone?</i>	2
1.2.3 Tornando-se Adulto	4
1.3 Objetivos	5
1.4 Estrutura da Tese	6
1.5 Trabalho Publicado	6
2 Linguagem Falada e Sistemas de Diálogo	9
2.1 Comunicação entre Humanos	9
2.1.1 Tomada de Palavra	11
2.1.2 Atos de Fala	12
2.1.3 Ação Conjunta	14
2.1.4 Estrutura Conversacional	15
2.1.5 Implicatura Conversacional	15
2.2 Sistema de Diálogo	16
2.2.1 Reconhecimento de Fala	16
2.2.2 Compreensão de Linguagem Natural	17
2.2.3 Gestão de Diálogo	18
2.2.4 Geração de Linguagem Natural	18
2.2.5 Síntese de Fala	19

2.3	Tipos de Gestão de Diálogo	19
2.3.1	Sistemas Baseados em Estados Finitos	20
2.3.2	Sistemas Baseados em Formulários	20
2.3.3	Sistemas Baseados em Modelos de Crenças, Desejos e Intenções	22
2.3.4	Processos de Decisão Markovianos	24
2.3.5	Sistemas Baseados no Estado da Informação	25
2.3.6	Sistemas Multi-Estratégia	26
2.4	Exemplos Representativos de Sistemas de Diálogo	26
2.4.1	Sistemas Genéricos	26
2.4.1.1	TRIPS	26
2.4.1.2	Queen's Communicator	28
2.4.2	Sistemas de Planejamento de Viagens	28
2.4.2.1	TRAINS	28
2.4.2.2	DARPA Communicator	30
2.4.3	Sistemas de Telecomunicações	30
2.4.3.1	Florence	31
2.4.4	Sistemas de Comércio Eletrônico	33
2.4.4.1	ADVICE	33
2.4.5	Sistemas de Procura de Informação	33
2.4.5.1	IMIX	34
2.4.6	Sistemas Robóticos	34
2.4.6.1	ASIMO	34
2.4.6.2	Jijo-2	36
2.4.6.3	BIRON e ToBI	38
2.4.6.4	HERMES	40
2.4.6.5	MARVIN e ARTOS	41
2.4.6.6	Carl	41
2.5	Ferramentas para Desenvolvimento de Sistemas e Gestores de Diálogo	41
2.5.1	Galaxy II	42
2.5.2	ARIADNE	42
2.5.3	TrindiKit	44
2.5.4	DIPPER	44
2.5.5	Speech Graffiti	45
2.5.6	RavenClaw/Olympus	45
2.6	Avaliação de Sistemas de Diálogo	46
2.7	Atualização do Estado da Arte	47

3	Arquitetura Computacional e Sistema de Diálogo	51
3.1	Arquitetura Computacional	51
3.1.1	Arquitetura Computacional Anterior	51
3.1.2	Nova Arquitetura – Open Agent Architecture (OAA)	54
3.1.2.1	Comunicação entre os Agentes	55
3.1.2.2	Desenvolvimento de Agentes OAA	55
3.1.2.3	Agentes OAA na Arquitetura do Carl	56
3.1.2.4	Aspectos da Implementação dos Agentes	60
3.2	Compreensão de Linguagem Falada	61
3.2.1	Atos de Fala	61
3.2.2	Linguagem de Representação	62
3.2.3	Análise Sintática	63
3.2.4	Análise Semântica	66
3.3	Aquisição e Gestão de Conhecimento	67
3.3.1	Trabalho Relacionado	67
3.3.2	Mecanismos de Inferência Básicos	68
3.3.2.1	Dedução	68
3.3.2.2	Indução	70
3.3.2.3	Analogia	71
3.3.3	Resposta a Perguntas	72
3.3.4	Procedimentos de Acesso ao Módulo KAM	73
3.4	Geração de Fala	76
3.4.1	Ferramentas Analisadas	76
3.4.1.1	Astrogen	76
3.4.1.2	Functional Unification Formalism (FUF)	77
3.4.2	Implementação	79
4	O Gestor de Diálogo	81
4.1	Requisitos para o Gestor de Diálogo	81
4.2	Arquitetura do Gestor de Diálogo Desenvolvido	83
4.3	Estado da Informação	85
4.4	Eventos e Condições	86
4.5	Ações	88
4.6	Tarefas	89
4.6.1	Exemplos de Tarefas	90
4.6.1.1	Armazenamento de Informação	90
4.6.1.2	Resposta a Questões Gerais	91
4.6.1.3	Resposta a Questões de Confirmação	91

4.6.1.4	Clarificação – Baixa Confiança do Agente de Reconhecimento de Fala Automático	92
4.6.1.5	Clarificação – Resolução de Pronomes	93
4.6.1.6	Tomada de Iniciativa	94
4.7	Regras de Atualização	95
4.7.1	Estrutura de uma Regra	96
4.7.2	Exemplos de Regras Gerais	96
4.7.2.1	Tomada de Palavra	96
4.7.2.2	Armazenamento de Informação	96
4.7.2.3	Iniciativa do Diálogo	97
4.7.2.4	Ambiente Ruidoso	97
4.7.3	Regras para Diálogos de Clarificação	98
4.7.3.1	Resolução de Pronomes Anafóricos	100
4.7.3.2	Clarificação de Pronomes Anafóricos	100
4.7.3.3	Lista de Referentes	101
4.8	Controle do Gestor de Diálogo	103
4.8.1	Módulo de Controle	103
4.8.2	Atualização do Estado da Informação	104
4.8.3	Ação	104
5	Avaliação	107
5.1	Avaliação de Funcionalidades	107
5.1.1	Compreensão de Frases Declarativas	107
5.1.1.1	Método	108
5.1.1.2	Resultados	109
5.1.2	Aquisição de Conhecimento e Resposta a Perguntas	110
5.1.2.1	Método	110
5.1.2.2	Resultados	111
5.1.3	Resolução de Pronomes	111
5.1.3.1	Método	111
5.1.3.2	Resultados	112
5.2	Avaliação Global do Sistema	113
5.2.1	Objetivos	113
5.2.2	Método	113
5.2.2.1	Treino do Reconhecedor Automático de Fala	113
5.2.2.2	Protocolo de Avaliação	114
5.2.3	Resultados	116
5.2.3.1	Realização das Tarefas	116
5.2.3.2	Aspectos Relativos à Interação	118

5.2.3.3	Análise de Falhas	124
5.2.3.4	Aquisição de Conhecimento	127
5.3	Avaliação de Usabilidade	127
5.4	Discussão	128
6	Conclusões	131
6.1	Resumo do Trabalho Realizado	131
6.1.1	Arquitetura Computacional	131
6.1.2	Representação do Conhecimento e Raciocínio	131
6.1.3	Geração de Linguagem Natural	132
6.1.4	Gestor de Diálogo	132
6.1.5	Avaliação de Funcionalidades	133
6.1.6	Avaliação Global do Sistema	133
6.2	Principais Resultados	134
6.3	Sugestões para Trabalho Futuro	135
	Bibliografia	137
A	Interface de Acesso ao Estado da Informação	151
B	Acesso às Tarefas no Estado da Informação	153
B.1	Exemplos de Tarefas	153
B.1.1	Abertura	153
B.1.2	Aquisição do Nome do Utilizador	154
B.1.3	Rejeição – Confiança Muito Baixa do Agente de Reconhecimento de Fala Automático	154
B.1.4	Clarificação – Análise Semântica Parcial	155
B.1.5	Bateria	155
B.1.6	Ambiente Ruidoso	155
B.2	Exemplos de Regras Gerais	156
B.2.1	Saudação	156
B.2.2	Despedida	156
B.2.3	Bateria	156
C	Frases Utilizadas nas Experiências para Avaliação de Funcionalidades	159
D	Frases Utilizadas nas Experiências para Resolução de Pronomes	161
E	Tabelas Extras	163

Lista de Figuras

1.1	Aparência do Carl em 2000	3
1.2	Carl na capa da revista <i>AI Magazine</i> (Vol 23, Número 1, Primavera de 2002)	4
1.3	Aparência do Carl desde 2002	5
2.1	Diagrama geral de um sistema de diálogo	17
2.2	Modelo BDI [McTear, 2002]	24
2.3	TRIPS [Allen <i>et al.</i> , 2000]	27
2.4	Arquitetura do sistema TRAINS-93 [Allen, 1994]	29
2.5	Arquitetura do <i>Spoken Dialogue System</i> da AT&T [Fabbrizio e Lewis, 2004]	31
2.6	Arquitetura Florence [Fabbrizio e Lewis, 2004]	32
2.7	Arquitetura desenvolvida para o robô ASIMO [Nakano <i>et al.</i> , 2005]	35
2.8	Estrutura interna do módulo MEBDP [Nakano <i>et al.</i> , 2005]	36
2.9	Módulos no sistema do robô Jijo-2 [Asoh <i>et al.</i> , 2001]	37
2.10	Robô Jijo: máquina de estados finito para controle do processo de diálogo [Asoh <i>et al.</i> , 2001]	38
2.11	Robô ToBI e seus componentes: câmera, microfone, braço e sensor laser [Wachsmuth <i>et al.</i> , 2009]	39
2.12	Robô ToBI – operações que manipulam a memória ativa [Wachsmuth <i>et al.</i> , 2009]	39
2.13	HERMES – Interface Homem-Máquina: observe que o módulo de situação interfere no reconhecimento, interpretação de comandos e alimenta a comunicação de saída [Bischoff e Graefe, 2002]	40
2.14	Arquitetura do TrindiKit – destaque para o Estado da Informação e módulo DME [Larsson e Traum, 2000]	44
3.1	Arquitetura computacional anterior do Carl [Seabra Lopes, 2002]	52
3.2	Módulo de gestão central na anterior arquitetura– um processo baseado em eventos	53
3.3	Exemplos de transições de estado na anterior arquitetura	54
3.4	Agentes OAA no robô Carl – Comunicação Lógica	56

3.5	Agentes OAA no robô Carl – ligações de comunicação direta entre os agentes de síntese e GTI.	60
3.6	A arquitetura do sistema Spoken Language Understanding (SLU) [Rodrigues <i>et al.</i> , 2004]	65
3.7	Exemplo de inferência dedutiva: ‘Sócrates é mortal’	69
3.8	Exemplo de inferência indutiva	71
3.9	Exemplo de inferência por analogia	71
3.10	Comportamento da fórmula 3.2: os valores de $conf(N, T)$ tendem para $\frac{N}{T}$ à medida que T aumenta	72
3.11	Arquitetura do Astrogen	77
4.1	Gestor de diálogo e sua interação com os demais agentes da arquitetura computacional	84
4.2	Diagrama de classes do gestor de diálogo	85
4.3	Estado da Informação como classe	86
4.4	Fluxo das atualizações	98
5.1	Conjuntos de relações extraídas e não extraídas, bem como relevantes e não relevantes	108
5.2	Fluxo das atualizações: resultado da experiência	112
5.3	Tomadas de palavra por participante	119
5.4	Tomadas de palavra por tarefa	120
5.5	Tipo de elocuições utilizadas por participante	121
5.6	Tipo de elocuições utilizadas por tarefa	122
5.7	Avaliação dos participantes	129

Lista de Tabelas

2.1	Formulário para um sistema de viagens aéreas	17
2.2	Exemplo de diálogo gerado pelo sistema ELIZA	18
2.3	Exemplo de sistema baseado em estados finitos – diálogo do sistema Nuance Automatic Banking [McTear, 2002]	21
2.4	Exemplo de sistema baseado em formulário – diálogo do sistema Philips Automatic Train Timetable Information [Aust <i>et al.</i> , 1995]	23
2.5	Exemplo de regras de obrigação [Traum e Allen, 1994]	24
2.6	Atualização do estado arte em sistemas de diálogo utilizados na interação homem-robô	48
3.1	Mensagens trocadas pelos agentes	57
3.2	Atos de fala reconhecidos e/ou gerados pelo Carl	62
3.3	Predicados usados para representação de conhecimento	63
3.4	Alguns exemplos de representação semântica de frases	64
3.5	Alguns exemplos de análise sintática	66
3.6	Procedimentos de acesso ao módulo KAM	75
3.7	Exemplos de utilização do agente NLG	80
4.1	Diálogo fictício com o robô Carl, o qual representa o objetivo/visão para o sistema de diálogo.	82
4.2	Descrição dos campos do Estado da Informação	87
4.3	Descrição dos eventos: mensagens de agentes ou temporizadores	88
4.4	Condições verificadas internamente	88
4.5	Regras relacionadas aos eventos gerados pelo agente NLU	99
4.6	Fatores de saliência utilizados pelo algoritmo 2	102
5.1	Resultado da análise semântica de frases declarativas (SLU/DM) – sem confirmação	109
5.2	Resultado da análise semântica de frases declarativas (SLU/DM) – com confirmação	110

5.3	Resumo da análise semântica de frases declarativas (SLU/DM)	110
5.4	Resultados do módulo de gestão e aquisição de conhecimento - utopia <i>vs.</i> realidade.	111
5.5	Resolução de pronomes.	113
5.6	Tarefas de interação usadas na avaliação	115
5.7	Sucesso global na realização das tarefas	116
5.8	Sucesso na realização das tarefas de cada participante	117
5.9	Sucesso na realização dos diferentes tipos de tarefa	117
5.10	Sucesso na realização das diferentes tarefas	118
5.11	Tomada de palavra	119
5.12	Tomadas de palavra por tipo de tarefa	120
5.13	Tipos de elocuições dos participantes	121
5.14	Tipo de elocuições utilizadas por tipo de tarefa	122
5.15	Resposta a perguntas realizadas pelo robô	123
5.16	Pergunta respondida pelo participante	123
5.17	Resumo das tarefas criadas pelos participantes	124
5.18	Algumas tarefas criadas pelos participantes	124
5.19	Elocuições dos participantes rejeitadas pelo gestor de diálogo	125
5.20	Desempenho do robô na identificação dos atos de fala	125
5.21	Matriz de confusão de atos de fala no agente NLU. Para facilitar a leitura, os acertos estão em negrito e a principal troca, sublinhada.	126
5.22	Falhas de interação do robô	126
5.23	Evidências de aquisição de conhecimento	127
5.24	Questionário sobre satisfação global	128
C.1	Frases utilizadas – avaliação de funcionalidades	160
D.1	Frases utilizadas – resolução de pronomes	161
E.1	Tomadas de palavra por participante	163
E.2	Tomadas de palavra por tarefa	164
E.3	Tipo de elocuições utilizadas por participante	166
E.4	Tipo de elocuições utilizadas por tarefa	167

Acrónimos

AAAI American Association for Artificial Intelligence – *Associação americana para a Inteligência Artificial.*

ACL Agent Communication Language – *Linguagem de comunicação entre agentes.*

APPIA Associação Portuguesa Para a Inteligência Artificial.

ASR Automatic Speech Recognition – *Reconhecimento automático de fala.*

ATN Augmented Transition Network – *Rede de transições aumentada.*

BDI Belief, Desire, Intention – *Crença, desejo, intenção.*

CARL Communication Action Reasoning and Learning in Robotics – *Comunicação, ação, raciocínio e aprendizagem em Robótica.*

CORBA Common Object Request Broker Architecture – *Arquitetura comum para agente de requisição de objetos.*

CPK Center for PersonKommunikation.

CRT Conversation Representation Theory – *Teoria de representação da conversação.*

DARPA Defense Advanced Research Projects Agency – *Agência de pesquisa de projetos avançados de defesa.*

DM Dialogue Manager – *Gestor de diálogo.*

DME Dialogue Move Engine – *Motor de movimento do diálogo.*

DRT Discourse Representation Theory – *Teoria de representação do discurso.*

EBTL Event-Based Temporal Logic – *Lógica temporal baseada em eventos.*

EL Episodic Logic – *Lógica episódica.*

EMMA Extensible MultiModal Annotation – *Anotação multimodal extensível.*

EURON European Robotics Research Network – *Rede europeia de pesquisa em Robótica.*

FD Functional Description – *Descrição funcional.*

FUF Functional Unification Formalism – *Formalismo de unificação funcional.*

GTI Graphical and Touch Interface – *Interface gráfica e tátil.*

HRCL Human-Robot Communication Language – *Linguagem de comunicação humano-robô.*

IEETA Instituto de Engenharia Electrónica e Telemática de Aveiro.

ICL Interagent Communication Language – *Linguagem de comunicação entre agente.*

KAM Knowledge Acquisition and Management – *Aquisição e gestão de conhecimento.*

KQML Knowledge Query and Manipulation Language – *Linguagem de consulta e manipulação de conhecimento.*

KR Knowledge Representation – *Representação de conhecimento.*

LTPOS Language Technology Part-Of-Speech – *Tecnologia de linguagem: classe gramatical.*

LFG Lexical Functional Grammar – *Gramática léxico-funcional.*

MBL Memory Based Learning – *Aprendizagem baseada na memória.*

MDP Markov Decision Process – *Processo de decisão markoviano.*

MEBDP Multi-Expert-based Behavior and Dialogue Planning – *Planejamento de comportamento e diálogo baseado em múltiplos especialistas.*

MUG Multimodal Functional Unification Grammar – *Gramática de unificação funcional multimodal.*

NLA Natural Language Analysis – *Análise de linguagem natural.*

NLG Natural Language Generation – *Geração de linguagem natural.*

NLSML Natural Language Semantic Markup Language – *Linguagem de marcação de semântica de linguagem natural.*

NLU Natural Language Understanding – *Compreensão de linguagem natural.*

NLP Natural Language Processing – *Processamento de linguagem natural.*

OAA Open Agent Architecture – *Arquitetura de agentes aberta.*

POS Part-Of-Speech – *Classe gramatical.*

SDS Spoken Dialogue System – *Sistemas de diálogo falado.*

SER Sentence Error Rate – *Taxa de erro de locução.*

SLU Spoken Language Understanding – *Compreensão de linguagem falada.*

SPASS Synergetic Prover Augmenting Superposition with Sorts – *Provador sinérgico de superposição ampliada com ordenações.*

SURGE Systemic Unification Realization Grammar of English – *Gramática de realização de unificação sistêmica de inglês.*

TiMBL Tilburg Memory Based Learner – *Aprendiz baseado em memória de Tilburg.*

TTS Text-To-Speech – *Sintetizador de fala.*

UML Unified Modeling Language – *Linguagem de modelagem unificada.*

WER Word Error Rate – *Taxa de erro de palavra.*

Capítulo 1

Introdução

1.1 Motivação

Imagine chegar em casa cansado depois de um longo dia de trabalho e, ao invés de ter que encarar mais um turno de labuta com tarefas domésticas, poder relaxar na sua poltrona com uma boa música, um bom livro ou filme, enquanto Rosie, sua recém adquirida robô de serviço, termina de preparar o jantar e lhe pergunta o que deseja para sobremesa! Ah, claro que durante o dia Rosie já andou a fazer as tarefas que você pediu para ela de manhã. Que tal?

Além de cuidar da casa, robôs como *Rosie* poderiam ajudar no acompanhamento de pessoas idosas ou com doenças que necessitem de atenção especial. Nesse caso, lembrariam o horário de tomar medicamentos e também poderiam participar de atividades lúdicas.

Podemos pensar em uma série de áreas de atuação para esses robôs inteligentes, como servir de anfitriões em eventos ou instituições, guias em museus, auxiliares de enfermagem, agentes de entretenimento, assistentes pessoais, além das já mencionadas tarefas domésticas.

Uma recente pesquisa, realizada por membros do Georgia Tech com o objetivo de identificar as necessidades de possíveis usuários de robôs domésticos, indica três grupos de tarefas domésticas desejadas: trabalhos demorados, gestão da casa e assistência pessoal [Sung *et al.*, 2009].

Para os entrevistados, *Rosie* deveria ser capaz de, entre outras coisas: lavar pratos e roupas, aspirar a casa, cuidar das plantas, cozinhar, limpar a casa, dar segurança à casa, atender à porta, relatar notícias, dar conselhos financeiros, fazer massagem, além de reproduzir música e exibir filmes!

Nessa mesma pesquisa, os participantes mostraram a necessidade de eficiência ao interagir com os robôs e indicaram decididamente **a fala como o modo de interação preferido**. Um dos participantes chegou a desenhar orelhas em um robô para ressaltar a importância dessa capacidade. Além da *acessibilidade* – essa característica de fácil interação através de linguagem falada –, a próxima geração de robôs de serviço também precisa ser *ativa e adap-*

tável [Seabra Lopes e Connell, 2001]. Um robô *ativo* é um robô que reage às mudanças de condições em seu ambiente, implicando em boa capacidade de percepção e ação. Por *adaptável* devemos entender que o robô deve se adequar a diferentes usuários e ambientes, e para isso, precisa ser capaz de decidir sobre tarefas e ser reconfigurável. Isso implica em possuir uma boa capacidade de aprendizado e aquisição de conhecimento, possibilitando tornar-se um robô companheiro [Dautenhahn, 2004; Dautenhahn *et al.*, 2005].

1.2 Enquadramento: o Projeto CARL

Projeto CARL (*Communication, Action, Reasoning and Learning in Robotics*) foi implantado na Universidade de Aveiro com o objetivo de estudar a integração das quatro dimensões do problema de construir um robô inteligente: comunicação, percepção & ação, raciocínio e aprendizagem [Seabra Lopes e Teixeira, 2000].

1.2.1 Infância

Nos seus primeiros anos, de 1999 a 2000, o robô Carl era apenas uma plataforma móvel comercial aumentada com capacidades de interação falada simples. A plataforma escolhida foi a Pioneer 2-DX da *ActivMedia Robotics* (atual *Mobile Robots Inc.* ¹). Ela inclui rodas motorizadas, sensores de colisão e sonares na frente e atrás do robô, além de uma placa de som. A plataforma adquirida também contém um micro-controlador baseado no processador Siemens C166 e um computador Pentium 266 MHz com barramento PC104, 64MB de memória e 3.2 GB de disco rígido. O sistema operativo utilizado era o Red Hat Linux.

Com esta plataforma móvel, foram desenvolvidas capacidades simples de navegação baseada em diálogo. Para tanto, uma coluna e um receptor para um microfone sem fios remoto (Shure WH20) foram adicionados à base (figura 1.1).

O aplicativo *Entropic grapHvite* foi utilizado para reconhecimento de fala. Através de comandos simples, como “*move forward*” ou “*turn left*”, era possível guiar o robô por diferentes salas e corredores num edifício. Alguns experimentos iniciais foram realizados no reconhecimento de frases mais complexas utilizando um vetor de microfones digital *Labtec LVA 7280* [Seabra Lopes e Teixeira, 2000].

1.2.2 Tornando-se Útil: *Hors d’Ouevres anyone?*

Para a edição de 2001 da *AAAI Mobile Robot Competition and Exhibition*, organizada pela American Association for Artificial Intelligence (AAAI), uma das competições anunciadas tinha como título “*Hors d’Ouevres anyone?*” (“*Aceita um acepipe?*”). Esta era uma competição de interação humano-robô, na qual os robôs participantes deveriam distribuir acepipes (*hors*

¹<http://www.mobilerobots.com>.



Figura 1.1: Aparência do Carl em 2000

d'Ouevres) numa recepção. O Carl estava presente [Seabra Lopes, 2001; Seabra Lopes *et al.*, 2003a].

De modo a satisfazer os requisitos principais da competição, várias melhorias foram introduzidas. Um dos problemas do Carl na época era o seu tamanho, baixo demais para uma interação amigável com humanos. Para pegar comida da base P2-DX seria necessário inclinar-se praticamente ao chão. Além disso, experimentos anteriores haviam demonstrado que o vetor de microfones posicionado a 25 cm do chão (ou 1,5 m do rosto do utilizador) possuía um desempenho muito ruim.

Dadas essas limitações, uma estrutura de fibra de vidro foi adicionada à base P2-DX, fazendo o Carl atingir aproximadamente 85 cm de altura (figura 1.2). Em vez do receptor de microfones sem fios usado anteriormente, esta estrutura carregava agora um vetor de microfones direcional, o *Andrea Electronics DA-400 v2*, e uma coluna de som. A mudança de microfones visava obter um melhor desempenho, especialmente em ambientes ruidosos. Agora, o rosto de uma pessoa de pé próxima ao robô ficava a uma distância de aproximadamente 1 m do vetor de microfones. Isso era suficiente para permitir um reconhecimento de fala num ambiente silencioso. A estrutura também incluía um recipiente para pequenos objetos, equipado com um sensor infravermelho para detectar a presença de objetos. Para uma navegação mais robusta, um conjunto de sensores infravermelhos foi adicionado à estrutura de fibra de vidro. Uma câmera *Sony EVI pan-tilt-zoom* também foi acoplada à base P2-DX.

Tendo em consideração os objetivos gerais do projeto e os requisitos específicos da compe-

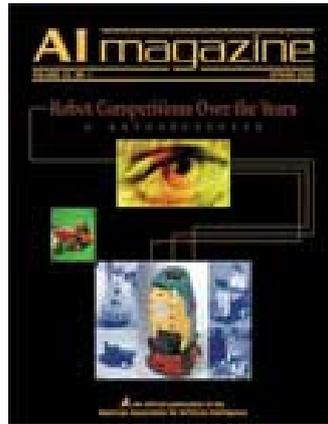


Figura 1.2: Carl na capa da revista *AI Magazine* (Vol 23, Número 1, Primavera de 2002)

tição “Hors d’Oeuvres anyone?”, uma nova arquitetura de software foi desenhada e implementada. Um gestor central baseado em eventos integrava a navegação com gestão de diálogo. Uma gramática, aceitando 12000 frases diferentes foi escrita. O reconhecimento de fala e a síntese de voz eram tratados por processos Linux baseados no *ViaVoice* da IBM. Análise de linguagem natural e construção de estruturas de frases eram realizados por outro processo Linux baseado na ferramenta NLP do CPK – *Center for PersonKommunikation* [Brondsted, 1999]. Numa lista de seis robôs inscritos na competição *Hors d’Oeuvres anyone?*, o Carl era o único com capacidades de diálogo em linguagem falada e terminou a disputa em terceiro lugar (*Third Place Award*).

1.2.3 Tornando-se Adulto

Em 2002, a meta do projeto CARL era resolver as principais limitações identificadas durante o evento da American Association for Artificial Intelligence (AAAI). Para isso, um computador portátil *Fujitsu-Siemens Lifebook* baseado no processador Intel Pentium III 700 Mhz, com 256 MB de memória, disco de 20 GB e sistema operativo Linux foi adquirido. Uma interface gráfica e uma cara animada tornaram o Carl mais amigável e acessível. O portátil possui ainda ecrã tátil, permitindo assim uma nova modalidade de interação. Uma câmera *Creative WebCam Pro* foi conectada via porta USB ao portátil. Com estas adições, Carl passou a ter 1,10 m de altura (figura 1.3). O vetor de microfones *Andrea Electronics DA-400 v2* foi substituído por outro vetor de microfones direcional, um *VoiceTracker* da *Acoustic Magic*.

Uma versão do Carl, que incluía as capacidades de interação humano-robô supradescritas, foi apresentada na recepção da IROS’2002 (*IEEE/RSJ International Conference on Intelligent Robots and Systems*) como parte do *First International Cleaning Robots Contest* (Lausanne, Switzerland, Outubro de 2002)².

²Algumas fotos da recepção da IROS’2002, mostrando a participação do Carl, ainda estão disponíveis no



Figura 1.3: Aparência do Carl desde 2002

Dois anos depois, o Carl foi novamente demonstrado num evento internacional, a RoboCup'2004. Desta vez, foram apresentadas novas habilidades no fornecimento de indicações de direção aos visitantes. Dada a posição corrente do robô, um caminho para o destino era planejado e, a seguir, oralmente explicado e mostrado graficamente no ecrã.

Ao mesmo tempo, avanços mais fundamentais foram obtidos no processamento robusto de linguagem falada (detalhes na secção 3.2).

O robô Carl continua bastante popular e foi incluído numa seleção de robôs que ilustram um folheto da European Robotics Research Network (EURON)³.

1.3 Objetivos

Podemos definir o nosso problema como sendo o de encontrar um modo acessível e adaptável de interagir eficientemente com robôs, ou agentes inteligentes de maneira geral, que permita a transmissão de conhecimento em ambos sentidos.

Partiremos da hipótese de que é possível desenvolver um sistema de diálogo baseado em linguagem natural falada que resolva o nosso problema.

Assim, o objetivo principal deste trabalho é a definição, implementação e avaliação de um sistema de diálogo utilizável na interação baseada em linguagem natural falada entre humanos

sítio da conferência: <http://www.iros02.ethz.ch/gallery/Events>.

³<http://www.euron.org/miscdocs/leaflet1.pdf>, consultado em fevereiro de 2010

e agentes inteligentes.

O segundo objetivo, complementar, é o de utilizar esse sistema conjuntamente com um módulo de aquisição e gestão de conhecimento de forma a adquirir, acumular e reutilizar informação fornecida por diferentes interlocutores humanos.

O terceiro objetivo é contribuir na evolução do robô Carl, utilizado como plataforma física para desenvolvimento do trabalho, apresentado na seção 1.2 e descrito em mais detalhe no capítulo 3.

1.4 Estrutura da Tese

O capítulo 2 aborda conceitos importantes para o desenvolvimento da tese. Aqui, são analisados os principais aspectos da linguagem falada entre os humanos, como tomada de palavra e atos de fala. Também são descritos os componentes típicos de um sistema de diálogo. A seguir, são apresentados diferentes tipos de gestão de diálogo, com exemplos de sistemas implementados, funcionalidades, ferramentas para desenvolvimento e algumas técnicas de avaliação.

O capítulo 3 mostra as evoluções na arquitetura computacional do Carl com a adoção da Open Agent Architecture (OAA). Também descreve como é realizada a compreensão de linguagem falada no robô e a sua geração de fala. Este capítulo introduz o módulo de aquisição e gestão de conhecimento desenvolvido para dar suporte aos diálogos.

Logo depois, o capítulo 4 descreve a implementação do gestor de diálogo, mostrando os requisitos levantados para o sistema, assim como a abordagem escolhida, de Estado da Informação. A arquitetura do gestor de diálogo desenvolvido é apresentada, com detalhes sobre as regras de atualização do Estado da Informação, a influência de eventos e condições sobre essas regras, e como as tarefas e ações estão relacionadas ao processo.

O capítulo 5 apresenta uma avaliação global do desempenho do sistema e de sua usabilidade, com descrição dos objetivos, da metodologia adotada e dos resultados obtidos, também são mostradas avaliações das principais funcionalidades do sistema, como compreensão de frases e resolução de pronomes.

Finalmente, o capítulo 6 traz um resumo do trabalho realizado, destaca os principais resultados e indica sugestões para trabalhos futuros.

1.5 Trabalho Publicado

Parte do trabalho aqui relatado já foi apresentado à comunidade científica.

O módulo de aquisição e gestão do conhecimento – Knowledge Acquisition and Management (KAM) – foi descrito no workshop “Natural Language Understanding and Cognitive Science – NLUCS” [Seabra Lopes *et al.*, 2005a], que faz parte da “International Conference on Enterprise Information Systems – ICEIS” (Miami, Flórida, Estados Unidos, 2005).

Este módulo permite que o robô adquira conhecimento a partir de diferentes interlocutores e é capaz de lidar com eventuais informações contraditórias. Além disso, como a aquisição de informação é realizada, principalmente, através de linguagem natural falada, o grau de confiança do reconhecedor é levado em consideração pelo sistema. A linguagem de representação do conhecimento baseia-se em redes semânticas e incorpora, também, algumas noções de Unified Modeling Language (UML). A resposta a perguntas utiliza mecanismos de inferência dedutivos e indutivos.

Uma análise em conjunto do módulo KAM e dos outros agentes que compõem o robô foi apresentada no “Interspeech” (Lisboa, Portugal, 2005) [Seabra Lopes *et al.*, 2005b]. A avaliação realizada do funcionamento do módulo KAM em conjunto com o agente Natural Language Understanding (NLU) revelou um bom desempenho.

Descrito aqui em mais detalhes no capítulo 4, o gestor de diálogo foi apresentado inicialmente no workshop “Natural Language Processing and Cognitive Science – NLPCS” (Madeira, Portugal, 2007) [Quinderé *et al.*, 2007a].

A integração do gestor de diálogo com os demais agentes foi analisada no “Interspeech” de 2007 (Antuérpia, Bélgica, 2007) [Quinderé *et al.*, 2007b].

Uma análise preliminar da capacidade de aquisição de conhecimento através de linguagem falada foi apresentada na *8th Conference on Autonomous Robot Systems and Competitions*, evento associado ao Festival Nacional de Robótica 2008 (Aveiro, Portugal, 2008) [Seabra Lopes *et al.*, 2008]. Esse trabalho foi agraciado com o prêmio de melhor artigo de inteligência artificial apresentado na conferência, prêmio este concedido pela Associação Portuguesa Para a Inteligência Artificial (APPIA). Posteriormente, o artigo também foi selecionado para publicação na revista *Robótica* [Seabra Lopes *et al.*, 2009].

Um resumo da avaliação do gestor de diálogo desenvolvido foi aceito para apresentação oral na RO-MAN 2013 (Gyeongju, Coreia do Sul, 2013) [Quinderé *et al.*, 2013].

É importante ressaltar que o robô Carl tem sido citado com alguma frequência na literatura científica [Samani *et al.*, 2011; Mohan e Laird, 2011; LeBlanc, 2010; Tan *et al.*, 2010; Johnson e Agah, 2009; Loutfi *et al.*, 2008; Koch *et al.*, 2008; Vasudevan *et al.*, 2007; Kiatisevi *et al.*, 2006; Eliasson, 2005; Toptsis *et al.*, 2005; Hanafiah *et al.*, 2004; Kiatisevi *et al.*, 2004; Ogura *et al.*, 2003; Lauria *et al.*, 2001].

Capítulo 2

Linguagem Falada e Sistemas de Diálogo

Este capítulo aborda a linguagem falada e os sistemas de diálogo. Inicialmente, trataremos da comunicação entre humanos, mostrando algumas propriedades presentes em nossas conversações. A seguir, analisaremos como se dá o processo de comunicação falada entre humanos e máquinas.

Após essa análise, os sistemas de diálogo são descritos, através de suas categorias e exemplos. Algumas ferramentas para desenvolvimento de sistemas de diálogo são mostradas e, por último, técnicas de avaliação de sistemas de diálogo são discutidas.

2.1 Comunicação entre Humanos

Antes de abordarmos a comunicação homem-máquina, ou homem-robô, precisamos entender um pouco o que caracteriza a comunicação entre os humanos, em particular a comunicação falada. Vários estudos foram realizados na tentativa de compreender todas as nuances deste tipo de comunicação. Nickerson [1977] identificou várias propriedades presentes nas conversações humanas, nomeadamente:

- Bi-direcionalidade – o fluxo de informação ocorre nos dois sentidos.
- Iniciativa mista – amplia o conceito de bi-direcionalidade. Aqui, cada participante pode conduzir a conversa, fornecer informações que não foram pedidas e realizar perguntas por vontade própria.
- Percepção de quem está no controle – o que Nickerson chama de controle é também conhecido como posse da palavra (*turn*), é o momento de cada participante falar no diálogo.

- Regras para transferência de controle – ou tomada de palavra, seria a passagem da palavra entre os participantes do diálogo. Abordaremos com mais detalhes este e o tópico anterior na seção 2.1.1.
- Sensação de presença – a noção de cada participante de que o outro está presente na conversa, prestando atenção no seu desenrolar.
- Comunicação não verbal – é como diz o ditado: “um olhar vale mais do que mil palavras”. Exageros à parte, a comunicação não verbal é facilmente identificada nas conversas, seja explicitamente através de acenos com a cabeça e de gestos como o de apontar, ou mais sutilmente por olhares, o fato é que as pessoas também se comunicam sem falar durante um diálogo.
- Intolerância ao silêncio – notou-se que as pessoas consideram desagradável se deparar com um silêncio não esperado durante uma conversa. Como exemplo, imagine uma conversa telefônica na qual uma das participantes, após um longo período de argumentações, faz uma pergunta e, para sua surpresa, escuta o silêncio como resposta.
- Estrutura – foi observado que os diálogos possuem uma estrutura em comum, com início, fim, e assuntos que são abordados ao longo de sua duração. Há protocolos para iniciar e finalizar uma conversação. Por exemplo, pode-se iniciar com uma saudação e finalizar com uma despedida.
- Escala de tempo particular – segundo as pesquisas de Nickerson, o controle de diálogo é tipicamente passado de um participante para o outro a cada quatro ou cinco segundos, em média.
- Banda larga – não que necessariamente todo diálogo tenha uma alta taxa de transferência de informação, muitos não chegam a transferir informação alguma!, mas isso é algo que pode ocorrer naturalmente.
- Linguagem informal – uma das principais diferenças em relação ao texto escrito é a questão da informalidade dos diálogos, marcados por frases gramaticalmente incorretas ou incompletas e de um vocabulário bem mais reduzido.
- Contexto situacional compartilhado – ao longo de um diálogo, facilmente encontramos falas que só fazem sentido dentro de um contexto, perceptível ou conceitual. Esse contexto é compartilhado pelos participantes, tornando-se peça fundamental para o esclarecimento das ambiguidades encontradas durante o diálogo.
- Conhecimento geral do mundo – obviamente, durante as conversações, os participantes fazem referência, direta ou indiretamente, a conhecimentos gerais do mundo.

- Conhecimento específico compartilhado – além do conhecimento geral do mundo, também podemos identificar conhecimentos específicos que são utilizados em diálogos. Esses conhecimentos são compartilhados por grupos, como familiares, classes profissionais, vizinhos, etc.
- História – pessoas que se encontram com muita frequência e por um longo período de tempo, provavelmente passam a conversar de maneira diferente, tanto no tom como no conteúdo.
- Participantes do mesmo nível – Nickerson sugere que as conversas mais agradáveis envolvem participantes do mesmo nível intelectual, sem negar com isso a existência de conversas entre pessoas de níveis diferentes.

Jurafsky e Martin [Jurafsky e Martin, 2008] destacam como principais as seguintes propriedades da conversação humana: 1) a tomada de palavra, 2) os atos de fala, 3) a noção de ação conjunta durante um diálogo, a estrutura conversacional e o conceito de implicatura conversacional.

As seções seguintes abordam as propriedades destacadas por Jurafsky e Martin em mais detalhes.

2.1.1 Tomada de Palavra

Uma das características principais de um diálogo é a de que somente uma pessoa deve falar a cada momento. E para tal, deve possuir a palavra (*turn*). A posse da palavra é, assim, caracterizada pela produção de uma fala, ou sequência de falas, por um dado participante numa conversa, sem interrupção por parte de outros participantes. Embora tal situação possa sugerir vários atropelamentos em busca da palavra, estudos indicam que o total de sobreposição nos diálogos é inferior a 5% [Levinson, 1983].

A transferência da posse da palavra entre participantes num diálogo é designada como “tomada da palavra” (*turn-taking*). Para ilustrar isso, observe que o seguinte diálogo fictício possui 6 *tomadas de palavra*¹:

¹Assumindo que o início do diálogo também representa uma tomada de palavra do participante.

Participante:	Goog morning, Carl! (<i>tomada 1</i>)
Carl:	Good morning, how may I help you? (<i>tomada 2</i>)
Participante:	Where's prof. Gustavo? (<i>tomada 3</i>)
Carl:	He is at the departament. (<i>tomada 4</i>)
Carl:	But prof. Ana, his assistant, is over there at her office.
Participante:	Thank you. (<i>tomada 5</i>)
Carl:	You're welcome. (<i>tomada 6</i>)

Existem ainda regras que orquestram a tomada de palavra num diálogo. Os estudos de Sacks et al [Sacks *et al.* , 1974] mostraram que existem pontos no diálogo que são mais susceptíveis a uma mudança de orador. Nesses pontos, a tomada de palavra ocorre de acordo com as seguintes regras:

1. Durante sua fala, o orador atual pode selecionar o próximo orador, por exemplo “Maria, não foi você que teve problemas com o notebook no mês passado?”
2. Se o orador atual não selecionar o próximo orador, então qualquer pessoa pode tomar a palavra. Isso pode acontecer com uma frase neutra, por exemplo: “Não sei como resolver o problema com meu notebook...”
3. Se ninguém tomar a palavra, o orador atual pode continuar.

Em um estudo sobre conversas telefônicas, pesquisadores mostraram que as pessoas adaptam o comportamento de tomada de palavra ao longo do diálogo [Ten Bosch *et al.* , 2004].

2.1.2 Atos de Fala

John L. Austin em seu trabalho intitulado “How to Do Things with Words” [Austin , 1962] identifica a existência de dois tipos de frases. O primeiro deles, historicamente mais estudado, é o constativo, no qual as frases podem ser analisadas como falsas ou verdadeiras, por exemplo: “Esta camisa é branca” ou “O voo para São Paulo está atrasado”. Entretanto, Austin demonstra a existência de um segundo tipo de frases que não podem ser analisadas da mesma forma, como as seguintes:

- “Nomeio este barco *Queen Elizabeth.*”, falado ao quebrar uma garrafa no casco da embarcação.
- “Aposto R\$100 com você que o Brasil ganha a copa de 2014!”.
- “Aceito”, dito durante uma cerimônia de casamento.

Essas frases não descrevem eventos ou coisas. Não faz sentido dizer que elas são falsas ou verdadeiras. Ao proferi-las, o orador está realizando uma ação. Austin propôs chamá-las de performativas. Com essa distinção, Austin foi de encontro ao pensamento anterior no qual *dizer* era simplesmente *atestar algo*.

Mais adiante, no mesmo livro “How to Do Things with Words” [Austin, 1962], que em sua versão brasileira recebeu o título “Quando Dizer é Fazer”, Austin elabora essas divisões de uma maneira um pouco diferente. Após uma tentativa frustrada de listar todos os verbos performativos, ele realiza uma análise em três dimensões, três sentidos para o uso de uma locução: dizer algo **é** fazer algo, **ao** dizer algo fazemos algo, e ainda **por** dizer algo fazemos algo.

Assim, podemos identificar várias forças, vários atos envolvidos quando alguém pronuncia uma frase, nomeadamente:

- Ato locutório – ato de proferir as palavras, formando uma frase com um determinado sentido.
- Ato ilocutório – o que as palavras querem dizer, a intenção do orador em proferir tal frase, ou seja, o efeito pretendido.
- Ato perlocutório – o que foi provocado pela frase dita, o efeito obtido com a frase.

Os conceitos devem ficar mais claros através de um exemplo, vejamos a frase:

Não beba mais!

O ato locutório foi realizado quando o orador proferiu tais palavras para transmitir o seu pensamento. Poderia ter proferido outras, como “Chega de bebida por hoje!”. Note que o sentido mantém-se apesar de nenhuma palavra ser igual.

A frase expressa o ato ilocutório de dar uma ordem ou conselho para que alguém pare de beber. Quem a proferiu pode depois comentar algo do tipo “**ao** falar aquilo, eu estava dando um conselho de amigo”.

O efeito perlocutório de tal frase pode ser o de impedir que a pessoa continue a beber, mas pode acontecer exatamente o contrário e a pessoa beber ainda mais! Um novo comentário do orador pode ser “**por** ter falado aquilo, eu o convenci a parar”.

Apesar do próprio Austin também ter classificado os atos de acordo com sua força ilocutória, a taxonomia criada por John R. Searle tornou-se mais conhecida [Searle, 1975]. Searle propõe a divisão dos atos ilocutórios nas seguintes classes:

- Assertivos – o orador afirma algo e compromete-se com aquilo (afirmar, jurar, concluir)
- Diretivos – o orador tenta fazer com que o ouvinte realize algo (perguntar, ordenar, pedir, aconselhar)

- Compromissivos – o orador compromete-se a agir de alguma forma (prometer, planejar, apostar)
- Expressivos – o orador expressa seus sentimentos (agradecer, desculpar, cumprimentar)
- Declarativos – o orador muda o estado do mundo com suas palavras (despedir alguém, nomear um barco, declarar casamento)

O termo ato de fala geralmente é associado aos atos ilocutórios acima descritos.

2.1.3 Ação Conjunta

Conhecimento comum (*common ground* [Stalnaker, 2002]) pode ser entendido como o conjunto de ideias, fatos, coisas nas quais duas ou mais pessoas acreditam. Durante a troca de atos de fala em um diálogo, existe a possibilidade de reafirmação e ampliação do conhecimento comum entre os envolvidos. Entretanto, para que isso ocorra, é necessário que cada participante do diálogo deixe claro que entendeu a fala do outro, ou seja, é preciso que haja uma correta compreensão (*grounding*) dos atos de fala [Clark e Brennan, 1991].

Uma ação é considerada conjunta quando é necessário que duas ou mais pessoas se coordenem para executá-la com sucesso. [Clark, 1996]. Dessa forma, podemos classificar o diálogo como uma ação conjunta.

Clark e Schaefer dividiram a ação conjunta no diálogo em duas fases: apresentação e aceitação [Clark e Schaefer, 1989]. Em um diálogo que envolva os participantes A e B, a primeira fase acontece quando o orador A apresenta a locução *l* para consideração de B. Nesse momento, A assume que se B der uma evidência *e* ou mais forte, então A pode acreditar que B compreendeu o que ele disse ao falar *l*.

Na fase de aceitação, B aceita a locução *l* ao apresentar evidência *e* de que compreendeu o que A quis dizer com a locução *l*.

Ou seja, na primeira fase o locutor apresenta algo, e, na segunda, recebe uma evidência de que foi compreendido.

No mesmo trabalho, Clark e Schaefer apresentam cinco tipos de evidências de compreensão (*grounding*), abaixo listados do mais fraco para o mais forte:

1. Atenção mantida – B demonstra que continua a prestar atenção e, portanto, segue satisfeito com a apresentação de A.
2. Próxima contribuição relevante – B inicia a próxima contribuição relevante. Por exemplo, uma resposta de B a uma pergunta realizada por A.
3. Reconhecimento – B acena com a cabeça, ou diz algo como “uh huh”, “yeah” ou algo parecido.

4. Demonstração – B demonstra tudo ou parte do que entendeu da fala de A.
5. Exibição – B mostra literalmente toda ou parte da fala de A.

2.1.4 Estrutura Conversacional

Iniciada por Sacks, Schegloff e Jefferson na década de 1960 [Goodwin e Heritage, 1990], a área conhecida como Análise de Conversação (*Conversational Analysis*) tem como objetivo estudar como ocorrem as interações durante uma conversação, como elas estão estruturadas. Esses estudos indicaram que as falas dos participantes tendem a aparecer em pares ao longo do diálogo. Esses pares foram chamados de pares de adjacência (*adjacency pairs*) [Hutchby e Wooffitt, 1998].

Schegloff e Sacks [Schegloff e Sacks, 1973] apontaram uma regra para os pares de adjacência que afirma que após uma produção reconhecível do primeiro do par, esse locutor deve parar e o próximo locutor deve então iniciar e produzir o segundo do par de acordo com o tipo do primeiro.

É fácil identificar a ocorrência desses pares em um diálogo. Quando alguém pergunta algo, é natural que a próxima contribuição seja uma resposta do ouvinte. Da mesma forma uma proposta é seguida por uma aceitação ou recusa, um pedido geralmente é acompanhado por uma concessão, uma saudação por outra, etc.

Também foi notado que além do diálogo ser organizado em pares de adjacência, parece existir uma preferência dentro dos pares. Por exemplo, um convite pode ser seguido de uma aceitação ou recusa, mas a aceitação tem uma preferência em relação a recusa. A análise de conversas identifica diferenças de interação nos dois caminhos. A aceitação tende a ser mais rápida, mais natural, enquanto a recusa tende a ser mais demorada, com hesitações e explicações [Pomerantz, 1984].

2.1.5 Implicatura Conversacional

Grice, em seu trabalho “Lógica e Conversação” [Grice, 1975], cria o termo implicatura conversacional para representar uma subclasse de inferências não convencionais utilizadas em diálogo. Para explicá-las, Grice sugere que as conversas seguem o princípio cooperativo.

Um diálogo não é constituído por uma série de falas aleatórias! Ao contrário, ele é formado por ações conjuntas, que possuem um objetivo ou direção comum. Ao longo do diálogo, podem ocorrer mudanças de direção. Entretanto, a cada etapa algumas serão consideradas inadequadas. Assim, Grice formula o princípio cooperativo que diz “faça sua contribuição tal como é requerida, no momento em que ocorre, pelo objetivo ou direção da conversa na qual você está inserido”.

Dentro desse princípio, Grice propôs a existência de quatro máximas:

- Máxima de Quantidade:

1. Faça sua contribuição tão informativa quanto é requerido (para o atual objetivo).
 2. Não faça sua contribuição mais informativa do que é requerido.
- Máxima de Qualidade: Tente fazer sua contribuição verdadeira.
 1. Não diga o que você acredita ser falso.
 2. Não diga algo sem evidência adequada.
 - Máxima de Relevância: Seja relevante.
 - Máxima de Modo: Seja claro.
 1. Evite obscuridade de expressão.
 2. Evite ambiguidade.
 3. Seja breve (evite prolixidade desnecessária).
 4. Seja ordenado.

2.2 Sistema de Diálogo

Um sistema de diálogo pode ser entendido como um programa capaz de estabelecer um diálogo com um ser humano. Podemos categorizá-los por aplicação, arquitetura interna, modalidade e por diversas outras formas.

Quando os dividimos por aplicação, podemos categorizá-los em: sistemas genéricos, sistemas de viagens, de telecomunicações, de comércio eletrônico, de busca de informação e sistemas robóticos, entre outros.

Por modalidade, eles podem ser classificados quanto ao uso de texto, de fala, de interface gráfica e de gestos. Quando mais de um desses modos são utilizados, diz-se que o sistema é multi-modal.

O ponto central de um sistema de diálogo, portanto, é a comunicação humano-máquina, bem mais simples do que a humano-humano, pelo menos por enquanto! A figura 2.1 mostra o diagrama geral de um sistema de diálogo, apresentando as cinco etapas principais desse processo de comunicação.

2.2.1 Reconhecimento de Fala

A primeira etapa de uma interação homem-máquina é o reconhecimento de fala – Automatic Speech Recognition (ASR) –, que recebe como entrada a voz humana e produz uma sequência de palavras. A dificuldade da tarefa do ASR varia muito em função do tamanho do vocabulário que ele é capaz de entender. Obviamente, é mais fácil entender uma sequência de dígitos do que toda e qualquer frase de uma determinada língua. Também há

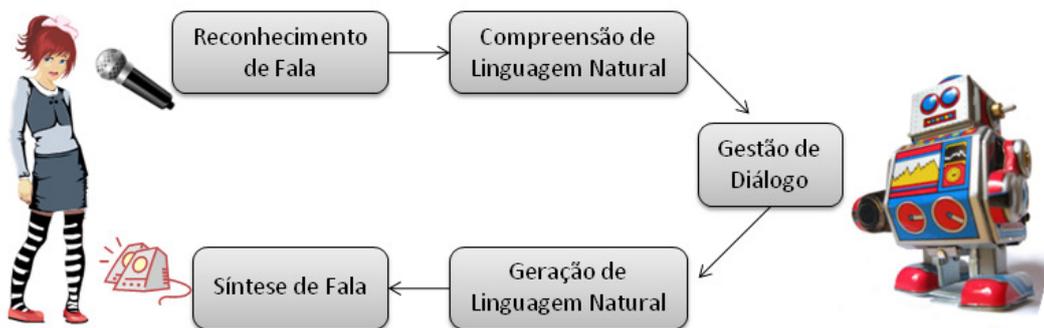


Figura 2.1: Diagrama geral de um sistema de diálogo

diferença entre sistemas treinados para funcionar somente com uma pessoa e outros que são independentes do locutor.

Um bom ASR deve ser rápido, permitindo respostas em tempo real, senão a usabilidade do sistema fica prejudicada. Ninguém gostaria de esperar vários segundos a cada interação.

Vale a pena ressaltar que muitos fatores influenciam o desempenho de um ASR, como por exemplo a existência de ruído no ambiente e o fato de os participantes serem ou não nativos na língua de interação. Diante dessas dificuldades, muitos ASR's atribuem um nível de confiança às frases reconhecidas, indicando, assim, o quão certos estão acerca do que acabou de ser dito.

2.2.2 Compreensão de Linguagem Natural

A próxima fase é a de compreensão de linguagem natural – Natural Language Understanding (NLU) –, cujo objetivo é extrair a semântica a partir da sequência de palavras reconhecida. Para isso, existem vários tipos de NLU. A escolha é feita em função das características do sistema de diálogo.

No caso de sistemas de viagens, uma abordagem comum é a de preenchimento de formulários (*frame-based*), que procura extrair da frase as informações necessárias para uma posterior consulta em base de dados apropriada. Vejamos o formulário apresentado na tabela 2.1. Ele pode ter sido preenchido a partir da pergunta “Quais são os voos disponíveis do Rio de Janeiro para Fortaleza na sexta-feira à noite?”.

Tabela 2.1: Formulário para um sistema de viagens aéreas

Campo	Informação
Origem	Rio de Janeiro
Destino	Fortaleza
Data	Sexta-feira
Horário	Noite

Outros sistemas dividem a tarefa de compreensão em duas partes. Primeiro realizam uma

análise sintática da frase e, a seguir, extraem a semântica usando gramáticas especialmente criadas para isso. Essas gramáticas podem, por exemplo, detectar relações entre entidades presentes na frase.

2.2.3 Gestão de Diálogo

A semântica extraída pelo NLU é enviada ao gestor de diálogo, que possui uma visão geral da estrutura da conversação e deve produzir uma resposta adequada.

É claro que, assim como na compreensão de linguagem natural, existem também vários tipos de gestores de diálogo. Um dos primeiros e mais conhecidos sistemas de diálogo é o ELIZA [Weizenbaum, 1966], que simplesmente procurava por palavras-chave e aplicava uma série de transformações no texto que recebia do usuário, mas não chegava a manter um contexto do diálogo. ELIZA assumia um papel de psicanalista e, apesar de suas limitações, era capaz de gerar diálogos como o mostrado na tabela 2.2.

Tabela 2.2: Exemplo de diálogo gerado pelo sistema ELIZA

Usuário	Os homens são todos iguais.
ELIZA	de que maneira
Usuário	Eles estão sempre nos incomodando.
ELIZA	você pode pensar num exemplo específico
Usuário	Bem, meu namorado me fez vir aqui.
ELIZA	seu namorado te fez vir aqui
Usuário	Ele diz que estou deprimida a maior parte do tempo.
ELIZA	lamento escutar que você está deprimida

Atualmente, a gestão de diálogo mais simples é realizada por máquinas de estados finitos. Um pouco mais de flexibilidade é oferecida pelos gestores baseados em formulários (*frame-based*). Dentre os sistemas avançados de gestão, destacamos os baseados no Estado da Informação (*information state based*), nos quais o gestor trabalha em função do contexto do diálogo, seja para atualizá-lo ou para decidir qual a melhor ação a ser tomada em um determinado instante. Mais detalhes sobre gestão de diálogo podem ser vistos na seção 2.3.

2.2.4 Geração de Linguagem Natural

A etapa de geração pode ser dividida em duas grandes partes, o planejamento de conteúdo e a geração de linguagem propriamente dita, que podem ser vistas como *o que dizer* e *como dizer*.

Na maioria dos sistemas, o planejamento de conteúdo é realizado pelo gestor de diálogo, mais capacitado para decidir se o sistema deve realizar uma pergunta ou responder algo naquele instante.

Para a geração de linguagem, a maneira mais simples de realizá-la é através da instanciação de padrões (*templates*). Por exemplo, em um sistema de viagens poderíamos ter o seguinte modelo de pergunta de confirmação: “Você pretende partir de <CIDADE-ORIGEM>?”, no qual somente a cidade seria inserida na estrutura de frase pré-estabelecida.

No outro modo de gerar linguagem, o conteúdo elaborado pelo gestor de diálogo é expresso por uma semântica e o módulo de geração de linguagem natural – Natural Language Generation (NLG) – deve transformar essa semântica em linguagem natural, tornando o sistema bem mais flexível e apto a gerar qualquer frase coberta por sua representação semântica.

2.2.5 Síntese de Fala

Por último, a etapa de síntese de fala recebe o texto gerado e produz sinal de fala. Aqui, vale a pena ressaltar que estamos falando de *qualquer* texto escrito. Isso porque há sistemas de resposta em voz (*voice response system*) que produzem fala artificial através da concatenação de palavras ou trechos de frases. Entretanto, esses sistemas somente são aplicáveis em cenários com uma estrutura de frase que se repita e estão limitados a um vocabulário da ordem de cem palavras. Podem ser utilizados, por exemplo, em estações ferroviárias, rodoviárias ou aeroportos.

A tarefa de um sintetizador de fala – Text-To-Speech (TTS) – pode parecer simples, mas na realidade é complexa e geralmente subdividida em várias etapas.

Primeiramente, é preciso realizar uma transcrição fonética do texto a ser lido. A seguir, são levados em consideração a entoação e o ritmo desejados. Após tudo isso, o submódulo de processamento digital de sinal transforma a informação simbólica em fala natural [Dutoit, 1997].

2.3 Tipos de Gestão de Diálogo

Não há um consenso na categorização dos sistemas de diálogo quanto à arquitetura interna. McTear os divide em: sistemas baseados em estados finitos (*finite state-based systems*), sistemas baseados em formulários (*frame-based systems*) e sistemas baseados em agentes (*Agent-Based Systems*) [McTear, 2002].

Adotaremos a divisão de Jurafsky e Martin [Jurafsky e Martin, 2008], que consideram a existência de três principais tipos:

1. Sistemas de Estados Finitos (*Finite State Systems*)
2. Sistemas Baseados em Formulários (*Frame-Based Systems*)
3. Sistemas Avançados, que podem ser:

- Baseados em modelos Belief, Desire, Intention (BDI) – podem ser utilizados em sistemas de diálogo baseados em planos.
- Processo de decisão markoviano – sistemas Markov Decision Process (MDP) utilizam probabilidade para definir o comportamento do gestor de diálogo.
- Estado da informação – sistema deste tipo armazenam informação acerca do contexto do diálogo e as decisões do gestor de diálogo são tomadas em função disso.

2.3.1 Sistemas Baseados em Estados Finitos

Neste tipo de sistema, uma máquina de estados finitos representa o diálogo. Isto significa que cada transição de estado deve ser codificada no sistema. A mudança de estado ocorre quando o utilizador fornece a informação pela qual o sistema está a espera. Essa informação é geralmente uma frase curta ou mesmo palavras isoladas.

Por saber exatamente a que questão o utilizador está respondendo a cada momento da interação, é possível ajustar o reconhecimento de fala para um melhor desempenho. Da mesma forma, a compreensão de linguagem natural pode ser beneficiada.

Teoricamente, é possível criar máquinas de estados finitos nas quais em cada estado existam transições que correspondam às possíveis respostas do utilizador. Mas isso significa uma explosão no número de estados que inviabiliza a construção de tais sistemas na prática.

Estes sistemas não oferecem muita liberdade ao utilizador porque as repostas precisam ser dadas na ordem previamente estabelecida, e, além disso, ele não deve responder mais do que lhe foi perguntado. Sistemas que controlam o diálogo desta forma são considerados de iniciativa única (*single initiative*) ou de iniciativa do sistema (*system initiative*).

Outro problema é relacionado com a estratégia de verificação da informação. Aqui, essa verificação é feita através de confirmação explícita, o que pode levar a diálogos muito longos.

A tabela 2.3 mostra um diálogo gravado de uma versão *demo* do sistema *Nuance Automatic Banking* [McTear, 2002]. Como podemos ver em *User2*, quando o utilizador é questionado sobre o valor do pagamento, ele informa o valor e também a data do pagamento. Porém, como o sistema estava a espera somente do valor, a data é solicitada novamente na próxima pergunta (*System3*). Além disso, a informação extra fornecida pelo utilizador em *User2* (*next Monday*), fez com que o sistema reconhecesse um número incorreto para o valor do pagamento, como pode ser visto em *System4*.

Como exemplo de sistema que adota este tipo de gestão de diálogo, podemos citar o robô Jijo-2, que será descrito na seção 2.4.6.2.

2.3.2 Sistemas Baseados em Formulários

Estes sistemas trabalham com formulários (*forms* ou *frames*), os quais possuem campos que precisam de ser preenchidos para que o sistema seja capaz de interagir com uma base de

Tabela 2.3: Exemplo de sistema baseado em estados finitos – diálogo do sistema Nuance Automatic Banking [McTear, 2002]

System0:	What would you like to do? (<i>O que você gostaria de fazer?</i>)
User0:	Pay a bill. (<i>Pagar uma conta.</i>)
System1:	What company would you like to pay? (<i>Qual empresa você gostaria de pagar?</i>)
User1:	Abbey National.
System2:	How much would you like to pay? (<i>Quanto você gostaria de pagar?</i>)
User2:	One hundred pounds next Monday. (<i>Cem libras próxima segunda-feira.</i>)
System3:	What date would you like the payment to be made on? (<i>Em qual data que você gostaria que o pagamento fosse feito?</i>)
User3:	Next Monday. (<i>Próxima segunda-feira.</i>)
System4:	You want to pay Abbey National one hundred pounds and sixty-one pence on April 5th? Is this correct? (<i>Você quer pagar cem libras e sessenta e um centavos para Abbey National no dia 05 de abril? Isso está correto?</i>)
User4:	No. (<i>Não.</i>)
System5:	Would you like to change the company, amount, or date? (<i>Gostaria de mudar a empresa, quantidade ou data?</i>)
User5:	Amount. (<i>Quantidade.</i>)
System6:	How much would you like to pay? (<i>Quanto você gostaria de pagar?</i>)
User6:	One hundred pounds. (<i>Cem libras.</i>)
System7:	You want to pay Abbey National one hundred pounds on April 5th? Is this correct? (<i>Você quer pagar cem libras para Abbey National no dia 05 de abril? Isso está correto?</i>)
User7:	Yes. (<i>Sim.</i>)

dados. Em geral, estes sistemas tem questões associadas aos campos dos formulários. Por exemplo, o campo “*ORIGIN CITY*” pode estar associado à questão “*Where are you leaving from?*”.

Sistemas deste tipo dão mais liberdade para o utilizador se comparados aos sistemas baseados em máquinas de estados finitos. Aqui, o utilizador não está mais limitado a responder estritamente à questão que lhe foi feita, ele pode fornecer outras informações e o sistema é capaz de lidar com isso. O utilizador também pode tomar a iniciativa do diálogo, por isso sistemas baseados em formulário são considerados de iniciativa mista (*mixed initiative*).

Para oferecer essa flexibilidade, o gestor de diálogo precisa manter o controle de quais campos já foram preenchidos e quais ainda precisam de informação. O gestor continua a fazer perguntas ao utilizador até que possua dados suficientes para interagir com a base de dados.

A verificação da informação pode ser feita implicitamente. Isto acontece quando o sistema utiliza uma informação adquirida do utilizador para elaborar uma nova questão. Se a informação não for recusada pelo utilizador, assume-se que ela é verdadeira. Caso contrário, as correções feitas são armazenadas.

Como pode ser visto na tabela 2.4, sistemas baseados em formulários são capazes de produzir diálogos mais fluentes do que sistemas baseados em estados finitos. Este diálogo é do sistema *Philips automatic train timetable information* [Aust *et al.*, 1995]. No exemplo, a primeira frase do sistema (*System0*) termina com uma questão aberta, “*How can I help you?*”. Isto permite que o utilizador fale mais normalmente. A verificação implícita é aplicada em *System2*, no qual o sistema utiliza a informação de partida e chegada previamente fornecida para perguntar a data da viagem. Como o sistema reconheceu mal a data, o utilizador teve que corrigi-la em *User2*. A partir daí, o diálogo continua normalmente.

Este tipo de sistema de diálogo é utilizado no sistema Queen’s Communicator, que será mostrado na seção 2.4.1.2.

2.3.3 Sistemas Baseados em Modelos de Crenças, Desejos e Intenções

Boa parte dos sistemas de diálogo que utilizam modelos de crenças, desejos e intenções – Belief, Desire, Intention (BDI) [Bratman *et al.*, 1991] – aplicam algoritmos de planejamento para a compreensão e gestão do diálogo. Por exemplo, se um agente precisa de uma informação, ele pode elaborar um plano que inclui fazer uma pergunta para obter a informação em falta. De forma análoga, um agente que escuta uma pergunta pode tentar inferir o porquê dela ter sido elaborada.

Aqui, os agentes trabalham de maneira cooperativa: tentam inferir o objetivo do utilizador, antecipar obstáculos e fornecer respostas para alcançar este objetivo. As falas são consideradas ações de um plano.

A figura 2.2 mostra o modelo BDI [McTear, 2002]. Podemos ver que o raciocínio do agente para seleção de objetivos de comunicação envolve suas crenças, incluindo as crenças compartilhadas, seus desejos e obrigações de discurso.

Existem alguns problemas com estes sistemas. Um deles é que para inferir o plano do utilizador, é necessário reconhecer que atos de fala foram realizados em cada frase. Atos de fala mal reconhecidos implicam em um plano de utilizador errado. E mecanismos adicionais são necessários para a recuperação nestes casos.

Traum e Allen abordam as obrigações de discurso em [Traum e Allen, 1994]. Eles argumentam que modelos de discurso que são baseados na análise da intenção do orador e na construção de um plano cooperativo entre os agentes, não explicam porque os agentes ainda têm que responder mesmo quando não há um plano compartilhado ou quando eles não sabem a resposta. Traum e Allen explicam este comportamento fazendo uma diferença entre as intenções do agente e suas obrigações. Estas últimas são as ações que o agente tem que executar devido a algum fato ocorrido. A tabela 2.5 mostra um exemplo de regras de obrigação que um agente pode vir a ter.

Um exemplo de utilização deste tipo de sistema de diálogo é o projeto TRAINS, que será descrito na seção 2.4.2.1.

Tabela 2.4: Exemplo de sistema baseado em formulário – diálogo do sistema Philips Automatic Train Timetable Information [Aust *et al.*, 1995]

System0	Good morning. This is the automatic time-table information service. How can I help you? (<i>Bom dia. Este é o serviço automático de informações de cronograma. Como posso ajudá-lo?</i>)
User0	Good morning. I need a connection from Hamburg to Munich. (<i>Bom dia. Eu preciso de uma conexão de Hamburgo para Munique.</i>)
System1	When would you like to go from Hamburg to Munich? (<i>Quando você gostaria de ir de Hamburgo para Munique?</i>)
User1	Next Friday, er, at eight o'clock. (<i>Próxima sexta-feira, er, às oito horas.</i>)
System2	So you want to travel on Sunday the 25th of September at 8 am? (<i>Então você quer viajar no domingo dia 25 de Setembro, às 8h?</i>)
User2	No, on Friday. (<i>Não, na sexta-feira.</i>)
System3	So you want to travel on Friday the 23rd of September at 8 am? (<i>Então você quer viajar na sexta-feira dia 23 de Setembro, às 8h?</i>)
User3	Exactly. (<i>Exatamente.</i>)
System4	There is the following connection: With Intercity Express 793, departure from Hamburg at 8:02, arrival in Munich at 14:05. Would you like me to repeat the connection? (<i>Há a seguinte conexão: Com Intercity Express 793, saída de Hamburgo às 08:02, chegada em Munique às 14:05. Você gostaria que eu repetisse a conexão?</i>)
User4	No, but I need a train for the return trip on Sunday. (<i>Não, mas eu preciso de um trem para a viagem de volta no domingo.</i>)
System5	At what time would you like to go from Munich to Hamburg on Sunday the 25th of September? (<i>A que horas você gostaria de ir de Munique a Hamburgo, no domingo dia 25 de setembro?</i>)
User5	I have to arrive in Hamburg late in the evening. (<i>Eu tenho que chegar em Hamburgo no final da noite.</i>)
System6	So you want to arrive in Hamburg between 8 pm and 11 pm? (<i>Então você quer chegar em Hamburgo entre 20h e 23h?</i>)
User6	Yes. (<i>Sim.</i>)
System7	There are three connections. This is the first one: With Intercity Express 880, departure from Munich at 15:54, arrival in Hamburg at 21:58. Would you like me to repeat the connection? (<i>Existem três ligações. Esta é a primeira: Com Intercity Express 880, partida de Munique às 15:54, chegada em Hamburgo às 21:58. Você gostaria que eu repetisse a conexão?</i>)
User7	No, thank you. (<i>Não, obrigado.</i>)
System8	Would you like the next connection? (<i>Gostaria da próxima conexão?</i>)
User8	No, thanks, this one is fine. (<i>Não, obrigado, esta está boa.</i>)
System9	Do you need another connection? (<i>Você precisa de outra conexão?</i>)
User9	No. (<i>Não.</i>)
System10	I wish you a pleasant journey. Thank you for calling. Good bye. (<i>Desejo uma viagem agradável. Obrigado pela sua ligação. Adeus.</i>)

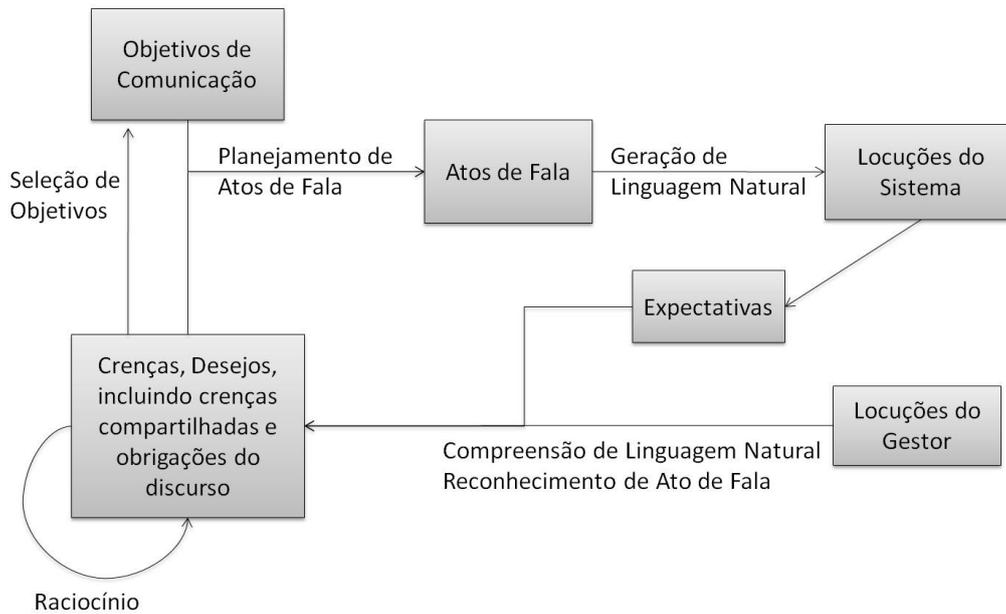


Figura 2.2: Modelo BDI [McTear, 2002]

Tabela 2.5: Exemplo de regras de obrigação [Traum e Allen, 1994]

Origem da Obrigação	Ação de Obrigação
S_1 aceita ou promete A	S_1 realizar A
S_1 requisita A	S_2 processar o pedido: aceitar A ou rejeitar A
S_1 questão Sim/Não acerca de P	S_2 responder se P
S_1 questão aberta $P(x)$	S_2 informar sobre x
Fala não compreendida ou incorreta	Corrigir fala

2.3.4 Processos de Decisão Markovianos

Podemos definir um processo de decisão markoviano (MDP) da seguinte forma [Jurafsky e Martin, 2008]:

- um conjunto de estados E , nos quais o agente pode se encontrar em determinado instante;
- um conjunto de ações A , as quais o agente pode tomar;
- uma função de recompensa $r(a, e)$, a qual o agente recebe por tomar uma ação em determinado estado.

Dado isso, podemos definir uma política P que determina o comportamento do agente, indicando que ações tomar em um dado estado de maneira a receber a melhor recompensa.

Como decidir qual a melhor política a adotar? A forma mais comum de se fazer isto é através do cálculo da recompensa cumulativa esperada, utilizando equação de *Bellman*.

Essa equação diz que a recompensa cumulativa esperada de um par estado/ação é dado pela recompensa imediata de tomar aquela ação somada a utilidade descontada esperada de todos os próximos estados, levando em consideração a probabilidade do agente ir até esses estados. O desconto nas recompensas futuras é utilizado para que o agente se preocupe mais com as recompensas atuais.

Para que possamos utilizar a equação de *Bellman*, é preciso um modelo que diga qual a probabilidade de um par estado/ação (e, a) levar a um estado novo e' . E também uma boa estimativa da recompensa $r(e, a)$. Para isto, é preciso dados etiquetados em quantidade suficiente. Há dois métodos para conseguir esses dados [Jurafsky e Martin, 2008]:

1. Ajustar o número de estados e políticas ao mínimo e então criar um sistema de diálogo que explore o espaço de estados através de conversas aleatórias. As probabilidades podem então ser definidas a partir do *corpus* criado. Isso foi usado por Singh *et al.* [2002] para otimizar a política de diálogo adotada para o sistema *NJFun*, de busca de entretenimento em New Jersey.
2. Construir um utilizador simulado. Este utilizador interage com o sistema inúmeras vezes e o sistema pode aprender as transições de estado e as probabilidades de recompensa a partir do *corpus*. Levin *et al.* [2000] adotam essa abordagem na interação com um sistema de planejamento de viagens. Eles usaram aprendizado supervisionado para estimar o modelo de utilizador, e então, promoveram várias interações do utilizador simulado com o sistema, para descobrir a estratégia ótima de diálogo.

2.3.5 Sistemas Baseados no Estado da Informação

Neste tipo de sistemas, é mantido um contexto do diálogo, chamado de *Estado da Informação*, que identifica o essencial do que ocorreu no diálogo até ao momento e também norteia as decisões do gestor de diálogo.

Um sistema de diálogo baseado no Estado da Informação deve conter os seguintes elementos [Larsson e Traum, 2000; Traum e Larsson, 2003]:

- Descrição do Estado da Informação – uma descrição dos componentes que integram o Estado da Informação.
- Representação formal – identificação formal dos componentes descritos no Estado da Informação.
- Movimentos de diálogo (*dialogue moves*) – que podem ser vistos como eventos, acontecimentos exteriores que provocam a atualização do Estado da Informação.
- Regras de atualização – definem quando e como atualizar o Estado da Informação, geralmente ativadas pelos movimentos de diálogo.

- Estratégia de atualização – determina quais regras aplicar e em que ordem.

Este tipo de sistema de diálogo é a base do sistema TrindiKit [Larsson e Traum, 2000], que será apresentado na seção 2.5.3. O sistema de diálogo desenvolvido para o robô Carl no âmbito desta tese (ver capítulo 4) também se baseia nesta aproximação.

2.3.6 Sistemas Multi-Estratégia

Há propostas para uso de múltiplas estratégias de diálogo no mesmo sistema de diálogo falado – Spoken Dialogue System (SDS). Neste caso, o gestor de diálogo ficaria encarregado de mudar a estratégia de acordo com várias circunstâncias. A ideia é aproveitar o melhor que cada estratégia tem para oferecer e tornar a interação homem-máquina o mais natural possível [Chu *et al.*, 2005]. Este novo gestor de diálogo será integrado ao sistema Queen’s Communicator.

2.4 Exemplos Representativos de Sistemas de Diálogo

Classificados por aplicação, os sistemas de diálogo podem ser agrupados em: sistemas genéricos, sistemas de planejamento de viagens, de telecomunicações, de comércio eletrônico, de procura de informação e sistemas robóticos, entre outros. Após apresentarmos alguns exemplos representativos desses sistemas de diálogo, daremos especial atenção aos sistemas ligados à robótica.

2.4.1 Sistemas Genéricos

Nesta categoria estão aqueles sistemas propostos para uso genérico, portanto facilmente adaptáveis a diferentes cenários.

2.4.1.1 TRIPS

Allen e colaboradores propuseram a construção de um sistema de diálogo genérico, chamado de TRIPS (*The Rochester Interactive Planning System*) [Allen *et al.*, 2000], que pudesse ser utilizado em diferentes domínios com um mínimo de esforço.

Os autores afirmam que a interação humano-computador é feita para a realização de tarefas específicas e chamam de “diálogos práticos” o tipo de conversação em tais interações.

O trabalho baseia-se em duas hipóteses. A primeira afirma que, embora ainda complexa, a competência de conversação exigida em diálogos práticos é bem menor do que a exigida em conversações humanas em geral. A segunda diz que, dentro dos diálogos práticos, a complexidade de interpretação e gestão de diálogo é independente da tarefa a ser realizada.

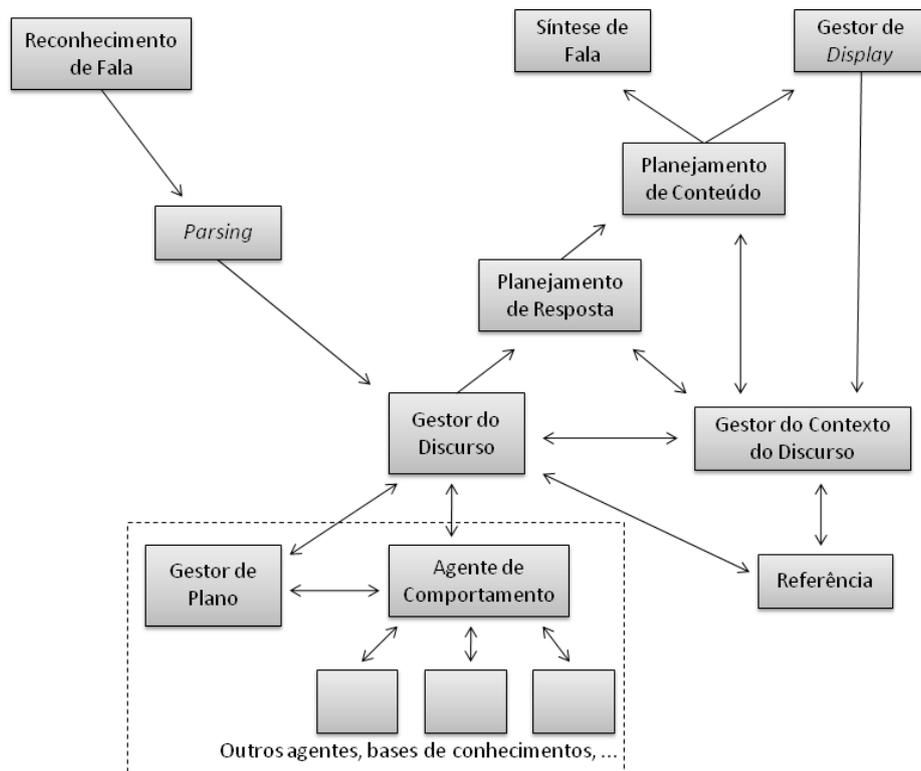


Figura 2.3: TRIPS [Allen *et al.*, 2000]

Com base nestas duas hipóteses, o sistema TRIPS tenta separar os componentes básicos do sistema de diálogo dos relacionados com a tarefa. Com isto, eles pretendem construir uma arquitetura comum para qualquer sistema de diálogo prático.

Eles propõem seis módulos para gestão de diálogo: gestor do contexto do discurso, resolução de referências, reconhecimento de intenções, agente de comportamento, gestor de plano e o planejamento de resposta. Desses módulos, o agente de comportamento e o gestor de plano são os específicos de domínio. Os outros módulos formam um grupo de diálogo genérico (*generic dialogue shell*). Todos os módulos tem acesso a uma hierarquia semântica comum e a um gestor da base de conhecimento global.

A figura 2.3 mostra a comunicação entre os vários módulos. Os módulos específicos do domínio estão marcados por linhas tracejadas.

A separação entre o agente de comportamento e o gestor de diálogo facilita a interação de iniciativa mista. O agente de comportamento pode ter objetivos mais prioritários do que os tratados pelo diálogo em curso e iniciar um novo diálogo para lidar com eles.

O sistema TRIPS define uma interface genérica para o agente de comportamento, a qual permite a construção do resto do sistema de forma independente do domínio.

A arquitetura de software utiliza troca de mensagens para a comunicação entre os componentes. Pode haver comunicação com outros sistemas com o uso de ferramentas como

Open Agent Architecture (OAA) [Martin *et al.*, 1999] ou Common Object Request Broker Architecture (CORBA) [OMG, 1995].

As mensagens trocadas utilizam o protocolo Knowledge Query and Manipulation Language (KQML) [McKay *et al.*, 1994; Labrou e Finin, 1998] ou a Agent Communication Language (ACL) [Dignum e Greaves, 2000]. Ambas possuem uma sintaxe parecida com Lisp e baseiam-se na semântica dos atos de fala.

Como na OAA, existe a figura de um facilitador central, o *TRIPS facilitator*, por onde as mensagens são encaminhadas aos demais componentes do sistema.

2.4.1.2 Queen's Communicator

O Queen's Communicator é um sistema de diálogo reutilizável que adota a metodologia orientada a objetos (implementação em java) [Hanna *et al.*, 2005].

Há dois grupos de agentes peritos implementados como objetos: os agentes de serviço e os de suporte.

Os agentes não fazem pedidos a outros agentes específicos, e sim, sobre um conhecimento desejado. Isto ocorre para reduzir o grau de dependência entre os agentes – o mesmo acontece com a OAA. O agente *DomainSpotter* gerencia as chamadas entre agentes, o mesmo papel do *facilitator* na arquitetura OAA.

O Queen's Communicator utiliza a estratégia de *slot-filling*, possuindo formulários como base.

2.4.2 Sistemas de Planejamento de Viagens

Estes sistemas abordam o planejamento de viagens, num cenário que pode envolver reserva de passagens, hotéis e aluguel de carro. Eles precisam adquirir do usuário uma série de informações como cidade de origem e destino, data de ida e volta, horário de preferência, etc.

2.4.2.1 TRAINS

O projeto TRAINS foi desenvolvido na Universidade de Rochester com a colaboração do NTT Information and Communication System Lab e da Universidade de Edinburgh. O objetivo principal do projeto era a construção de um agente de planejamento de viagens que utiliza conversação (*conversational planning agent*) [Allen, 1994].

O sistema TRAINS amplia os tradicionais sistemas de planejamento para permitir uma iniciativa mista (*mixed-initiative planning*), a qual ocorre quando dois agentes estão envolvidos na construção de um plano. Isso é feito com o suporte do planejamento colaborativo e incremental.

Este sistema trabalha com duas linguagens de representação de conhecimento: Episodic Logic (EL) e Event-Based Temporal Logic (EBTL). A primeira é utilizada no módulo de

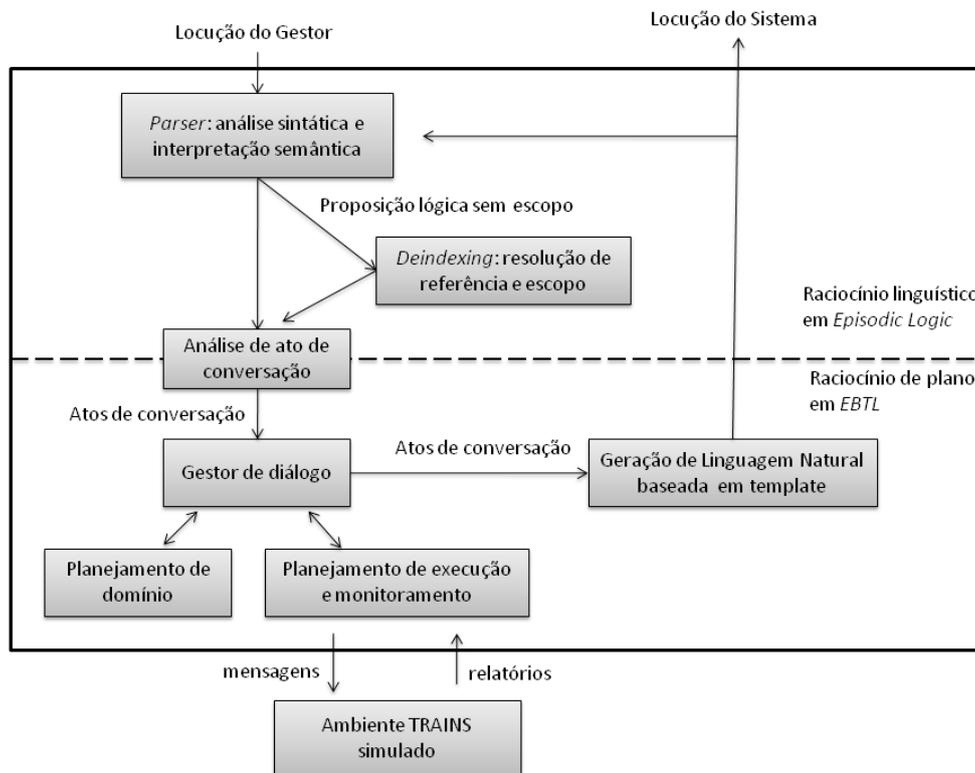


Figura 2.4: Arquitetura do sistema TRAINS-93 [Allen, 1994]

linguística e a segunda no de planejamento. Embora EL seja mais ampla e complexa do que EBTL, Allen [1994] alega que, na maioria das vezes, nenhuma informação é perdida na tradução de uma para outra.

A figura 2.4 mostra a arquitetura do sistema TRAINS, versão de 1993 [Allen, 1994]. Entre os módulos apresentados, destacamos: o resolução de referências (*deindexing*), o gestor de diálogo e o planejamento de domínio.

O módulo de desindexação (*deindexing*) é responsável por interpretar as expressões de indexação, encontrando possíveis hipóteses para as referências e resolvendo ambiguidades de escopo. Este módulo baseia-se na Conversation Representation Theory (CRT), uma extensão da Discourse Representation Theory (DRT). A CRT é capaz de lidar com ambiguidades semânticas e de permitir a representação de informação pragmática presente nos diálogos.

O sistema utiliza a teoria dos atos de conversação com quatro níveis de interação:

- Tomada de palavra (*turn-taking*): atos de fala que mantém, liberam ou tomam a vez no diálogo.
- Conhecimento comum: atos de fala que lidam com conhecimento comum ou compartilhado, por exemplo, reconhecimento (*acknowledgement*) ou reparar (*repair*).
- Atos de fala nucleares: atos ilocutórios encontrados nas abordagens tradicionais, por

exemplo, informar(*inform*).

- Argumentação: caracterizam as relações entre frases.

O gestor de diálogo mantém um modelo mental interno e o contexto do discurso que inclui a posse da palavra (*turn*), o plano corrente e informação sobre conhecimento comum. O modelo mental possui: crenças e propostas sobre o domínio; um conjunto de objetivos do discurso; um conjunto de atos de fala que o sistema irá realizar; e um conjunto de obrigações de discurso pendentes.

Para decidir o que fazer a seguir, o gestor de diálogo utiliza a seguinte prioridade:

1. Obrigações de discurso resultantes de atos de fala anteriores.
2. Obrigação geral de não interromper a vez do outro.
3. Atos de fala planejados.
4. Obrigação geral relativa a conhecimento comum do conteúdo, por exemplo: confirmar ou pedir confirmação de falas, ou consertar suas próprias falas que tenham sido ignoradas até então.
5. Objetivos do discurso da negociação do plano de domínio, por exemplo: aceitar ou rejeitar propostas feitas pelo coordenador, propor ou pedir aceitação de propostas feitas pelo sistema.
6. Objetivos do discurso em alto nível, por exemplo: chamar o módulo de planejamento para expandir o plano de domínio, que fornecerá conteúdo para que o sistema elabore novas propostas.

2.4.2.2 DARPA Communicator

Projeto da Defense Advanced Research Projects Agency (DARPA) com o objetivo de promover o desenvolvimento de sistemas de diálogo falado em vários institutos de pesquisa. O cenário proposto envolveu reserva de passagens aéreas, hotéis e envio de e-mail de confirmação [Walker *et al.*, 2002].

Os participante do projeto foram: AT&T, BBN Technologies, Carnegie Mellon University, University of Colorado, IBM, Lucent Bell Labs, MIT, MITRE e SRI International.

Todos os sistemas utilizaram uma arquitetura comum, a Galaxy II do MIT. Esta arquitetura encontra-se descrita na subsecção 2.5.1.

2.4.3 Sistemas de Telecomunicações

Sistemas de Telecomunicações são desenvolvidos para automatizar as centrais de atendimento, os *call centers*. Esses sistemas tem como objetivo identificar o motivo da ligação do cliente, atendê-lo, se possível, ou redireciná-lo para o grupo de atendimento mais adequado.

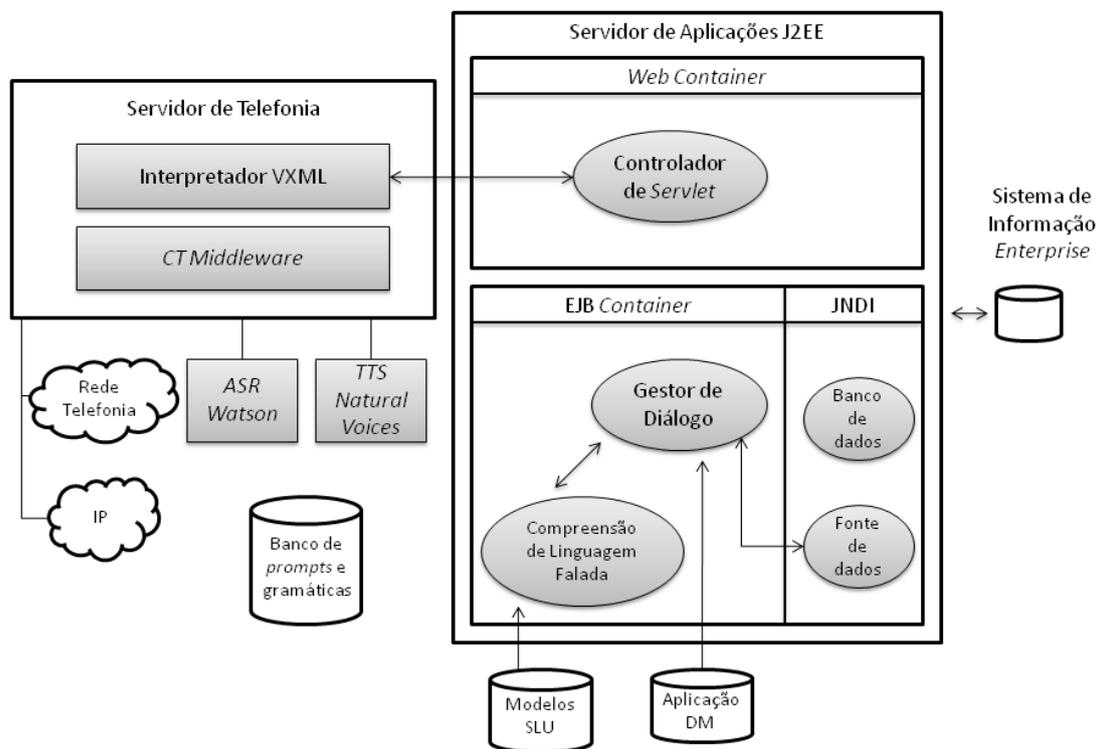


Figura 2.5: Arquitetura do *Spoken Dialogue System* da AT&T [Fabrizio e Lewis, 2004]

2.4.3.1 Florence

Florence é uma plataforma de gestão de diálogo desenvolvida pela AT&T [Fabrizio e Lewis, 2004]. Esta plataforma foi criada para dar suporte a construção de SDS com diferentes estratégias de diálogo, não apenas com foco no redirecionamento de chamadas (*call routing*) ou na interação baseada em planos.

A figura 2.5 mostra a arquitetura SDS da AT&T, a qual possui três partes: o cliente – browser VoiceXML integrado no servidor de telefonia; o servidor de aplicação; e o sistema de informação da empresa (*Enterprise information system*). O gestor de diálogo gera páginas VoiceXML, ativa a gramática do ASR e pede ao módulo de compreensão de linguagem falada a representação semântica das falas.

A saída do módulo de compreensão de linguagem falada pode ser tanto em Natural Language Semantic Markup Language (NLSML) ou Extensible MultiModal Annotation (EMMA), ambas representações semânticas baseadas em XML.

A figura 2.6 mostra a arquitetura Florence. Ela faz uso de uma pilha para manter o controle dos diálogos e subdiálogos. A informação de cada diálogo é armazenada em um objeto chamado controlador de fluxo (*flow controller*, ou FC), o qual inclui o contexto local. A aplicação como um todo mantém um contexto global. A informação pode passar do contexto local para o global e também de um diálogo para o seu subdiálogo. Quando um diálogo cria

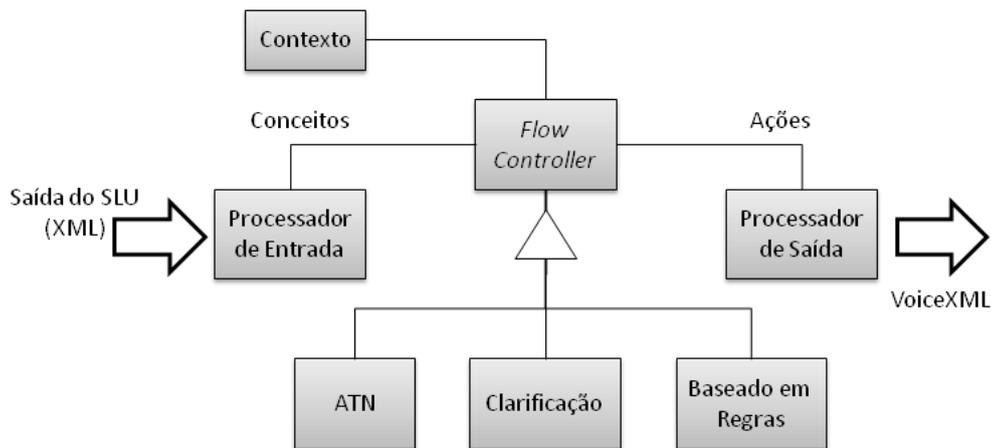


Figura 2.6: Arquitetura Florence [Fabrizio e Lewis, 2004]

um subdiálogo, este assume o controle. O sistema comunica-se com o utilizador através de ações.

Várias instâncias do diálogo podem coexistir na pilha. Quando uma entrada chega, ela vai para o contexto de cada *FC* na pilha. Mas apenas o *FC* com o controle do diálogo pode adicionar ações ao processador de saída e mudar variáveis no contexto. Ele pode retornar o controle ao diálogo anterior na pilha, como também pode criar um subdiálogo.

A aplicação termina quando o diálogo de nível mais alto retorna o controle. O *FC* também pode declarar terminada a sua tomada de palavra. Neste caso, o sistema retorna uma estrutura com as ações que foram disparadas e o diálogo é interrompido até que o utilizador fale novamente.

Cada diálogo pode ser guiado por uma estratégia diferente, como rede de transições expandida – Augmented Transition Network (ATN) [Woods, 1969] –, clarificação ou baseada em regras.

As redes de transições expandidas (ATN) são constituídas por nós e arcos direcionais que os unem. Tal como nos autômatos de estados finitos, existem nós iniciais e finais. Da mesma forma, a transição entre os nós depende da condição associada ao arco. Entretanto, as ATN permitem recursividade – os arcos podem estar associados a sub-redes – e o armazenamento de informações em variáveis, assim como a realização de testes adicionais em seus arcos.

A estratégia de diálogo mais utilizada no Florence é a ATN, que atua na entrada e no contexto local para controlar o fluxo de interação. Sua estrutura contém estados, transições, condições e um conjunto de variáveis. Os estados podem executar operações em variáveis do contexto e, opcionalmente, invocar outras redes ou sub-diálogos, retornar o controle para um diálogo anterior, ou declarar terminada a tomada de palavra. A transição de um estado para outro é escolhida depois do estado atual terminar sua execução. A escolha é feita de acordo com as condições das transições de saída. As ações associadas são disparadas e o controle

é passado para o estado a seguir. Esta estratégia é apropriada para redirecionamento de chamadas e outros diálogos simples.

A estratégia de clarificação utiliza uma árvore de categorização hierárquica para ajudar o utilizador na escolha do tópico. A árvore possui condições que descrevem as categorias e tópicos, assim como perguntas para informações mais específicas. Aqui o utilizador pode dar mais informações do que o sistema pediu e também tomar a iniciativa no diálogo.

2.4.4 Sistemas de Comércio Eletrónico

Os sistemas desta categoria visam facilitar e expandir as transações eletrônicas através do uso de diálogo, simulando a atuação de um vendedor humano.

2.4.4.1 ADVICE

O sistema ADVICE foi desenvolvido pela Universidade de Madrid para servir como um assistente virtual no comércio eletrônico [García-Serrano *et al.*, 2004].

A arquitetura do sistema é constituída por três agentes: o de interface, o de interação e o de inteligência. O gestor de diálogo encontra-se no agente de interação.

Os autores baseiam-se na teoria dos atos de fala para definir uma estrutura semântica, que é utilizada como linguagem interna pelo sistema.

As tarefas do agente de interação incluem: interpretar a intenção do utilizador, expressa pelos atos de fala; e identificar o que é preciso ser realizado e como. Para isso o agente é dividido em três componentes:

- Modelo de sessão (*session model*) – contém informação estática do contexto do diálogo.
- Gestor de diálogo – controla a conversação através da definição do estado atual do diálogo. Faz isto levando em consideração a estratégia de diálogo e a intenção ou objetivo de comunicação que precisa ser realizado.
- Gerador de discurso (*discourse maker*) – gera a estrutura semântica que descreve o que precisa ser dito a seguir pelo agente de interface.

2.4.5 Sistemas de Procura de Informação

Estes sistemas são desenvolvidos com o objetivo de auxiliar o usuário na tarefa de encontrar resposta a determinadas questões. Eles adicionam o diálogo como uma nova forma de interação.

2.4.5.1 IMIX

Este projeto tem como objetivo transformar um sistema de resposta a perguntas num SDS de busca de informação que utiliza interação multimodal e diálogo de iniciativa mista. [Akker *et al.*, 2005]

O sistema IMIX adota a abordagem de Estado da Informação e uma arquitetura baseada em agentes.

O gestor de diálogo possui duas funções principais: reconhecimento dos atos de diálogo do utilizador com suas respectivas atualizações de contexto e a geração de atos de diálogo de acordo com o novo contexto.

O sistema utiliza uma ontologia dos atos de diálogo baseada na teoria da interpretação dinâmica (*Dynamic Interpretation Theory*).

2.4.6 Sistemas Robóticos

Aqui, mostraremos os sistemas de diálogo desenvolvidos especialmente para uso em robôs. Eles precisam ser bastante flexíveis para lidar com diferentes tipos de tarefas, desde comandos de navegação até pedidos de informação.

2.4.6.1 ASIMO

ASIMO é um robô de serviço com conversação (*conversational service robot*). Nakano *et al.* [Nakano *et al.*, 2005] propõem um modelo em duas camadas para o módulo de planejamento de diálogo e comportamento de robôs deste tipo. Multi-Expert-based Behavior and Dialogue Planning (MEBDP), foi o nome dado para esse módulo.

A figura 2.7 mostra a arquitetura em duas camadas utilizada pelo ASIMO. Podemos ver que o módulo MEBDP encontra-se na camada deliberativa, recebendo como entrada o resultado do reconhecimento de fala e gestos, assim como outros dados do módulo de integração de informações multi-modal.

Os autores indicam quatro características essenciais para o módulo de planejamento de diálogo e comportamento:

1. Diálogo orientado a tarefa: no qual o robô precisa entender os pedidos humanos e dar informações através de diálogo.
2. Diálogo de iniciativa mista.
3. Troca dinâmica de domínio: mesmo tendo iniciado o diálogo em um domínio, o robô precisa ser capaz de trocá-lo quando assim solicitado.
4. Controle de interrupção: mesmo executando outras tarefas, o robô precisa atender a pedidos de interrupção.

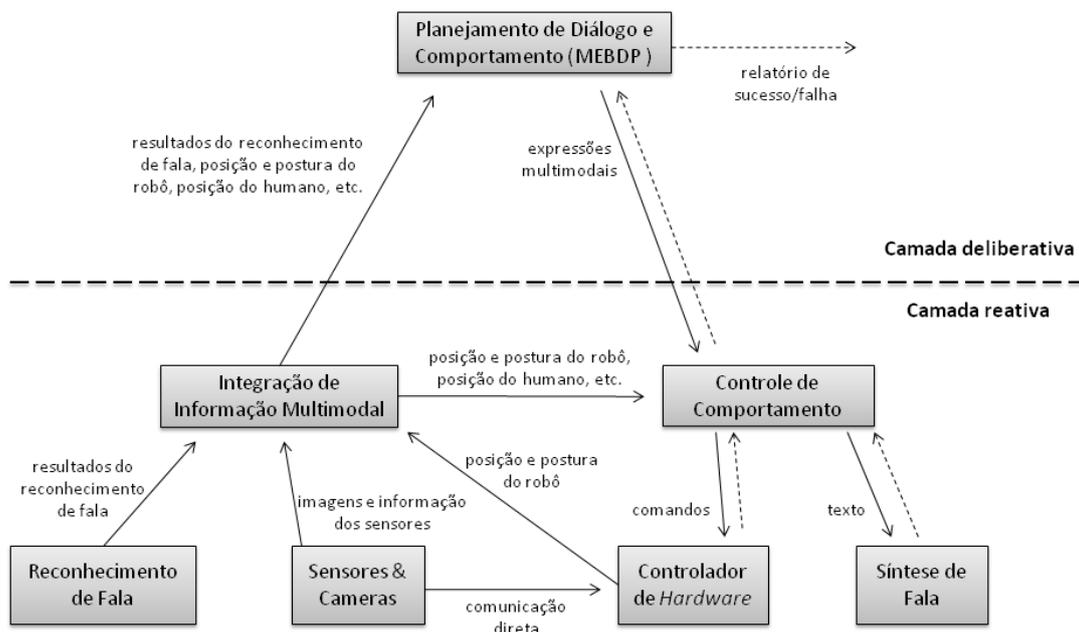


Figura 2.7: Arquitetura desenvolvida para o robô ASIMO [Nakano *et al.*, 2005]

O módulo MEBDP também se divide em duas camadas: a superior é responsável pelo planejamento e divisão da tarefa em subtarefas; a inferior, executa as subtarefas utilizando peritos adequados. Vale a pena ressaltar que ambas encontram-se na camada deliberativa da arquitetura geral.

Há quatro peritos no modelo proposto:

1. Perito na escuta do pedido.
2. Perito no fornecimento da informação.
3. Perito no planejamento da ação física.
4. Perito no diálogo de obtenção de informação.

A figura 2.8 mostra os módulos e como eles se comunicam no modelo MEBDP. Cada perito possui um estado interno, chamado Estado da Informação, e quatro métodos principais:

1. *Understand*: atualiza o Estado da Informação de acordo com o reconhecimento de fala.
2. *Select next action*: devolve uma ação abstrata baseado no atual Estado da Informação. Se considerar a subtarefa completa, devolve *finish*.
3. *Detect-interruption*: determina se a fala anterior é uma interrupção à subtarefa atual.
4. *Handle-interruption*: devolve a ação a ser executada após uma interrupção ser detectada.

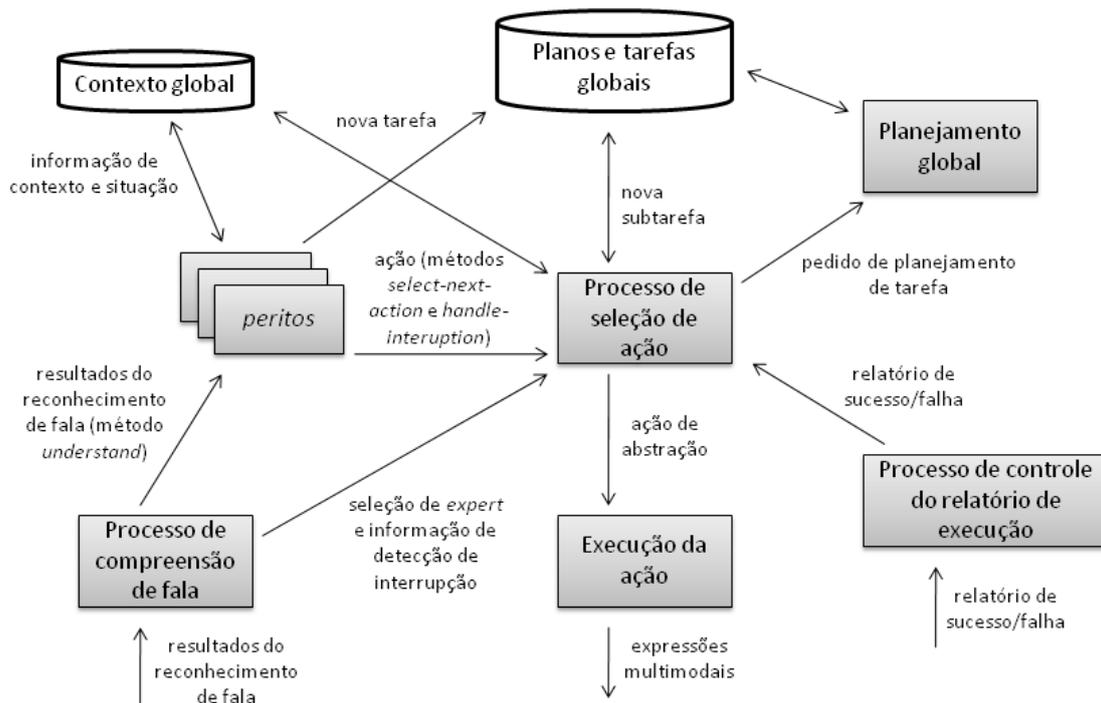


Figura 2.8: Estrutura interna do módulo MEBDP [Nakano *et al.*, 2005]

O módulo *understanding* está sempre em execução e quando recebe um reconhecimento de fala ou gesto, envia o mesmo para todos os peritos através do método *understand*. Após isto, seleciona o perito que melhor se adequa ao novo Estado da Informação.

2.4.6.2 Jijo-2

Jijo-2 é um robô de serviço para escritório com capacidades de comunicação e aprendizagem sobre o ambiente [Asoh *et al.*, 2001]. Entre suas habilidades, destacamos:

- detecção da origem do som;
- detecção da face do utilizador;
- acesso a base de dados, tanto para leitura como para registo de novas localizações;
- envio de e-mail;
- condução da pessoa até um gabinete em um escritório.

O sistema permite que frases com informação subentendida sejam utilizadas. Por exemplo, no decorrer de um diálogo, é perguntado “Onde está mr. Asho?” e após isso, “Dr. Hara?”. O sistema entende a pergunta como “Onde está Dr. Hara?”. Este tipo de construção é muito comum em japonês, língua para qual o sistema de diálogo foi projetado.

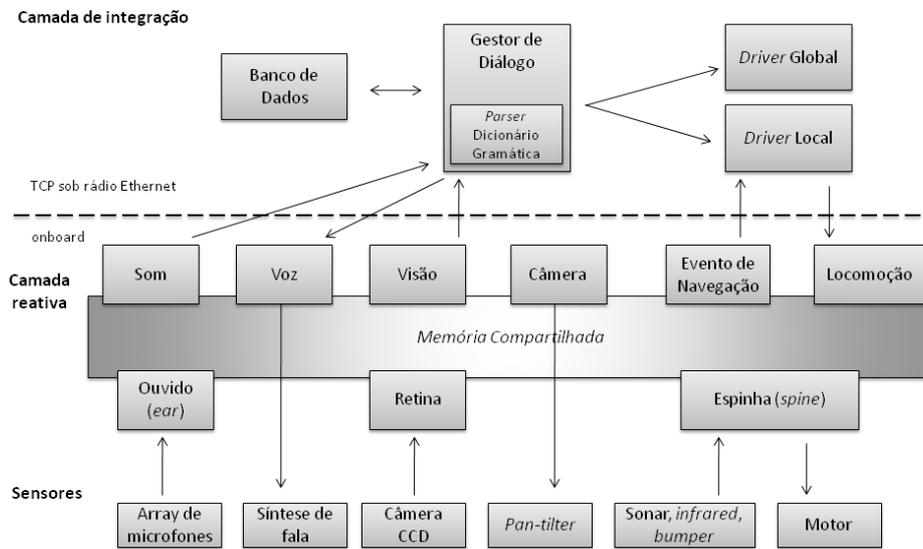


Figura 2.9: Módulos no sistema do robô Jijo-2 [Asoh *et al.*, 2001]

A arquitetura do Jijo-2 é mostrada na figura 2.9. A camada de integração (*integrator layer*) foi implementada em linguagem EusLisp (Lisp orientado a objetos) e é executada numa máquina remota, as outras camadas foram implementadas em linguagem **c** e são executadas no robô. Os módulos comunicam-se através do protocolo TCP/IP e a gestão é baseada em eventos.

O processamento de fala inclui as seguintes etapas:

1. Localização do som e redução de ruído através de processamento de sinal.
2. Extração da semântica (japonês).
3. Interpretação da frase utilizando uma *task frame* e todo o conhecimento do sistema.
4. Gestão do diálogo e do comportamento do robô com o uso de padrões de resposta pré-definidos.

Como mostra a figura 2.10, o diálogo do sistema é controlado por uma máquina de estados finitos. A resposta dada pelo robô varia de acordo com o estado no qual ele se encontra. Cada tarefa que o robô conhece é associada a um formulário cujos campos representam a informação que é necessária para a execução da mesma. Quando uma determinada tarefa é solicitada, o gestor de diálogo tenta preencher os campos do formulário associado.

Encontram-se preparadas cinco diferentes tipos de tarefas: consulta a base de dados, atualização de base de dados, identificação de pessoas, navegação e chamada de pessoas.

O contexto do diálogo é guardado através de uma lista presente no *estado de atenção*, como podemos ver na figura 2.10. Esta lista enumera as entidades referenciadas no discurso.

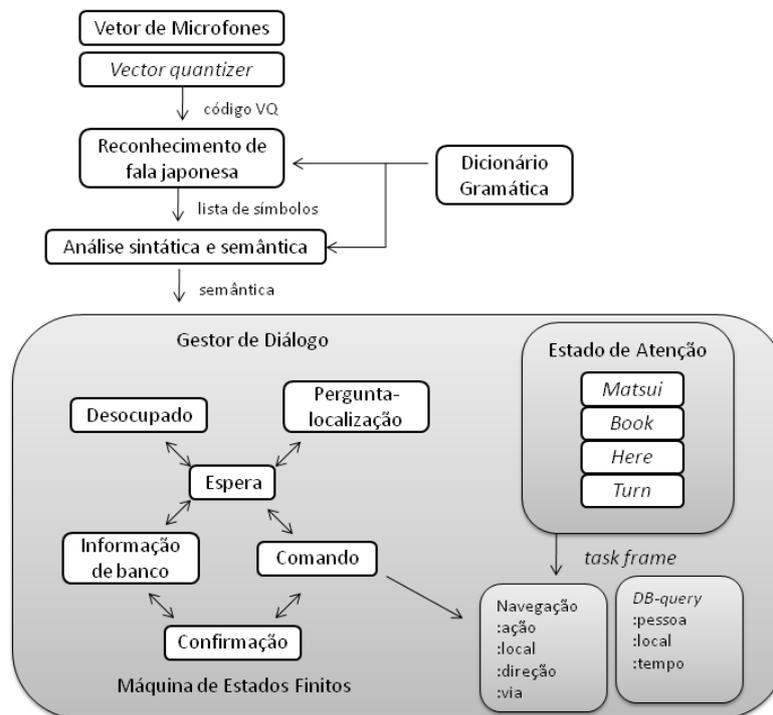


Figura 2.10: Robô Jijo: máquina de estados finito para controle do processo de diálogo [Asoh *et al.*, 2001]

Asoh *et al.* [2001] afirmam que o maior problema do sistema diálogo é o fato da máquina de estados ser construída de forma *ad hoc* e muito dependente da tarefa. Outro problema é o da representação semântica a partir de falas ser também informal, o que torna difícil estender o sistema para atender a outras tarefas.

2.4.6.3 BIRON e ToBI

BIRON (*Bielefeld Robot companION*), ou o robô companheiro de Bielefeld, Alemanha, foi projetado tendo em vista o cenário de uso pessoal em casa, abordando tarefas como a apresentação de ambientes ao robô [Haasch *et al.*, 2004].

ToBI (*Team of Bielefeld*) é um robô desenvolvido pela Universidade de Bielefeld para participar da edição de 2009 da RoboCup@Home, cujo objetivo é incentivar a pesquisa em robótica em atividades domésticas [Wachsmuth *et al.*, 2009].

Esse robô, que é o sucessor do BIRON, tem sua arquitetura baseada na plataforma *GuiaBot* da *Mobile Robots Inc.* com algumas customizações e adições de sensores, como mostra a figura 2.11.

A arquitetura computacional é baseada no conceito de memória ativa, na qual os componentes do sistema, além de terem acesso a informações, podem ser notificados em determinadas mudanças ou inserções de dados. A figura 2.12 mostra as operações permitidas na arquitetura.



Figura 2.11: Robô ToBI e seus componentes: câmera, microfone, braço e sensor laser [Wachsmuth *et al.*, 2009]

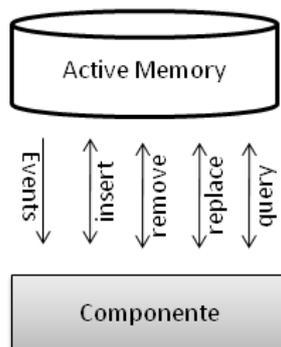


Figura 2.12: Robô ToBI – operações que manipulam a memória ativa [Wachsmuth *et al.*, 2009]

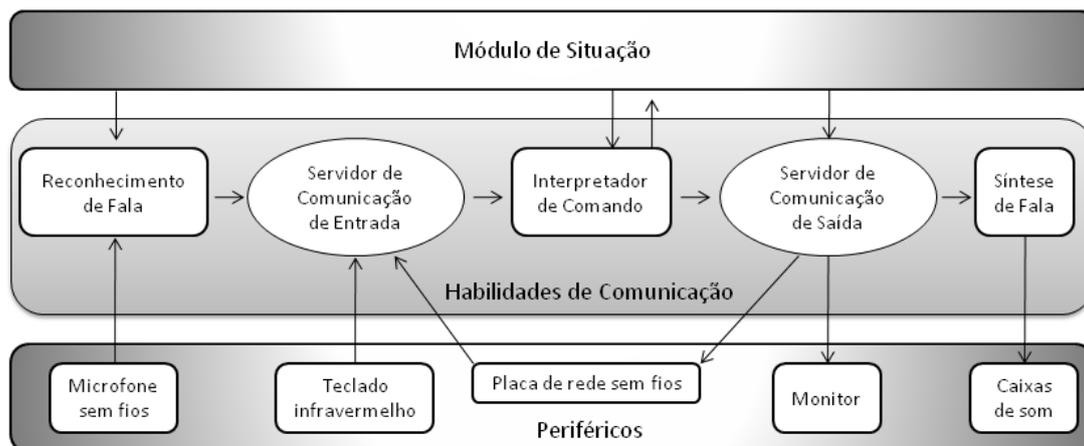


Figura 2.13: HERMES – Interface Homem-Máquina: observe que o módulo de situação interfere no reconhecimento, interpretação de comandos e alimenta a comunicação de saída [Bischoff e Graefe, 2002]

Os robôs BIRON e ToBI compartilham o mesmo sistema de diálogo. Nele, as contribuições no diálogo são representadas em uma estrutura chamada unidade de interação (*interaction unit*), composta por uma camada de intenção, que possui os objetivos de comunicação, e uma camada de conversação, que por sua vez é subdividida em dois módulos geradores, um verbal e outro não-verbal [Li *et al.*, 2006]. Dessa forma, a questão da multimodalidade também é suportada pelo sistema.

2.4.6.4 HERMES

HERMES é um robô humanoide que entende e fala inglês, francês e alemão. Possui a capacidade de comunicação multimodal e foi testado como guia de museu durante um período de seis meses [Bischoff e Graefe, 2002, 2003].

Seu sistema de comunicação baseia-se em linguagem natural, tanto para receber informações como para fornecê-las. Além da fala, o usuário pode interagir com o robô através de um teclado ou por email. Do mesmo modo, o robô pode se comunicar por fala, email ou mensagens exibidas em um monitor. A figura 2.13 mostra um diagrama da interface homem-máquina do robô HERMES. O interpretador de comandos é constituído por um parser, e por analisadores léxico, sintático e semântico.

Para melhorar o desempenho do reconhecedor de fala, o módulo de situação altera as regras gramaticais ativas de acordo com o contexto de momento. Por exemplo, quando o robô pede uma clarificação, são utilizadas regras que permitem identificar “*yes*” ou “*no*”, além das regras permanentes. Além do contexto de clarificação, existem vários outros, como navegação, entretenimento, serviço e manipulação.

Durante um diálogo, o nível de experiência do usuário é identificado e a quantidade de infor-

mação transmitida pelo robô é alterado em função disso. Usuário novatos recebem instruções detalhadas sobre como interagir com o robô, assim como explicações sobre o comportamento e ações que o próprio robô irá executar.

2.4.6.5 MARVIN e ARTOS

MARVIN (*Mobile Autonomous Robot Vehicle for Indoor Navigation*) serve como uma plataforma de teste para exploração autônoma de ambientes fechados. ARTOS (*Autonomous Robot for Transport and Service*) foi otimizado em tamanho e consumo de energia para funcionar em ambiente doméstico, num projeto de assistência a pessoas idosas [Koch *et al.*, 2008].

A abordagem do grupo é a de utilizar robôs como o ponto focal de interação com os demais dispositivos presentes em um ambiente. Os autores alegam que, principalmente para pessoas idosas, o robô atuaria como um parceiro de interação e que seria bem mais conveniente do que falar com interruptores de luz!

O sistema de diálogo desses robôs é centrado em um componente chamado motor de fala, que recebe uma descrição de diálogo do próprio robô ou de outras aplicações. A partir daí, o diálogo pode ser ativado e o motor de fala se encarrega de chamar as aplicações registradas, passando variáveis de estado que tenha sido alteradas na interação.

A descrição do diálogo é baseada em XML e contém elementos para controlar a entrada e a geração de fala. Também define condições e valores de variáveis para guiar o fluxo do diálogo. Ao ser registrado, o diálogo é mapeado para uma máquina de estados.

2.4.6.6 Carl

O robô Carl faz parte de várias revisões da literatura na área [Johnson e Agah, 2009; Koch *et al.*, 2008; Loutfi *et al.*, 2008] e por isso não o excluiremos desta breve listagem. Ele foi desenvolvido dentro de um projeto da Universidade de Aveiro que visava estudar a integração das quatro dimensões do problema de construir um robô inteligente: comunicação, percepção & ação, raciocínio e aprendizagem [Seabra Lopes e Teixeira, 2000]. Mais informações sobre o robô podem ser obtidas na seção 1.2 e no capítulo 3 e sua nova gestão de diálogo é apresentada no capítulo 4.

2.5 Ferramentas para Desenvolvimento de Sistemas e Gestores de Diálogo

O desenvolvimento de sistemas de diálogo pode ser facilitado através de diversas formas, desde interfaces gráficas que auxiliem o analista na elaboração do código até bibliotecas de estratégias de diálogo e modelos de domínio que possam ser reutilizadas em vários sistemas de diálogo.

Outro caminho para simplificar a criação de tais sistemas seria o estabelecimento de um padrão de interface de interação falada. Dessa forma, aprender a interagir com um segundo sistema de diálogo seria mais fácil porque os comandos comuns seriam os mesmos do primeiro sistema.

Descrevemos agora as seguintes arquiteturas e ferramentas para o desenvolvimento de sistemas e gestores de diálogo: Galaxy II, ARIADNE, TrindiKit, DIPPER e RavenClaw/Olympus. Também apresentamos uma proposta de interface padrão para interação homem-máquina por fala, a Speech Graffiti.

2.5.1 Galaxy II

Galaxy II [Seneff *et al.*, 1998] foi a arquitetura escolhida para servir de base para os sistemas que participaram do programa DARPA Communicator (mais detalhes na subseção 2.4.2.2).

A estratégia de controle adotada foi a de regras sequenciais.

Foram definidos servidores para ASR, NLU, NLG e síntese de fala. A interação entre eles é feita através de um concentrador (*hub*) com uma linguagem interpretada própria.

Através dessa linguagem, é possível definir os servidores, máquina e porta, assim como as operações e programas permitidos por eles. Os programas contêm as regras, que consistem em condições e ações que devem ser executadas.

Os servidores fornecidos comunicam-se entre si através de formulários semânticos (*semantic frames*). Para aproveitá-los, é preciso que os novos servidores entendam esta semântica.

Além dos servidores citados, a arquitetura Galaxy II também fornece um modo de depuração para o concentrador no qual é executada uma regra de cada vez, auxiliando assim, a identificação de eventuais erros na programação.

2.5.2 ARIADNE

Proposta de arquitetura para desenvolvimento rápido de sistemas de diálogo falado [Dencke, 2002], focada na separação entre os algoritmos de processamento de diálogo e os aspectos específicos de um domínio. Dessa forma, torna possível a reutilização dos módulos que são independentes de domínio.

A arquitetura ARIADNE é dividida em três níveis:

- Máquina abstrata de diálogo
- Camada de padrões de interação
- Camada de controle do diálogo

O nível mais baixo da arquitetura, a máquina abstrata de diálogo, é responsável pela análise sintática, histórico do discurso (representado em uma árvore), descrição dos objeti-

vos do diálogo, representação semântica (estruturas de características tipadas – *typed feature structures*) e pela geração de linguagem natural, que permite escolha de padrões.

Na camada de padrões de interação, a classificação do estado do diálogo é feita através das seguintes características:

1. *CurrentQuality* – qualidade a locução atual, depende do nível de confiança do reconhecedor de fala, da análise sintática e da representação semântica.
2. *OverallQuality* – valor cumulativo de *CurrentQuality*, usado para detectar falhas no diálogo.
3. *CurrentSpeechAct* – indica o valor do ato de fala da locução atual.
4. *Reference* – indica a necessidade de consulta a base de dados para resolução de referência no discurso.
5. *ReferenceExpressions* – indica se as expressões de referência puderam ser resolvidas. Caso seja possível, mostra se a referência é única ou não.
6. *Intention* – informa quantas descrições de objetivos de diálogo são compatíveis com a informação no discurso. Se houver apenas uma, indica se toda a informação necessária para invocar o serviço associado está presente.

Há quatro padrões de interação:

1. Questão - busca informação do utilizador para adicionar ao discurso
2. Desfazer - remove informação do discurso.
3. Correção - remove e adiciona informação do discurso.
4. Estado - altera o fluxo do diálogo.

Por último, a camada de controle do diálogo instancia os padrões de interação e mantém o estado do diálogo, sendo independente de domínio.

Para comprovar a eficácia da proposta da arquitetura ARIADNE, de separação entre os algoritmos de processamento de diálogo e os aspectos específicos de um domínio, foi conduzido um experimento no qual os participantes deveriam construir um sistema de diálogo escolhendo o domínio e linguagem de sua preferência [Denecke, 2002]. Os autores mostram que o uso da arquitetura permitiu um rápido desenvolvimento dos sistemas.

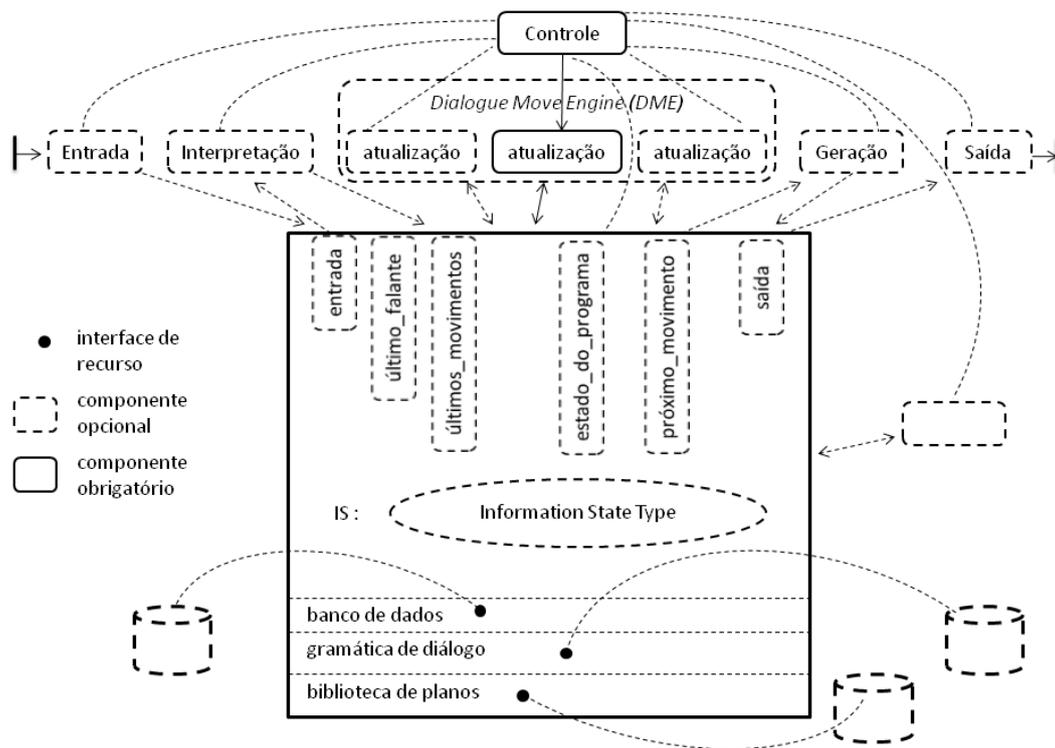


Figura 2.14: Arquitetura do TrindiKit – destaque para o Estado da Informação e módulo DME [Larsson e Traum, 2000]

2.5.3 TrindiKit

TrindiKit é uma ferramenta de construção de sistemas de diálogo baseada na abordagem de Estado da Informação (ver secção 2.3.5). De acordo com esta abordagem, uma das principais funções a ser implementada é a de atualização do Estado da Informação através dos chamados *dialogue moves*. Por isto, esta ferramenta é classificada como uma *Dialogue Move Engine (DME) Toolkit* [Larsson e Traum, 2000].

A figura 2.14 mostra a arquitetura do TrindiKit. Podemos ver o módulo DME, assim como o Estado da Informação (IS - *Information State*) e os vários módulos que compõem um sistema de diálogo.

Para facilitar a construção de um sistema de diálogo, o TrindiKit fornece módulos padrão de entrada, interpretação, geração e saída, além de métodos de conversão de itens e de inspeção do IS.

2.5.4 DIPPER

A ferramenta DIPPER, desenvolvido a partir do TrindiKit, é uma coleção de agentes para prototipagem de sistemas de diálogo falado [Bos *et al.*, 2003a]. Os agentes utilizam OAA e lidam com reconhecimento de fala, geração de linguagem natural e gestão de diálogo. Baseia-se

na abordagem de Estado da Informação [Traum e Larsson, 2003] para modelagem de diálogo.

Utiliza a teoria de representação do discurso (*Discourse Representation Theory - DRT*) e possui resolução de ambiguidades.

Para inferência, duas técnicas são utilizadas: prova de teoremas com *Synergetic Prover Augmenting Superposition with Sorts (SPASS)* [Weidenbach, 1999] e construção de modelos com *MACE* [McCune, 1998].

Além da série de agentes OAA fornecidos, a grande diferença em relação ao TrindiKit é a nova forma de atualizar o Estado da Informação, que passa a ser realizada simplesmente através da declaração de regras, sem a necessidade de utilizar uma linguagem específica para definir o algoritmo de controle de atualização.

2.5.5 Speech Graffiti

Tomko et al. [Tomko *et al.*, 2005] propõem uma interface padrão para interação homem-máquina por fala. A ideia surgiu de dois outros paradigmas de interação: *Graphical User Interface* (GUI) dos Macintosh e do sistema de escrita Graffiti para PDA.

As GUI possuem um conjunto básico de interação (duplo clique, arrastar, etc.), uma vez aprendidos, podem ser utilizados em qualquer sistema. Para sistemas de diálogo falado, existiriam comandos padrões, por exemplo: *options* sempre iria informar sobre o que o sistema pode falar.

Para utilizar o alfabeto Graffiti nos PDA, o utilizador tem que adaptar sua escrita, mas isto é compensado pela melhoria no reconhecimento de escrita. O mesmo pode ser usado em sistemas de diálogo falado, ou seja, pode-se restringir o que o utilizador pode falar, de maneira a aumentar o desempenho do sistema.

A ferramenta *Speech Graffiti* fornece mecanismos para realizar *interações universais*, ações básicas utilizadas praticamente em todas as interfaces faladas, facilitando assim, o desenvolvimento de sistemas de diálogo.

2.5.6 RavenClaw/Olympus

RavenClaw [Bohus e Rudnicky, 2009] é uma plataforma para gestão de diálogo, sucessora da arquitetura AGENDA, que foi usada no *CMU Communicator* [Rudnicky *et al.*, 1999].

Tal como no TRIPS, os componentes específicos da tarefa ficam separados dos componentes de diálogo em geral. Com isso, a ferramenta pretende facilitar a criação de sistemas de diálogo ao oferecer um conjunto de funcionalidades comuns a sistema deste tipo, como controle de erros, temporização e tomada da palavra.

Dessa forma, a arquitetura do RavenClaw é dividida em duas partes:

1. Especificação de tarefa do diálogo – onde estão os aspectos específicos do domínio. Consiste num plano hierárquico para controlar a interação.

2. Motor do diálogo – onde estão os aspectos independentes de domínio. Utiliza uma pilha para manter a estrutura do diálogo e uma lista para controlar o que o sistema espera ouvir do usuário em um dado instante.

Além do gestor de diálogo, é preciso uma série de outros componentes para implementar um sistema de diálogo. Olympus [Bohus *et al.*, 2007] provê uma infraestrutura completa para esse fim, utilizando o RavenClaw como gestor e ferramentas para reconhecimento de fala, análise sintática, análise de confiança, geração de linguagem e síntese de fala.

Alguns sistemas já foram desenvolvidos com base na plataforma Olympus, como:

- RoomLine – sistema telefônico para reserva de salas na Universidade de Carnegie Mellon.
- “Let’s Go!” – sistema de informação de linhas de ônibus para a cidade de Pittsburgh.
- LARRI (*Language Based Retrieval of Repair Information*) – sistema multi-modal para auxílio no trabalho de mecânicos especializados em aeronaves F/A-18.
- TeamTalk – sistema para controle de um grupo de robôs simultaneamente.

2.6 Avaliação de Sistemas de Diálogo

Dybkje et al. discutem sobre avaliação em alguns projetos de sistemas de diálogo [Dybkje et al., 2004]. O trabalho tenta identificar as principais avaliações técnicas e testes de usabilidade utilizados nesses projetos.

Para os reconhecedores de fala, os critérios incluem:

- Taxa de erro de palavra (Word Error Rate (WER)) – indica o quanto o reconhecedor errou em relação ao que seria a transcrição correta. Pode ser definida como:

$$WER = 100 \times \frac{Inseridos + Trocados + Apagados}{Total\ de\ Palavras\ no\ Trecho\ Transcrito\ Corretamente}$$

- Taxa de erro de locução (Sentence Error Rate (SER)) – indica quantas locução tiveram pelo menos um erro em relação ao total de locuções.
- Cobertura de vocabulário – porcentagem do discurso falado que é possível cobrir a partir de um determinado conjunto léxico.
- Perplexidade – utilizado para avaliar modelos de linguagem. Quanto maior for a probabilidade que um modelo atribuir aos dados de teste, melhor ele será. Essa probabilidade é inversamente proporcional a perplexidade.
- Desempenho em tempo real – capacidade de resposta rápida do reconhecedor que permita interação com humanos.

É crescente o uso de novas métricas, como precisão de conceito e taxa de erro de conceito – as quais utilizam *conceito* ao invés de palavras para determinar a eficiência do sistema. Podemos entender *conceito* como as unidades semânticas da frase [Boros *et al.*, 1996]. Por exemplo, no domínio de sistemas de viagens a frase “*Quero ir do Porto a Lisboa*” possui as unidades semânticas: *cidadeOrigem:Porto* e *cidadeDestino:Lisboa*.

Em relação à síntese de fala, a percepção do usuário continua a ser o principal indicador de desempenho.

Quando o objetivo é avaliar todo o sistema, vários critérios são utilizados, como:

- Sucesso de transação – mostra a taxa entre as transações que terminaram com sucesso e o total de transações.
- Tomadas de palavra ou tempo para realização de tarefa – indica quantas tomadas de palavras ou quanto tempo foi necessário para completar uma determinada tarefa.
- Taxa de correção de palavra – indica a quantidade de locuções do usuário ou do sistema que foram utilizadas para corrigir erros, dividido pelo total de locuções.

Existem grandes desafios em definir técnicas de avaliação para sistemas multimodais. São precisas métricas para avaliar reconhedores, interpretadores e geradores de gesto e expressões faciais, assim como uma diversidade de outros componentes de entrada e saída.

Sobre usabilidade, Dybkjea *et al.* [2004] destacam a dificuldade existente em medi-la, mesmo em sistemas unimodais. Afirmam que um sistema com boa usabilidade deve satisfazer as necessidades do usuário. Estas vão além de funcionalidades apropriadas, pois o sistema deve ser fácil de ser entendido e de interagir com ele. Como boa parte das medidas de usabilidade é subjetiva, a forma mais comum de avaliar esta característica é através do preenchimento de questionários ao final da interação com o sistema de diálogo.

PARADISE (PARAdigm for DIalogue System Evaluation) é uma plataforma para avaliação de sistemas de diálogo [Walker *et al.*, 1997]. Os autores afirmam ser possível comparar diversas estratégias de diálogo, calcular o desempenho tanto de subdiálogos como diálogos inteiros e também comparar agentes que desempenham tarefas diferentes através de uma normalização na complexidade das tarefas.

A medida de desempenho do PARADISE é uma função que leva em consideração tanto o cumprimento da tarefa como o custo do diálogo. O PARADISE foi escolhido para avaliar o desempenho dos sistemas que participaram do projeto DARPA Communicator (ver secção 2.4.2.2).

2.7 Atualização do Estado da Arte

Identificamos a necessidade de realizar uma atualização do estado da arte para acompanhar a evolução da pesquisa em sistemas de diálogo utilizados na interação homem-robô. Para isso,

Tabela 2.6: Atualização do estado arte em sistemas de diálogo utilizados na interação homem-robô

Robô/Sistema	Tipo	Iniciativa	Avaliação	Aprendizado
TeamTalk	baseado em planos	mista	n/a	não
ASIMO árbitro	máquina de estados finitos	sistema	70% de tarefas completadas	não
ToBI	baseado em padrões de interação genéricos	mista	não	sim
PHOPE	baseado em exemplos	n/a	90% com 0% WER; 79% com 30% de WER	não

foi efetuada uma busca na ferramenta *Publish or Perish*² por artigos publicados entre 2008 e 2012, na área de engenharia, com as palavras-chave “dialog system”, “dialog manager” e “human robot”. A ferramenta encontrou um total de 61 artigos. Na tabela 2.6, encontra-se uma síntese das características dos sistemas/robôs identificados.

O TeamTalk [Rudnicky *et al.*, 2010], baseado em RavenClaw [Bohus e Rudnicky, 2009], é um sistema de diálogo que possibilita a interação entre uma pessoa e um grupo de robôs simultaneamente. Não foi encontrada avaliação de desempenho para o TeamTalk e a sua descrição não menciona aprendizagem. Nakadai *et al.* [2008] descrevem a implementação de um árbitro para o jogo “rock-paper-scissors” que utiliza o robô Honda ASIMO como plataforma. Nessa implementação, o sistema de diálogo utilizado é baseado em máquina de estados finitos, a iniciativa de diálogo é do robô, e não envolve aprendizagem. Na avaliação, 70% das tarefas foram realizadas. O robô ToBI [Wachsmuth *et al.*, 2010] foi criado para participar da competição RoboCup@Home, possui um sistema de diálogo de iniciativa mista baseado em padrões de interação genéricos, que foi desenvolvido com o uso do Pamini [Peltason e Wrede, 2010]. ToBI possui capacidade de aprendizagem, mas não encontramos resultados de avaliação de desempenho. O robô PHOPE [Oh *et al.*, 2009] foi projetado para servir de guia em ambientes fechados, como um prédio. Esse robô utiliza um gestor de diálogo baseado em exemplos, no qual a decisão de que ação tomar em determinado ponto do diálogo é tomada após uma consulta a uma base de dados de exemplos de diálogos. Na avaliação, um utilizador simulado foi utilizado para gerar 1000 diálogos simulados. O resultado dessa avaliação foi apresentado em função da taxa de erro de reconhecimento de fala (WER): para reconhecimentos perfeitos, 90% das tarefas foram completadas; para uma taxa de erro de 30%, o percentual de tarefas foi de 79%.

O sistema de diálogo desenvolvido por Bohus e Horvitz [2009] não foi incluído na ta-

²Harzing, A.W. (2007) Publish or Perish, disponível em <http://www.harzing.com/pop.htm>.

bela 2.6 por não ter sido utilizado em um robô, e sim, em agentes conversacionais. Este trabalho descreve os desafios de construir um sistema de diálogo capaz de interagir com vários interlocutores simultaneamente. Em um dos sistemas desenvolvidos na plataforma, o agente conversacional realiza tarefas típicas de um recepcionista. Não há detalhes de avaliação neste artigo e a aprendizagem é relatada como trabalho futuro.

O estudo conduzido por Peltason e Wrede [2011] ficou fora da tabela 2.6 por ser um comparativo entre diferentes ferramentas de construção de sistemas de diálogo: Ravenclaw, Collagen/Disco, Dipper (Open Agent Architecture), PaMini. Nesse trabalho, que utilizou o robô Curious como plataforma, as autoras concluem que nenhuma das ferramentas possui a solução para todos os problemas e indicam que o foco da pesquisa futura na área será em processos semânticos, como raciocínio ou na comunicação interna das ferramentas.

Capítulo 3

Arquitetura Computacional e Sistema de Diálogo

Neste capítulo, a arquitetura computacional e os principais componentes são mostrados. Primeiramente, a seção 3.1.1 apresenta a arquitetura computacional anteriormente adotada, que precisou de ser substituída por uma que facilitasse a adição e remoção de módulos.

A seguir, os componentes principais são descritos mais detalhadamente. A seção 3.2 apresenta a forma como se dá a compreensão de fala no robô, assim como a linguagem de representação utilizada nesse contexto.

Um sistema de aquisição e gestão de conhecimento foi desenvolvido para gerir a informação adquirida ao longo do diálogo. A seção 3.3 descreve esse sistema que permite a aquisição de informações contraditórias.

Por fim, a seção 3.4 descreve o novo agente NLG (Natural Language Generation).

3.1 Arquitetura Computacional

3.1.1 Arquitetura Computacional Anterior

A arquitetura computacional desenvolvida para o Carl em 2001-2002 baseava-se num conjunto de processos que corriam em ambiente Linux. Na versão usada em 2001 na competição *Hors d'Oeuvres Anyone?*, da American Association for Artificial Intelligence (AAAI), todos os processos do Carl corriam no computador da plataforma móvel Pioneer 2-DX. Em 2002, foi acrescentado um computador pessoal portátil e novos processos computacionais para lidar com o ecrã tátil desse computador, interface gráfica e cara animada. Os vários módulos da arquitetura comunicavam entre si através de uma biblioteca muito simples baseada em *sockets*. A figura 3.1 mostra a arquitetura encontrada quando iniciamos o nosso trabalho.

O processamento de fala era efetuado por um conjunto de processos em Linux, baseado nas ferramentas Nuance para reconhecimento de fala e no sistema Festival para síntese de voz.

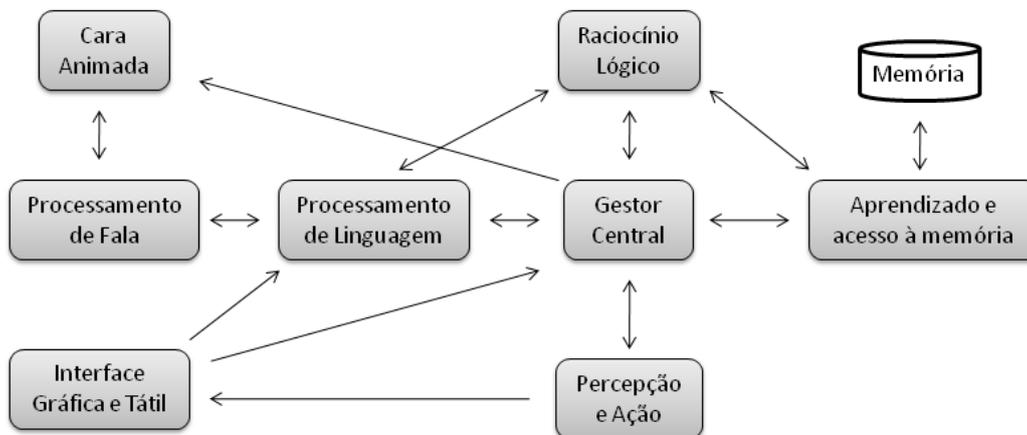


Figura 3.1: Arquitetura computacional anterior do Carl [Seabra Lopes, 2002]

O processamento de linguagem incluía a análise e a geração de fala. A análise de linguagem natural, projetada para ser robusta em relação a problemas de reconhecimento de fala, foi uma das áreas que recebeu mais atenção dentro do projeto. (seção 3.2). A geração de linguagem natural era baseada em padrões.

A interação via ecrã tátil era controlada pelo módulo Interface Gráfica e Tátil (GTI, de *Graphical and Touch Interface*). Linguagem escrita podia ser capturada através de um teclado virtual no ecrã. Uma cara animada apresentava emoções apropriadas.

Raciocínio de alto nível, incluindo inferência indutiva e dedutiva, era executado sobre a máquina de inferência Prolog¹. Outro módulo da arquitetura era responsável pela aquisição de conhecimento, dando ao Carl a capacidade de aprendizagem.

Outro processo Linux tratava da percepção geral e ação, incluindo a navegação. Era baseado em Saphira/ARIA, arquitetura para controle dos robôs *Pioneer*, a qual fornece bibliotecas com elementos básicos de ação e interpretação de sensores (ARIA - *ActivMedia Robotics Interface for Application*) [ActivMedia, 2002], assim como a possibilidade de realizar tarefas mais complexas (Saphira) [Konolige *et al.*, 1997; Konolige, 2001]. Toda a computação era feita no robô. Os processos de percepção e ação eram executadas no computador da plataforma *Pioneer*, enquanto os demais processos corriam no computador portátil acrescentado ao Carl em 2002.

O gestor central era um sistema baseado em eventos, os quais desencadeavam transições de estado. Atividades aparentemente diferentes como gestão de diálogo e navegação, foram integradas numa arquitetura comum. O gestor central era em grande parte implementada em Prolog, mas com partes em C/C++, tanto por razões de eficiência como para acesso às facilidades de comunicação inter-processos do Linux.

O gestor central era essencialmente uma função de transição de estados (figura 3.2) descrita

¹SWI Prolog, uma versão livre com uma boa interface C/C++, é utilizado.



Figura 3.2: Módulo de gestão central na anterior arquitetura– um processo baseado em eventos

como um conjunto de cláusulas Prolog. Cada cláusula, especificando uma transição, tinha uma cabeça com o seguinte formato:

```
state_transition(
    State,
    Events,
    Restrictions,
    SpeechAct,
    Actions,
    NewState
)
```

Onde, *State* era o estado actual. *Events* era uma lista de eventos que causavam a transição de *State* para *NewState*, desde que as restrições identificadas em *Restrictions* fossem satisfeitas. Estes eventos podiam ser de fala, navegação, interface tátil, tempo ou eventos do corpo do robô. *SpeechAct*, quando não vazio, indicava uma mensagem verbal que o robô deveria emitir naquela transição. Note que esta estratégia de gestão de diálogo deixava a iniciativa para o utilizador. Finalmente, *Actions* era uma lista de outras ações que o robô deveria executar. Estas poderiam ser de navegação, de controle da interface gráfica e da cara animada, mas também poderiam ser de atualização interna.

No total, o espaço de estados do Carl incluía 15 estados e 40 transições de estado. A figura 3.3 mostra dois exemplos de transição de estado. O primeiro é uma transição de um estado de movimento normal (*explore*, *wander* ou *stay*) para um estado no qual a principal atividade do robô era ir para a área de abastecimento (*refil area*). O evento que habilitava esta transição era a ausência de biscoitos na bandeja do robô (*no_biscuits*). Esta atividade, evento e transição de estado foram introduzidas para a competição da AAI.

A segunda transição de estados na figura 3.3 é uma transição para o mesmo estado, o de interação (*interacting*). O evento que provocava a transição era a recepção de uma instância do ato de fala *tell*. O robô imediatamente parava, confirmava a recepção e memorizava a informação. O instante de ocorrência desse evento era armazenado, assim o robô poderia mais

```

state_transition(
    State,
    [no_biscuits],
    ( member(State,[explore,wander,stay]) ),
    nothing,
    [ retract_all_times,assert_go_to_refill_time,
      execute_task(go_to_refill_area) ],
    going_to_refill
).

state_transition(
    interacting,
    [ heard(tell(Phrase)) ],
    true, % no restrictions
    acknowledge_told_fact(Phrase),
    [ execute_motion(stop),retract_all_times,
      memorize_told_fact(Phrase),
      assert_last_heard_time ],
    interacting
).

```

Figura 3.3: Exemplos de transições de estado na anterior arquitetura

tarde perceber que a interação terminou mesmo sem um explícito “good bye” do utilizador humano.

3.1.2 Nova Arquitetura – Open Agent Architecture (OAA)

Como referido na seção 3.1, na arquitetura computacional anteriormente adotada no robô, os vários processos comunicavam-se através de uma biblioteca *ad-hoc* desenvolvida pela equipe de investigação da época. Uma mudança fez-se necessária porque o projeto precisava de uma arquitetura que facilitasse a adição e remoção de módulos. Além disso, havia incompatibilidades entre a biblioteca e novas versões do sistema operativo Linux.

Optamos por adotar a Open Agent Architecture (OAA) [Martin *et al.* , 1999], usada por exemplo em [Bos *et al.* , 2003b]. Essa arquitetura foi escolhida por ser aberta, flexível, extensível e distribuída, requisitos considerados essenciais pela equipe.

A OAA é uma comunidade multi-agente que permite interações flexíveis entre os mesmos. Há um agente central, chamado facilitador (*facilitator*), responsável pela coordenação.

Nesta arquitetura há três tipos de agentes:

1. O solicitador, que encaminha ao facilitador um pedido de execução de um serviço.
2. O fornecedor, que declara suas capacidades ou serviços ao facilitador.

3. O facilitador, que mantém uma lista das capacidades dos agentes fornecedores disponíveis.

Quando o facilitador recebe um pedido, ele procura o agente fornecedor adequado a resolvê-lo, delega-lhe a tarefa e devolve o resultado ao solicitador. Também pode haver uma comunicação direta entre solicitador e fornecedor, como será apresentado mais adiante nesta seção.

Deve-se notar que o mesmo agente pode atuar como fornecedor e solicitador no período de sua execução.

3.1.2.1 Comunicação entre os Agentes

A comunicação entre os agentes faz uso da Interagent Communication Language (ICL), que é baseada numa extensão de Prolog e possui a mesma sintaxe. A unidade básica é o *termo* e pode representar:

- Variáveis - iniciam com letra maiúscula ou com o caractere ‘_’. Exemplos: A, Y67, _any.
- Números - escritos como na maioria das linguagens. Exemplos: 77, 82.75.
- Átomos - iniciam com letra minúscula ou estão entre aspas simples (‘’). Exemplos: house, echo2000, ‘Stephen Patrick’.
- Estruturas - possuem um *functor* e um ou mais argumentos. O exemplo a seguir possui *functor* **book** e 3 argumentos: book(1, ‘Moby Dick’, shelf(34)).
- Listas - as listas podem ser representadas por parênteses retos, com os elementos, de qualquer tipo, separados por vírgulas. A lista vazia é representada por []. Exemplos: [9,8,7], [car(‘Panda’), year(1979)].

3.1.2.2 Desenvolvimento de Agentes OAA

Basicamente, para participar de uma comunidade OAA, um agente precisa:

1. Incluir a biblioteca OAA².
2. Conectar-se e registrar-se com o agente facilitador.
3. Fornecer uma lista de suas capacidades (pode estar vazia).
4. Registrar funções que lidem com suas capacidades.

Várias linguagens podem ser utilizadas pelo agente, tais como: C, Java, Prolog (Sicstus e Quintus) e Lisp.

²Esta biblioteca pode ser obtida no site www.ai.sri.com/oaa.

3.1.2.3 Agentes OAA na Arquitetura do Carl

A mudança de arquitetura computacional no robô Carl implicou em transformar todos os módulos existentes em agentes. Nesse processo, alguns módulos deram origem a vários agentes, outros fundiram-se em um só. O processamento de linguagem natural foi distribuído por três agentes, um dedicado à análise sintática (NLA, de *Natural Language Analysis*), outro dedicado à análise semântica (NLU, de *Natural Language Understanding*) e outro ainda dedicado à geração de linguagem natural (NLG, de *Natural Language Generation*). Por outro lado, o raciocínio lógico, o aprendizado e acesso à memória passaram a ser um módulo interno do agente gestor, o *manager*.

A figura 3.4 mostra a comunicação lógica³ entre os agentes. Para o compreensão de uma fala do usuário e geração de uma resposta adequada do robô, os agentes tipicamente envolvidos seriam o ASR, NLA, NLU, Manager, NLG e Synthesis. Observe que há um fluxo de comunicação entre eles. As mensagens trocadas pelos agentes são mostradas na tabela 3.1.

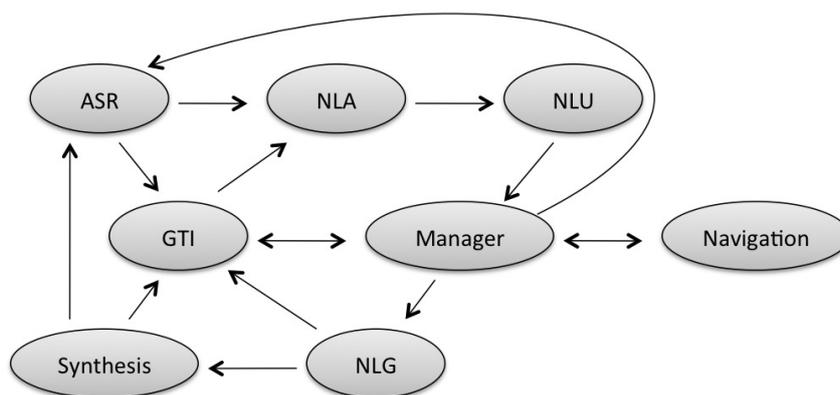


Figura 3.4: Agentes OAA no robô Carl – Comunicação Lógica

Apresenta-se em seguida uma descrição dos diversos agentes OAA incluídos na arquitetura computacional do robô Carl:

Agente de Reconhecimento Automático de Fala — O agente de reconhecimento automático de fala (ASR, de *Automated Speech Recognition*) funciona como o sistema de audição do robô. É responsável por traduzir o sinal acústico em texto, no nosso caso, em inglês. Seu desempenho pode variar em função de vários fatores, como ruído de fundo, qualidade do equipamento utilizado, quantidade de palavras compreensíveis, além da pronúncia de quem estiver interagindo com ele.

O agente Automatic Speech Recognition (ASR) descodifica a entrada de voz em sequência de palavras. Estas sequências são então processadas pelos agentes NLA e NLU, que precisam

³Comunicação lógica porque foi abstraído da figura a presença do agente facilitador, responsável por gerenciar as trocas de mensagens.

Tabela 3.1: Mensagens trocadas pelos agentes

De	Para	Mensagem	Comentário
ASR	NLA GTI	recognized (Sentence, RecConf)	Frase reconhecida pelo ASR e respectiva confiança.
NLA	NLU	parsed_sentence(AnalysisType, Sentence, RecConf, SyntacticAnalysis)	Análise sintática realizada pelo agente NLA. Contém a análise sintática (<i>SyntacticAnalysis</i>), seu tipo (<i>AnalysisType</i>), o nível de confiança do ASR (<i>RecConf</i>) e a frase reconhecida (<i>Sentence</i>).
NLU	Manager	nlu_event(AnalysisType, Sentence, RecConf, Semantics)	Contém o tipo de análise sintática realizada no agente NLA para uma frase reconhecida pelo agente ASR (<i>Sentence</i>), <i>AnalysisType</i> , o nível de confiança do reconhecimento, <i>RecConf</i> , e o resultado da análise semântica realizada pelo agente NLU, <i>Semantics</i> .
Manager	NLG	nlg(SpeechAct, SpeechActType, Main, Semantics, Sentence)	Pedido de geração de uma frase a partir de uma semântica. <i>SpeechAct</i> é o ato de fala da frase a ser gerada; <i>SpeechActType</i> é usado para especificar alguns atos de fala. <i>Main</i> identifica a principal entidade da pergunta. <i>Semantics</i> contém as relações semânticas que caracterizam a frase a ser gerada. <i>Sentence</i> armazena a frase gerada.
Manager	Navigation	navigation (Command)	Envio de comando de navegação a ser executado.
Manager	ASR	asr (Command, Value)	Execução de um comando do ASR.
Manager	GTI	face (Command)	Envio de comando para a cara animada.
Manager	GTI	interface (Command)	Envio de comando para o agente GTI.
Navigation	Manager	navigation_event (Event, EventData)	Informa a ocorrência de um evento de navegação, <i>Event</i> , e dados associados, <i>EventData</i> .
NLG	Synthesis GTI	synthesis (Sentence)	Frase a ser falada pelo robô.
Synthesis	ASR	asr (Command)	Para e reinicia o processo de reconhecimento de fala.
Synthesis	GTI	asr (Command) ^a	Alterna o aviso exibido ao utilizador entre “You Can Speak” (<i>Pode falar</i>) e “Let me finish...” (<i>Deixe-me terminar...</i>), de acordo com o status do reconhecimento de fala.
Synthesis	GTI	mouth (Value)	Controla a abertura da boca na cara animada.
GTI	Manager	gti_event (Event)	Informa a ocorrência de um evento na interface gráfica, por exemplo o toque de um botão.
GTI	NLU	recognized (Sentence, 100)	Frase escrita através do ecrã tátil (o segundo parâmetro indica uma confiança de 100%).

^a A mesma mensagem enviado pelo agente de síntese para o ASR é capturada também pelo agente GTI para troca do aviso ao utilizador.

ser tolerantes e robustos em relação a erros no reconhecimento das frases, caso contrário o sistema de diálogo ficará sem ação a maior parte do tempo.

É importante extrair significado de cada saída do ASR ao invés de ignorá-la, mesmo se não for gramaticalmente correta. Para ser capaz de obter a parte com significado de tais saídas, o módulo Natural Language Analysis (NLA) dum sistema Spoken Language Understanding (SLU) deve ser robusto a vários tipos de erros.

Foram desenvolvidos dois agentes de reconhecimento automático de fala. O primeiro utiliza o produto Nuance 8.0 como base e oferece as mesmas funcionalidades do módulo em uso até ao momento.

O segundo foi desenvolvido como projeto de final de licenciatura [Abreu, 2006]. Este agente utiliza a biblioteca Java da OAA e disponibiliza alguns recursos adicionais, como a possibilidade de troca dinâmica de gramática ou dicionário e o ajuste de nível no microfone. Este agente baseia-se no Sphinx-4 [Walker *et al.*, 2004], sistema avançado de reconhecimento de fala escrito inteiramente na linguagem de programação Java. O Sphinx foi criado através de uma colaboração entre a Universidade de Carnegie Mellon, *Sun Microsystems Laboratories*, *Mitsubishi Electric Research Labs* (MERL) e *Hewlett Packard* (HP).

Por utilizar Java, o agente baseado em Sphinx consome mais recursos computacionais do que o primeiro, especialmente, a memória. Por isso, foi utilizado em testes, mas o primeiro ainda é o agente padrão na arquitetura.

Agente de Análise Sintática de Linguagem Natural — O agente de análise sintática de linguagem natural (NLA, de *Natural Language Analysis*) recebe frases vindas dos agentes ASR e GTI. As frases podem ser analisadas de duas formas. A primeira – a ideal – utiliza o programa *LCFlex*, que realiza uma análise sintática profunda [Rosé e Lavie, 1998]. O *LCFlex* pode lidar com frases mesmo que possuam alguns erros gramaticais. A segunda forma de análise é utilizada quando a frase possui demasiados erros de gramática. Baseia-se numa análise sintática superficial realizada pelo Tilburg Memory Based Learner (TiMBL) [Daelemans *et al.*, 2002], que é um programa que implementa várias técnicas de aprendizagem baseada em memória (Memory Based Learning (MBL)) e é utilizado para classificar as frases. Mais detalhes sobre esse processo são fornecidos na seção 3.2.3.

Agente de Compreensão de Linguagem Natural — O agente de compreensão de linguagem natural (Natural Language Understanding (NLU)) atua sobre o resultado do processamento do agente NLA. Sua função essencial é gerar a estrutura semântica, ou seja, o significado. Ao final, transmite o resultado para o agente gestor. Para mais detalhes sobre o funcionamento do agente NLU, ver a seção 3.2.4.

Agente de Interface Gráfica e Tátil Agente de interface gráfica e tátil (GTI, de *Graphical and Touch Interface*) lida com as interações através do ecrã tátil disponibilizando seis botões

que podem ser pressionados: ‘You are Ready’, ‘Die’, ‘Yes’, ‘No’ , ‘Set Home’ e ‘Go Home’; gerencia a entrada de linguagem escrita, que pode ser capturada através de um teclado virtual; e por último, apresenta a cara animada do Carl.

Conseqüentemente, as frases faladas pelo robô e aquelas reconhecidas pelo agente ASR podem ser visualizadas, assim como informações do agente de navegação, como velocidade e carga da bateria. Além disso, o agente GTI alterna a mensagem exibida ao utilizador entre “You Can Speak” (*Pode falar*) e “Let me finish...” (*Deixe-me terminar...*), de acordo com o status do reconhecimento de fala.

Agente Gestor — Este é o agente que controla o robô como um todo. Mantém registro da maior parte dos eventos do sistema, como eventos do agente Graphical and Touch Interface (GTI), de tempo (*timeouts*), de percepção e frases analisadas. Seu funcionamento será abordado em detalhes no capítulo 4.

Este agente também inclui o módulo de Aquisição e Gestão do Conhecimento (KAM, de *Knowledge Acquisition and Management*), desenvolvido para dar suporte a diálogos de alto nível. Este módulo permite a integração de informação vinda de diferentes utilizadores e é capaz de lidar com factos contraditórios. A descrição completa do módulo é apresentada na seção 3.3.

Uma curiosidade técnica, embora este agente tenha sido desenvolvido em Prolog, a biblioteca de Prolog do OAA não pôde ser usada por incompatibilidade entres versões. Isto acontece porque utilizamos o SWI-Prolog e a biblioteca fornecida é para o Sicstus e Quintus Prolog. Assim, tivemos que utilizar a biblioteca C do OAA através da interface C do SWI-Prolog.

Agente de Geração de Linguagem Natural — O agente de geração de linguagem natural (NLG, de *Natural Language Generation*) é baseado no sistema Multimodal Functional Unification Grammar (MUG) [Reitter, 2004]. Este agente produz uma frase a partir de um ato de fala e uma lista de relações semânticas. No final do processo, envia uma mensagem para o agente de Síntese de Fala com a frase gerada.

Assim como o agente Gestor, o agente Natural Language Generation (NLG) foi implementado em Prolog e comunica-se com o Facilitador da mesma maneira descrita anteriormente. Para mais informações sobre este agente, ver a seção 3.4

Agente de Síntese de Fala — O agente de síntese de fala utiliza o sistema *Festival Speech Synthesis* [Black *et al.*, 1999]. Sua função é transformar a frase gerada pelo agente NLG em áudio, ondas sonoras. Além disso, também envia valores de abertura da boca para o agente GTI para sincronização dos lábios da cara animada.

Imediatamente antes do início da síntese, envia uma mensagem para o agente ASR interrompendo o processo de reconhecimento de fala. Assim que a síntese termina, envia outra mensagem reiniciando o processo. Esta interrupção é necessária para evitar que o robô tente

reconhecer a própria fala. Dessa forma, é preciso um pouco de atenção ao interagir com o robô, pois ele fica “surdo” enquanto fala. Existe uma indicação gráfica desta situação para facilitar a interação.

Agente de Navegação — Este agente é responsável pela navegação do robô. Recebe comandos como: *stop*, *move forward*, *move backward*, *turn left*, *turn right*, *wander*, *explore*, entre outros. É baseado na arquitetura para controle dos robôs *Pioneer*. Esta é subdividida em Saphira, com funções de mais alto nível, e ARIA, de elementos básicos de ação e interpretação de sensores [Konolige *et al.*, 1997].

Este agente lida com a percepção geral do robô, incluindo sonares, e também pode enviar informação de navegação para ser mostrada pelo agente GTI. É o único agente que não é executado no computador portátil incorporado no robô, mas, sim, no computador da base do robô.

3.1.2.4 Aspectos da Implementação dos Agentes

A figura 3.5 mostra como as mensagens são trocadas pelos agentes. Como se pode ver, todos os agentes comunicam entre si através do facilitador, este é o comportamento normal numa comunidade de agentes OAA. Além dessa comunicação padrão, os agentes de síntese de fala e GTI possuem uma ligação direta.

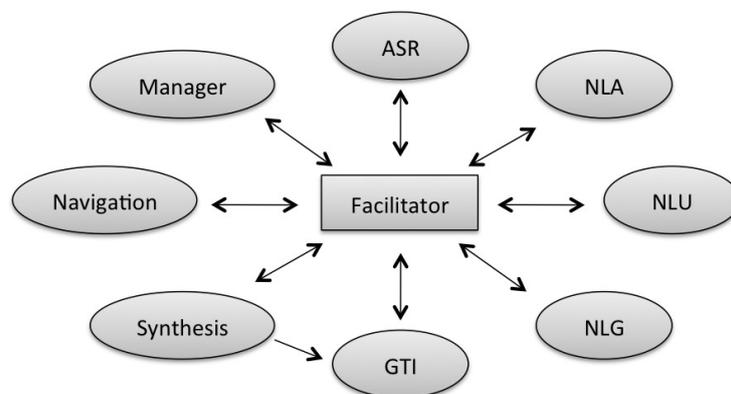


Figura 3.5: Agentes OAA no robô Carl – ligações de comunicação direta entre os agentes de síntese e GTI.

A comunicação direta entre os agentes de síntese e GTI deve-se a questões de desempenho. As mensagens que passam pelo facilitador não são lentas, no entanto, para a tarefa de sincronia de lábios, a comunicação mais rápida teve que ser utilizada.

Alguns detalhes mais técnicos. Como dito anteriormente nesta seção, um agente precisa da biblioteca OAA para conectar-se e registrar-se com o agente facilitador. Após a conexão e registro, um agente comum irá simplesmente chamar a função *oaa_MainLoop*, que o colocará

em espera de eventos do facilitador. Estes eventos vão usualmente ativar as funções registadas pelo agente.

Entretanto, alguns agentes possuem o seu próprio ciclo de controlo, como os agentes GTI e Gestor. Estes não podem simplesmente chamar *oaa_MainLoop*. Eles precisam de um controle maior através da função *oaa_ProcessAllEvents*. Esta não bloqueia o agente, apenas verifica e processa os eventos que o agente possa vir a ter.

3.2 Compreensão de Linguagem Falada

Para que um robô aceite instruções ou adquira conhecimento de utilizadores humanos através de interação em linguagem falada, é preciso uma capacidade robusta de compreensão de linguagem falada (SLU, de *Spoken Language Understanding*).

No robô Carl, a compreensão de fala é realizada pelos agentes ASR, NLA e NLU, como já foi explicado. Nesta seção, são descritos em maior detalhe os agentes NLA e NLU, desenvolvidos inicialmente no âmbito de um trabalho de mestrado [Rodrigues, 2004]. O agente NLA foi adaptado para se comunicar com a arquitetura OAA no escopo deste doutoramento, enquanto o agente NLU teve de ser reconstruído, por motivos expostos a seguir.

Como o módulo ASR do Carl utilizava uma ferramenta comercial, o Nuance 8.0, o esforço centrou-se no subsistema de compreensão de linguagem. Este é dividido em duas partes: na primeira, o agente NLA extrai a estrutura sintática da frase reconhecida pelo agente ASR; na segunda parte, o agente NLU obtém a respectiva informação semântica.

3.2.1 Atos de Fala

No Carl, a comunicação humano-robô é realizada através de diálogos em linguagem falada e escrita, assim como interações táteis. A comunicação é modelada como uma troca de mensagens, ou atos de fala. Na fase inicial do projeto CARL, foi definida a linguagem HRCL (de *Human-Robot Communication Language*) [Seabra Lopes e Teixeira, 2000], composta por um conjunto de tipos de mensagens que podem ser trocadas entre um robô e um utilizador humano. A linguagem HRCL foi inspirada na linguagem KQML (de *Knowledge Query and Manipulation Language*), que por sua vez é a linguagem de interface da linguagem ACL (de *Agent Communication Language* [Labrou e Finin, 1997]). No âmbito deste trabalho, são suportados os tipos de mensagens ou atos de fala mais básicos da HRCL, tendo sido acrescentados outros. A tabela 3.2 mostra os tipos de atos de fala que o Carl é capaz de reconhecer e/ou gerar⁴. Como assinalado na tabela, alguns atos de fala apenas são suportados pelo reconhecimento (falas do utilizador humano), como *ready(S,R)*; e outros apenas pela geração (falas do robô), como, *offer_help(S,R)*.

⁴S=sender, R=receiver.

Tabela 3.2: Atos de fala reconhecidos e/ou gerados pelo Carl

Tipo	Descrição
tell(S,R,C)	<i>S</i> informa a <i>R</i> que a frase <i>C</i> é verdadeira.
reply(S,R,C)	<i>S</i> responde a <i>R</i> a frase <i>C</i> .
ask(S,R,C)	<i>S</i> pede a <i>R</i> para fornecer uma instanciação da frase <i>C</i> .
ask_if(S,R,C)	<i>S</i> quer saber se <i>R</i> considera a frase <i>C</i> verdadeira.
achieve(S,R,C)	<i>S</i> pede a <i>R</i> para executar a ação <i>C</i> no seu ambiente físico.
ready(S,R)	<i>S</i> (humano) comanda <i>R</i> (robô) a iniciar a interação.
sorry(S,R)	<i>S</i> pede desculpa a <i>R</i> .
thanks(S,R)	<i>S</i> agradece a <i>R</i> .
bye(S,R)	<i>S</i> despede-se de <i>R</i> .
confirm(S,R)	<i>S</i> confirma a <i>R</i> uma informação anterior.
deny(S,R)	<i>S</i> nega a <i>R</i> uma informação anterior.
greet(S,R)	<i>S</i> faz saudação a <i>R</i> .
rejection(S,R)	<i>S</i> (robô) rejeita a informação anterior de <i>R</i> (humano) por motivo de baixa confiança do ASR.
observation(S,R)	<i>S</i> (robô) faz um comentário a <i>R</i> (humano) relacionado a uma informação dada anteriormente.
feedback(S,R)	<i>S</i> (robô) dá um retorno a <i>R</i> (humano) indicando a recepção de uma informação anterior.
introduce(S,R)	<i>S</i> (robô) se apresenta a <i>R</i> (humano).
offer_help(S,R)	<i>S</i> (robô) oferece ajuda a <i>R</i> (humano).
confirmation(S,R)	<i>S</i> (robô) pede confirmação a <i>R</i> (humano) sobre informação anterior.
clarify(S,R)	<i>S</i> (robô) pede clarificação a <i>R</i> (humano) acerca de uma frase apenas parcialmente compreendida.
inform(S,R,C)	<i>S</i> (robô) informa <i>R</i> (humano) um estado interno <i>C</i> , por exemplo, baixo nível de bateria.
suggest(S,R,C)	<i>S</i> (robô) sugere a <i>R</i> (humano) que considere fazer <i>C</i> .

3.2.2 Linguagem de Representação

Domínios complexos necessitam de uma representação de conhecimento (Knowledge Representation (KR)) genérica e flexível [Russell e Norvig, 2010]. Esse é o caso de sistemas de diálogo em robôs móveis inteligentes. A definição da linguagem de representação da semântica das frases baseia-se em definições típicas de redes semânticas e nos diagramas de classes e objetos da Unified Modeling Language (UML) [Rumbaugh *et al.*, 2004]. Naturalmente, esta linguagem é também usada para representar o conhecimento acumulado pelo robô, como será descrito na próxima seção.

As redes semânticas [Sowa, 2005] abrangem os principais requisitos de representação para dar suporte a um diálogo de alto nível. As entidades são facilmente representadas e a inferência

é bastante simples, bastando seguir as relações entre entidades. As redes semânticas permitem uma computação eficiente e também uma mais fácil compreensão através da apresentação gráfica. Com as redes semânticas, a regra de herança pode ser facilmente aplicada. Nesta regra, as propriedades do supertipo são copiadas para o subtipo, exceto se houver uma redefinição da propriedade no subtipo.

A linguagem de representação proposta [Seabra Lopes *et al.* , 2005a] assume que serão atribuídos identificadores únicos, gerados pelo sistema, aos objetos. Na implementação, são usados identificadores inteiros gerados sequencialmente à medida que novos objetos são introduzidos na base de conhecimento. Os predicados que fornecem informação acerca de objetos específicos são *instance(ObjID, Type)* e *name(ObjID, Name)*.

A relação de generalização entre dois tipos é declarada com *subtype(SubType, Type)*. Outras relações (*composition, association, attribute, function*) aplicam-se tanto a tipos como a objetos. A tabela 3.3 apresenta os predicados utilizados e, a título de exemplo, a tabela 3.4 mostra representações semânticas de algumas frases.

Tabela 3.3: Predicados usados para representação de conhecimento

Predicado	Descrição
<i>type(Type)</i>	Declara um tipo.
<i>subtype(SubType, Type)</i>	Relação de generalização entre <i>SubType</i> e <i>Type</i> .
<i>instance(Obj, Type)</i>	Declara que <i>Obj</i> é uma instância de <i>Type</i> .
<i>name(Obj, Name)</i>	Declara o nome de um dado objeto.
<i>function(Obj, Function)</i>	Declara a função de um dado objeto.
<i>attribute(Type, Attribute)</i>	Declara um atributo de um dado tipo.
<i>attribute(Type, Attribute, Value)</i>	Declara um atributo de um dado tipo com um valor por omissão (<i>default</i>).
<i>attribute(Obj, Attribute, Value)</i>	Declara o valor de um atributo de um objeto dado.
<i>composition(CompType, Type)</i>	Relação de composição entre dois tipos.
<i>composition(CompObj, Obj)</i>	<i>CompObj</i> é componente de <i>Obj</i> .
<i>association(Name, TypeX, TypeY)</i>	Associação genérica ente os tipos <i>X</i> e <i>Y</i> .
<i>association(Name, Obj1, Obj2)</i>	Associação entre dois objetos.
<i>association(Name, Obj, Type)</i>	Associação entre um objeto e um tipo.

Como se vê na tabela 3.3, as relações tradicionais dos diagramas de classes UML, como generalização, associação e composição, são suportadas. Dessa forma, as características principais dos formalismos de representação de conhecimento em inteligência artificial e de modelagem de informação em engenharia de software são garantidas.

3.2.3 Análise Sintática

O agente NLA do Carl baseia-se no trabalho de [Rodrigues *et al.* , 2004]. O agente utiliza o LCFlex [Rosé e Lavie, 1998], um analisador flexível projetado para interpretação

Tabela 3.4: Alguns exemplos de representação semântica de frases

Frase	Semântica
Bob is a cat	name(1, bob), instance(1, cat)
The color of Bob is black	name(1, bob), attribute(1, color, black)
Birds fly	attribute(bird, fly, yes)
Cats are felines	subtype(cat, feline)
Cats have legs	composition(cat, leg)
Bob has a leg	name(1, bob), composition(2,1), instance(2, leg)
Bob likes milk	name(1, bob), association(like, 1, milk)
Cats like fish	association(like, cat, fish)
Bob likes Minie	name(1, bob), name(3, minie), association(like, 1, 3)

robusta eficiente, baseado no formalismo Lexical Functional Grammar (LFG) para definição de gramática. Dada a sequência de palavras extraída pelo reconhecedor de fala, o LCFlex produz uma sequência de palavras o mais próxima possível da entrada. Para isso, o LCFlex recorre à supressão de palavras e à unificação de características flexível e seletiva.

Embora o LCFlex possa processar sequências de palavras que não são descritas, completamente, pela gramática do sistema, se esta sequência tiver um grande desvio da gramática, os resultados podem perder a parte significativa da mesma. Para estes casos, uma abordagem complementar usando a aprendizagem baseada em memória (MBL, de *Memory-Based Learning*) foi desenvolvida.

MBL é baseado na ideia de que o comportamento inteligente pode ser obtido através de raciocínio por analogia. Os algoritmos MBL pegam um conjunto de exemplos como entrada e produzem um classificador capaz de classificar novos padrões de entrada. Se o sistema aprende a detectar informação importante a partir dos exemplos, ele deve ser capaz de identificar informação significativa em qualquer sequência de palavras. A ferramenta TiMBL [Daelemans *et al.*, 2002] é usada para este fim.

A arquitetura do sistema SLU híbrido é representada na figura 3.6. O sistema possui dois caminhos complementares: o analisador baseado em conhecimento e a abordagem MBL guiada por dados, onde a última é utilizada quando a primeira falhar. Portanto, a prioridade do sistema é utilizar para a análise sintática o LCFlex, sistema capaz de realizar uma análise mais profunda que o TiMBL, extraíndo mais informação das sequências de palavras. Se o analisador falhar ou se a frase for considerada como tendo um grande desvio da gramática, TiMBL é utilizado para extrair informação significativa.

Para construir a estrutura sintática de uma frase, a informação usada é o papel morfossintático de cada palavra ao invés da própria palavra. Assim, as entradas lexicais da gramática utilizada pelo LCFlex são etiquetas Part-Of-Speech (POS) ao invés das palavras.

As etiquetas POS são atribuídas em tempo de execução por dois etiquetadores, o Language

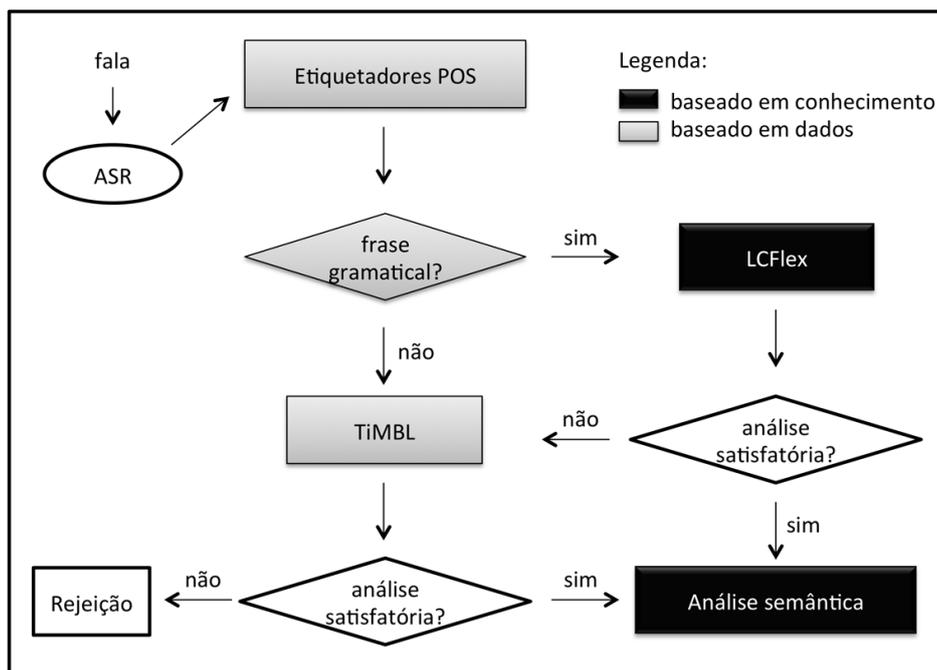


Figura 3.6: A arquitetura do sistema SLU [Rodrigues *et al.*, 2004]

Technology Part-Of-Speech (LTPOS) [Mikheev, 1997] e o BrillTagger [Brill, 1995]. Os etiquetadores usam o conjunto de etiquetas *Penn Treebank* [Marcus *et al.*, 1993] que possui 36 etiquetas, portanto o léxico da gramática utilizada possui apenas 36 entradas, o que a torna pequena e fácil de manter.

Se ambos os etiquetadores concordarem em mais de 75% das etiquetas e a frase for classificada como gramaticalmente correta pelo TiMBL, a frase é encaminhada para o LCFlex para uma análise sintática profunda. Caso contrário, ela é encaminhada para outra instância do TiMBL para uma análise superficial.

Se a estrutura sintática da frase retornada pelo LCFlex incluir mais de 75% do total de palavras da frase, o resultado da análise é encaminhado para o módulo de análise semântica. Caso contrário, o sistema considera que a frase tem um grande desvio da gramática e a análise final é superficial, feita por uma segunda instância do TiMBL. Esta também pode rejeitar a frase, caso não consiga uma análise válida.

Com esta arquitetura híbrida, um sistema foi desenvolvido capaz de executar uma análise profunda se a frase é em grande parte aceita pela gramática, mas também capaz de realizar uma análise superficial caso a frase possua erros severos. Uma explicação mais detalhada sobre o sistema SLU é dada em [Rodrigues *et al.*, 2004; Rodrigues, 2004; Seabra Lopes *et al.*, 2003a].

A tabela 3.5 mostra exemplos do resultado da análise sintática realizada pelo LCFlex e pelo TiMBL. Observe que o LCFlex produziu uma árvore sintática completa, enquanto o

TiMBL identificou apenas um nome. O agente NLA envia o resultado da análise sintática para o agente NLU usando a mensagem *parsed_sentence* (ver tabela 3.1).

Tabela 3.5: Alguns exemplos de análise sintática

Frase	Tipo de Análise	Resultado
“mario is a professor”	profunda (LC-Flex)	analysis('((STYPE TELL) (NP ((PNOUN Mario))) (VP ((VERB is) (NP ((DET a) (CNOUN professor))))))')
“paul perry”	superficial (TiMBL)	analysis(tell_,none,name_(_6337,paul))

3.2.4 Análise Semântica

O agente NLU recebe como entrada a estrutura sintática produzida pelo agente NLA e produz como resultado a semântica da frase utilizando a linguagem de representação do conhecimento apresentada na seção 3.2.2.

Este agente foi criado para a nova arquitetura desenvolvida no âmbito do presente trabalho, não existindo como processo independente na arquitetura anterior. Seu desenvolvimento foi necessário por dois motivos [Seabra Lopes *et al.*, 2005a]. O primeiro, de ordem estrutural. Por razões históricas, o processo de extração de semântica estava integrado noutro agente, o gestor. Na nova arquitetura computacional do Carl, e por razões de modularidade, entendeu-se que a análise semântica deveria ser realizada por um agente independente, criando-se assim o agente NLU.

O outro motivo para a criação de um novo agente NLU foi o desenho e adoção de uma nova linguagem de representação do conhecimento (ver seção 3.2.2). Era necessário que o agente NLU produzisse uma semântica usando a nova linguagem de representação de conhecimento, aliás também usada no módulo Knowledge Acquisition and Management (KAM).

Quando a análise sintática é realizada pelo LCFlex, um predicado recursivo faz a travessia da árvore sintática para extrair as relações semânticas e o ato de fala, como *tell* ou *ask*. Por exemplo, se o utilizador informar ao Carl que “Professor James is in France”. O sistema extrai a seguinte semântica: [name(*X*, *james*), function(*X*, *professor*), association(*be_in*, *X*, *Y*), name(*Y*, *france*)].

Quando o TiMBL realiza a análise sintática, o trabalho do agente NLU é de apenas repassar as entidades identificadas, por exemplo, um nome ou um tipo.

A informação a ser transmitida do agente NLU para o gestor de diálogo é colocada na seguinte estrutura semântica: sem(*SA*, *SA_data*, *Main*, *Rels*). Nela, o primeiro argumento *SA*, de *Speech Act*, é o ato de fala da frase reconhecida. *SA_data* contém dados adicionais para alguns atos de fala. Por exemplo, no caso do ato de fala *ask*, podemos ter como *SA_data*

where, what ou *who*. *Main* identifica a principal entidade numa pergunta. Por último, *Rel*s contém a lista com as relações que representam a semântica da frase.

3.3 Aquisição e Gestão de Conhecimento

O cenário no qual o robô Carl deve atuar envolve diálogo com vários utilizadores e troca de informação com eles. O robô deve possuir capacidades de aquisição de conhecimento, que acumulará numa base de conhecimento (KB, de *Knowledge Base*), e de resposta a perguntas. Para esse efeito, foi desenvolvido o módulo de Aquisição e Gestão de Conhecimento (KAM, de *Knowledge Acquisition and Management*). É importante notar que se pretende que o robô seja capaz de iniciar uma interação com um utilizador humano sem conhecimento prévio. O módulo KAM está integrado no agente gestor.

Como o robô pode adquirir conhecimento a partir de diferentes origens (utilizadores), o módulo KAM deve lidar com eventuais informações contraditórias. Além disso, como a aquisição de informação é realizada, principalmente, através de linguagem natural falada, o grau de confiança do reconhecedor deve ser levado em consideração.

As funcionalidades do módulo KAM são fornecidas em grande parte pelos seguintes procedimentos⁵:

- *kb_tell(+UserName, +RecConf, +Semantics)* – O interlocutor *UserName* informa *Semantics* ao sistema, sendo *RecConf* a confiança no reconhecimento fornecida pelo agente ASR.
- *kb_ask(+Semantics, -Reply, -Confidence)* – O sistema é perguntado acerca de *Semantics*. A resposta (*Reply*) e respectiva confiança (*Confidence*) são retornadas.

O procedimento *kb_tell* simplesmente armazena a informação dada pelos interlocutores na base de conhecimento do módulo KAM. Por exemplo, a chamada *kb_tell(user1, 100, [subtype(mouse, animal)])* resulta no armazenamento do fato *int_(user1, 100, subtype(mouse, animal))*.

O procedimento *kb_ask* utiliza mecanismos de inferência para fornecer a melhor resposta possível. Mais detalhes sobre o seu funcionamento encontram-se na seção 3.3.3.

3.3.1 Trabalho Relacionado

Estas questões foram, em parte, abordadas em outros contextos. Benferhat et al apresentam estratégias para resolução de conflitos desenvolvidas para o tratamento de exceções e

⁵Aqui, e no resto da tese, adotamos a convenção do Prolog na qual o símbolo ‘+’ antes do parâmetro indica que o valor do mesmo é passado para o procedimento (parâmetro de entrada) e o símbolo ‘-’ indica que o valor é retornado (parâmetro de saída). Outra convenção do Prolog adotada é a de que os nomes das variáveis começam com letra maiúscula, enquanto os nomes de constantes e predicados/procedimentos começam com letra minúscula.

revisão iterativa de crenças (*iterated belief revision*), mas que também podem ser aplicadas na combinação de informação de diferentes origens [Benferhat *et al.*, 2004]. Nestas estratégias, as informações conflitantes são enfraquecidas, ao invés de completamente eliminadas. Os autores alegam que, desta forma, conseguem pelo menos manter, e na maioria dos casos, aumentar a quantidade de informação resultante, quando comparado ao uso de outras estratégias.

Brazdil e Torgo desenvolveram um método para construir uma base de conhecimento integrada a partir de várias bases separadas [Brazdil e Torgo, 1990]. Neste método, as bases de conhecimentos são caracterizadas através de testes qualitativos e quantitativos, que indicam os benefícios e a precisão de cada base individualmente. A seguir, os valores destes testes são utilizados para construir a base integrada.

Qi *et al* propõem um modelo de negociação para o problema de consolidação de crença (*belief merging*) baseado em lógica de possibilidade [Qi *et al.*, 2007]. Neste modelo, cada fonte de crença é tratada como uma base de crenças possibilística (*possibilistic belief base* – PBB), e a consolidação é realizada em duas etapas. Na primeira, crenças de algumas bases são enfraquecidas para diminuir as inconsistências. Isto é feito com um modelo de negociação de crenças com prioridades. Na segunda, as bases de conhecimento resultantes são combinadas através de um operador conjuntivo, o qual pode ter um efeito de reforço na lógica possibilística.

3.3.2 Mecanismos de Inferência Básicos

A inferência lógica é um processo pelo qual novas informações são derivadas a partir de outras já conhecidas. Existem três tipos básicos de inferência: a dedução, a indução e a analogia [Michalski, 1994]. Considere a equação 3.1, na qual BC (*base de conhecimento*) identifica o conjunto de informações que já estão representadas em um sistema, ou que um indivíduo possui, P identifica uma premissa e C identifica uma consequência lógica de BC e P .

$$P \cup BC \models C. \quad (3.1)$$

A dedução é usada para derivar a consequência C , dados BC e P . Na indução, a premissa P é hipotetizada, dados C e BC . Finalmente, a analogia resulta da combinação entre dedução e indução.

3.3.2.1 Dedução

A dedução é conservadora da verdade. Michalski, no âmbito da sua “Inferential Theory of Learning” [Michalski, 1994], identifica diversos tipos de inferência dedutiva. Estes mecanismos são explicados e diferenciados em termos do nível de detalhe das declarações e da extensão dos respectivos conjuntos de referência (conjuntos de objetos a que essas declarações se referem). Assim, por exemplo, a abstração é um mecanismo que reduz o nível de detalhe

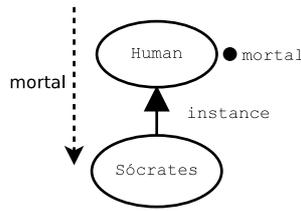


Figura 3.7: Exemplo de inferência dedutiva: ‘Sócrates é mortal’

de uma declaração, mantendo o conjunto de referência. No módulo de aquisição e gestão do conhecimento do Carl, utilizam-se essencialmente mecanismos de especialização dedutiva, ou seja, mecanismos que reduzem o conjunto de referência mantendo o conteúdo semântico da declaração. Concretamente, este tipo de inferência é utilizado quando um tipo possui um atributo ou relação com valor atribuído e uma questão é colocada sobre esse atributo numa instância ou subtipo desse tipo.

No exemplo abaixo, temos como *BC* a regra de todos os homens serem mortais e como *P* o fato de Sócrates ser um homem. Tomando essas duas informações como verdadeiras, a dedução permite inferir a consequência lógica *C* de que Sócrates é mortal. A figura 3.7 mostra isso graficamente.

Todos os homens são mortais
Sócrates é um homem
Portanto, Sócrates é mortal.

Isto não é mais do que a aplicação da regra da instanciação universal [Michalski, 1994]:

$$\frac{\begin{array}{ll} e \in H & e \text{ é um elemento de } H \\ \forall x \in H, p(x) & \text{todo elemento de } H \text{ possui a propriedade } p \end{array}}{\begin{array}{ll} p(e) & e \text{ possui a propriedade } p \end{array}}$$

Utilizando a linguagem descrita na seção 3.2.2 para representar o exemplo dado acima, a base de conhecimento deverá indicar que um indivíduo com o nome ‘socrates’ é uma instância do tipo ‘man’:

BC = [name(i37, socrates), instance(i37, man), type(man)]

Aqui, ‘i37’ é o identificador internamente atribuído ao indivíduo em questão (Sócrates). Por sua vez, a premissa indicará que o tipo ‘man’ tem o atributo ‘mortal’ com o valor ‘yes’:

P = [attribute(man, mortal, yes)]

Com esta premissa, e dado o conhecimento prévio, o sistema será capaz de inferir a seguinte consequência:

$C = [\text{attribute}(\text{i37}, \text{mortal}, \text{yes})]$

Assim, colocada a questão sobre a mortalidade de ‘socrates’ com a pergunta $[\text{name}(X, \text{socrates}), \text{attribute}(X, \text{mortal}, \text{yes})]$, o módulo de gestão do conhecimento unificará a variável X com o identificador ‘i37’ e responderá que sim, ou seja, ‘socrates’ é mortal.

Além da instanciação universal, o módulo de gestão do conhecimento suporta também a regra conhecida como *dictum de omni et nullo* (máxima de todos e nenhum). Trata-se também de uma regra de especialização dedutiva. Neste caso, o conjunto de referência da premissa é reduzido a um qualquer subconjunto, e não a uma instância específica, como acontece na instanciação universal. Por exemplo, sabendo que os homens são mamíferos e tendo como premissa que os mamíferos são mortais, pode-se inferir que os homens são mortais.

3.3.2.2 Indução

Quando temos a base de conhecimento prévio, BC , e a consequência, C , a indução pode ser utilizada para supor a premissa P . Por exemplo, dado o fato de Tom ser um gato (BC) e a consequência de que Tom gosta de leite (C), podemos hipotetizar a premissa de que todos os gatos gostam de leite (P).

Tom é um gato

Tom gosta de leite

Portanto, talvez todos os gatos gostem de leite.

Esta inferência é um exemplo de generalização indutiva [Michalski, 1994]:

$e \in C$	e é um elemento de C
$p(e)$	e possui a propriedade p
$\forall x \in C, p(x)$	<i>talvez, todo elemento de C possua a propriedade p</i>

A generalização indutiva é o inverso da especialização dedutiva. Neste trabalho, a inferência por generalização indutiva é usada quando objetos de um mesmo tipo possuem atributos ou associações não conhecidos para o tipo, ou quando subtipos possuem informações que o supertipo não possui.

No módulo de gestão do conhecimento do Carl, o exemplo da figura 3.8 poderia ser representado através das seguintes afirmações:

$BC = [\text{name}(\text{i80}, \text{tom}), \text{instance}(\text{i80}, \text{cat}), \text{name}(\text{i83}, \text{jim}), \text{instance}(\text{i83}, \text{cat}), \text{type}(\text{cat})]$

$C = [\text{association}(\text{like}, \text{i80}, \text{milk}), \text{association}(\text{like}, \text{i83}, \text{milk})]$

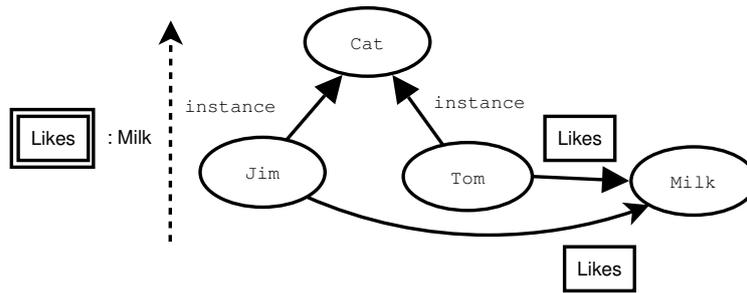


Figura 3.8: Exemplo de inferência indutiva

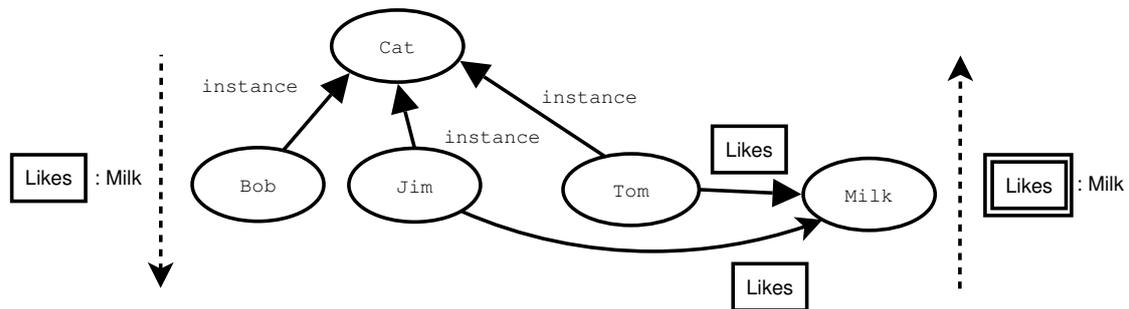


Figura 3.9: Exemplo de inferência por analogia

Neste exemplo, os átomos ‘i80’ e ‘i83’ são os identificadores internamente atribuídos pelo módulo KAM aos indivíduos em questão (os gatos Tom e Jim).

Se a questão “*what does a cat like?*” for colocada, a indução será utilizada para responder, generalizando a partir das duas instâncias conhecidas do tipo *cat*. Partindo da pergunta já representada internamente como [*association(like, cat, X)*], procura-se unificar a variável *X* nas instâncias de *cat*, obtendo a resposta $X=milk$, que é então transmitida ao interlocutor do Carl.

3.3.2.3 Analogia

Finalmente, a analogia é a combinação da dedução e indução. O conhecimento ilustrado na figura 3.9 pode ser representado com as mesmas afirmações usadas no exemplo anterior, relativo à indução (figura 3.8), acrescidas de [*?name(X, bob), instance(X, cat)*]. Agora suponha que alguém pergunta “*what does Bob like?*” (“de que gosta o Bob?”)⁶. Como esta informação não está nem no objeto *Bob* nem no tipo *cat*, o módulo tem que: primeiro utilizar indução a partir dos objetos *Tom* e *Jim* até o tipo *cat*; e então utilizar dedução a partir do tipo *cat* até o objeto *Bob*.

⁶[*name(X, bob), association(like, X, Y)*].

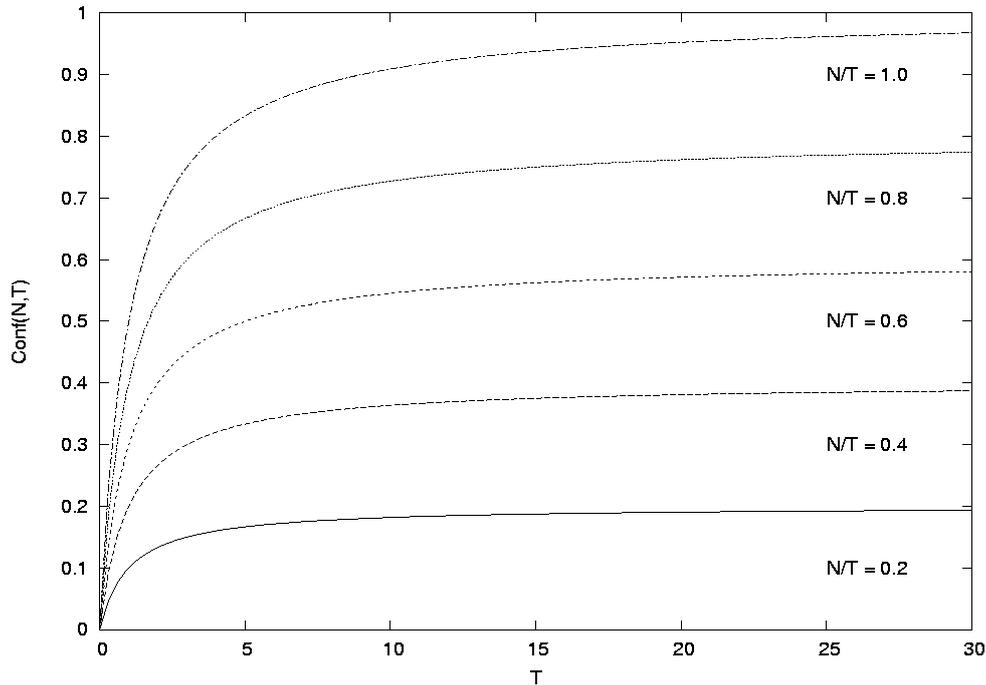


Figura 3.10: Comportamento da fórmula 3.2: os valores de $conf(N, T)$ tendem para $\frac{N}{T}$ à medida que T aumenta

3.3.3 Resposta a Perguntas

Para se obter resposta a uma dada pergunta, utiliza-se o procedimento *kb_ask*. Nele, o grau de confiança retornado é calculado como se segue. Considere um atributo de um objeto ou tipo para o qual um certo valor é suportado por N interlocutores e que um total de T interlocutores forneceram valores para este atributo. Neste caso, a confiança que o valor mencionado é o correto é dada por:

$$Conf(N, T) = \frac{N}{T + 1} \quad (3.2)$$

Observe que a confiança em respostas que se baseiam em poucas afirmações é reduzida, como ilustrado na figura 3.10.

Se for colocada uma questão acerca do valor de um atributo de um tipo, o Algoritmo 1 é utilizado para determinar esse valor. O passo do algoritmo que faz a travessia da árvore de descendentes de um tipo calcula o número de ocorrências dos valores possíveis do atributo no tipo e em todos os subtipos (e sub-subtipos, etc.) e respectivos objetos. Se houver um supertipo, o resultado da travessia da árvore é combinado com um resultado similar herdado do supertipo. Nesta combinação, é formada uma lista dos pares (valor, confiança) que foram encontrados no tipo e no supertipo. Quando o mesmo valor é encontrado em ambos, a média das confianças é utilizada. Após isso tudo, o resultado final é uma lista ordenada pela confiança

associada aos pares.

Algoritmo 1: GET_VALUE

Entrada: Tipo, Atributo

Saída: $((V_1, C_1), \dots, (V_k, C_k))$ onde V significa Valor, C confiança e k o número de diferentes valores do Atributo

início

$V_1, \dots, V_k \leftarrow$ valores possíveis do Atributo

Numa travessia da árvore, calcular o número de ocorrências, N_1, \dots, N_k , de cada valor possível do Atributo no Tipo e nos seus subtipos (e sub-subtipos, etc) e respectivos objetos

$T \leftarrow \sum_{i=1}^k N_i$ (número total de afirmações sobre valores do Atributo)

para $i \leftarrow 1$ **até** k **faça**

$C_i \leftarrow \text{conf}(N_i, T)$ (De acordo com a Equação 3.2)

se Tipo não possui supertipo **então**

retorna $((V_1, C_1), \dots, (V_k, C_k))$

$ST \leftarrow$ supertipo de Tipo

$((V_1, C'_1), \dots, (V_k, C'_k)) \leftarrow \text{GET_VALUE}(ST, \text{Atributo})$

para $i \leftarrow 1$ **até** k **faça**

$C''_i \leftarrow (C_i + C'_i)/2$

retorna $((V_1, C''_1), \dots, (V_k, C''_k))$

fim

Se a questão for acerca de um valor de um atributo num objeto, a confiança para todos os valores possíveis é calculada no objeto (através da Equação 3.2) e no respectivo tipo (Algoritmo 1). O valor com a confiança mais elevada é retornado.

Uma avaliação do módulo de aquisição e gestão de conhecimento é apresentada no capítulo 5, seção 5.1.2.

3.3.4 Procedimentos de Acesso ao Módulo KAM

Os procedimentos de acesso ao módulo KAM estão listados na tabela 3.6. Na nomenclatura, adotaremos o prefixo kb (*knowledge base*) para identificar que estes procedimentos fazem uso da base de conhecimento.

Os dois procedimentos básicos, *kb_tell* e *kb_ask*, suportam a introdução de novo conhecimento e a obtenção de respostas para perguntas do utilizador, respectivamente. Na introdução de novo conhecimento, são gerados identificadores internos para os objetos referidos pela primeira vez. Por exemplo, dada a frase, “Bob is in the lab”, a chamada *kb_tell(user1, 72, [name(X, bob), association(be_in, X, Y), instance(Y, lab)])* armazena as seguintes cláusulas na base de conhecimento:

```
int_(user1, 72, name(i23, bob)).
```

```
int_(user1, 72, instance(i24, lab)).
int_(user1, 72, type(lab)).
int_(user1, 72, association(be_in, i23, i24)).
```

Neste exemplo, os átomos ‘i23’ e ‘i24’ foram os identificadores gerados para o Bob e o laboratório onde ele estava. A relação unária *type* é acrescentada à base de conhecimento para todos os tipos referenciados nas relações *instance* e *subtype* que ocorrem na semântica de uma frase.

Posteriormente, dada a pergunta “Where is Bob?”, a chamada *kb_ask*([*name*(*X*, *bob*), *association*(*be_in*, *X*, *Y*)], *Reply*, _) irá retornar através do parâmetro de saída *Reply* a semântica [*name*(*i23*, *bob*), *association*(*be_in*, *i23*, *i24*), *instance*(*i24*, *lab*)].

Considere agora a introdução das frases “Cats like milk” e “Bob is a cat” através das chamadas *kb_tell*(*user1*, 77, [*association*(*like*, *cat*, *milk*)]) e *kb_tell*(*user1*, 82, [*name*(*X*, *bob*), *instance*(*X*, *cat*)]). Teríamos o seguinte acréscimo à base de conhecimento:

```
int_(user1, 77, association(like, cat, milk)).
int_(user1, 77, type(cat)).
int_(user1, 77, type(milk)).
int_(user1, 82, name(i23, bob)).
int_(user1, 82, instance(i23, cat)).
int_(user1, 82, type(cat)).
```

Nesse cenário, a chamada *kb_ask*([*name*(*X*, *bob*), *association*(*like*, *X*, *Y*)], *Reply*, _), correspondente à pergunta “What does Bob like?”, irá retornar a semântica [*name*(*i23*, *bob*), *association*(*like*, *i23*, *milk*)] no parâmetro *Reply*.

O procedimento *kb_generate_ask* pode ser usado para gerar questões a colocar a um interlocutor do robô com o objetivo de suprir lacunas da sua base de conhecimento. Esse procedimento é capaz de elaborar questões quando se verificar na base de conhecimento alguma das seguintes situações: entidade sem um tipo definido; entidade do tipo *human* sem uma função associada; e tipo sem um supertipo definido.

A geração de comentários sobre a declaração “Peter is in France”, feita pelo procedimento *kb_generate_comment*, poderá ser realizada quando houver na base de conhecimento uma das situações abaixo: informação sobre um outro sujeito para a mesma associação e objeto – exemplo: “Jim is in France”; informação sobre uma associação diferente para o mesmo sujeito – exemplo: “Peter likes Portugal”.

Como exemplo do uso do procedimento *kb_alternate_reply*, suponha que a questão de confirmação “Is Mario a doctor?”, tratada através de *kb_ask_if*, , tenha recebido resposta negativa. Uma resposta complementar pode ser encontrada através da chamada *kb_alternate_reply*([*name*(*X*, *mario*), *function*(*X*, *doctor*)], *NewReply*, *NewConfidence*, *Sta-*

Tabela 3.6: Procedimentos de acesso ao módulo KAM

Procedimento	Descrição
kb_tell(+UserName, +RecConf, +Semantics)	Percorre todas as relações existentes na lista <i>Semantics</i> , gera identificadores para objetos até então desconhecidos, e adiciona as relações à base de conhecimento, associando-lhes <i>UserName</i> e <i>RecConf</i> .
kb_ask(+Semantics, -Reply, -Confidence)	Dada a semântica de uma pergunta aberta no primeiro parâmetro, produz a semântica da resposta e a respectiva confiança. Caso não seja possível obter resposta, retorna “failed” no segundo parâmetro.
kb_ask_if(+Semantics, -Reply, -Confidence)	Dada uma pergunta que pede resposta sim/não com semântica passada no primeiro parâmetro, faz uso do procedimento <i>kb_ask</i> para determinar a resposta.
kb_generate_ask(+UserName, +AskHistory, -ReplyVar, -AskType, -Ask, -Status)	Tenta gerar uma pergunta de acordo com a base de conhecimento atual e levando em consideração o histórico de perguntas já realizadas. <i>Status</i> informa se foi possível gerar uma pergunta. <i>AskHistory</i> é uma lista de perguntas que já foram utilizadas e que, portanto, não deverão ser repetidas. <i>UserName</i> informa o nome do utilizador atual. Finalmente, <i>Ask</i> é uma lista de relações que representam a pergunta gerada, <i>AskType</i> informa o tipo de pergunta e <i>ReplyVar</i> , a principal entidade em questão.
kb_generate_comment(+Semantics, -Comment, -Confidence, -Status)	Tenta gerar um comentário, <i>Comment</i> , relacionado à declaração representada por <i>Semantics</i> , de acordo com a base de conhecimento atual. <i>Status</i> indica se isto foi possível e <i>Confidence</i> mostra a confiança associada.
kb_alternate_reply(+Semantics, -NewReply, -NewConfidence, -Status)	- Busca uma resposta alternativa para uma questão de confirmação, de resposta sim/não, transformando-a em uma questão aberta que possa ser respondida com a base de conhecimento atual. Sua utilização típica se dá após um retorno negativo na chamada de <i>kb_ask_if</i> .
kb_describe(+Name, -LDescription, -Status)	- Este procedimento tenta descrever o objeto identificado por <i>Name</i> através da geração de uma lista de relações, <i>LDescription</i> . <i>Status</i> indica se isto foi possível.

tus). Se a base de conhecimento tiver outra informação acerca da profissão de Mario, a resposta da chamada acima poderia ser:

```
NewReply = [name(i77, mario), function(i77, professor)]
NewConfidence = 60
Status = ok
```

Neste caso, *Status* indica sucesso e *NewConfidence*, uma confiança de 60% na informação “Mario is a professor”.

3.4 Geração de Fala

Uma versão inicial, simples, do agente de geração de linguagem natural (NLG) foi desenvolvida com colaboração de Pedro do Mar [do Mar, 2006]. Após essa primeira versão, foram incluídas novas funcionalidades ao agente, como ampliação da semântica e atos de fala aceites.

A técnica de geração de linguagem natural utilizada anteriormente no robô Carl era baseada em padrões (*templates*). Esta técnica é adotada em vários sistemas de diálogo, tem um bom desempenho, mas possui limitações em relação às frases que são produzidas, reduzindo a variabilidade e também limitando a semântica utilizada.

Devido à própria filosofia do projeto CARL, que envolve comunicação, ação, raciocínio e aprendizagem, o robô precisa de ser capaz de elaborar frases de uma maneira flexível. Assim, o agente NLG foi projetado com a intenção de satisfazer as necessidades de geração de linguagem natural do projeto.

A principal mudança é que a geração do texto passa a ser efetuada a partir da semântica que se pretende transmitir e não mais por padrões.

3.4.1 Ferramentas Analisadas

Para a implementação deste agente, várias ferramentas de geração de linguagem natural foram analisadas. Um resumo das principais será mostrado a seguir.

3.4.1.1 Astrogen

Gerador desenvolvido por Hércules Dalianis da Universidade de Estocolmo [Dalianis, 1996] em linguagem Prolog. A figura 3.11 mostra a arquitetura do Astrogen. Ele é composto pelo gerador de superfície (*surface generator*) e pelos módulos de planeamento de texto (*text planner*) e de planeamento de frase (*sentence planner*), que juntos formam o gerador profundo (*deep generator*).

Na fase de geração profunda, a ferramenta realiza a agregação, ou seja, procura eliminar informação redundante, no entanto, essa supressão não deve alterar o sentido do texto origi-

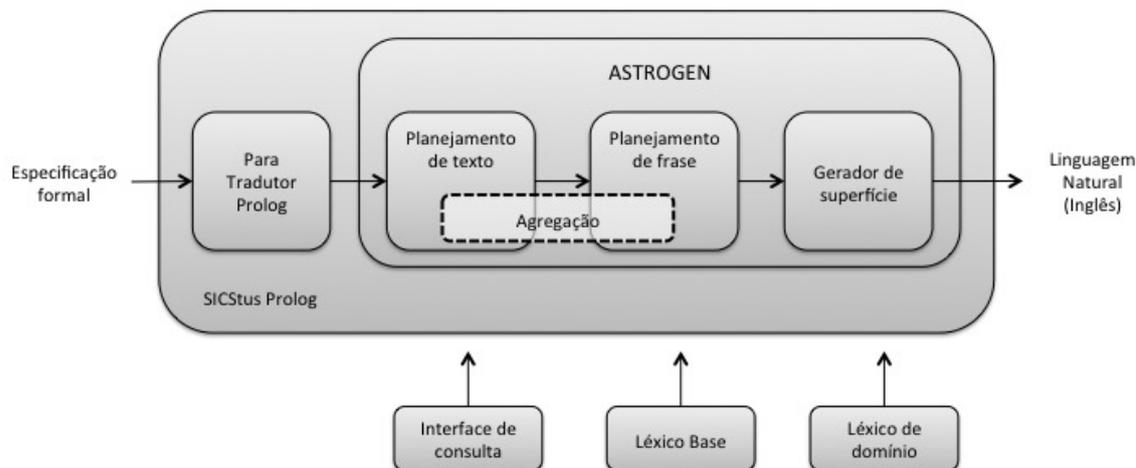


Figura 3.11: Arquitetura do Astrogen

nal. Por exemplo, sem agregação, o pedido de geração $paraphrase(f(pres,isa, john,subscriber), f(pres,isa,mary,subscriber))$, tem como resultado⁷:

*John is a subscriber and
Mary is a subscriber.*

Entretanto, com a agregação ativa, a geração é reduzida à frase:

John and Mary are subscribers.

O controle da agregação é realizado através de parâmetros de configuração do módulo de geração profunda e pode ser ativada em vários níveis.

Antes do gerador de superfície produzir as frases, o módulo de pronomes (*pronominalisation*) pode ser chamado para realizar alterações no texto. Além do presente, o Astrogen possibilita a geração de frases em outros dois tempos verbais: passado e futuro.

O Astrogen foi utilizado anteriormente no robô Carl, tendo sido abandonado por questões de desempenho, de falta de variabilidade nas frases geradas e pela dificuldade em expandir o esquema de frases original.

3.4.1.2 Functional Unification Formalism (FUF)

Em sua tese de doutoramento na Universidade de Columbia, Michael Elhadad apresentou o FUF [Elhadad, 1993], um sistema de geração de linguagem natural que se baseia no formalismo FUG (*Functional Unification Grammars*) e implementado em Common Lisp.

⁷<http://people.dsv.su.se/hercules/ASTROGEN/ASTROGEN.html>.

O FUG utiliza numa estrutura de dados chamada *functional description* (FD), capaz de guardar aspectos funcionais e estruturais de uma representação linguística. Um FD descreve restrições em um objeto e pode ser visto como uma lista de pares atributo-valor, onde um atributo é um rótulo e um valor pode ser tanto um rótulo como um outro FD. Abaixo mostramos um exemplo de FD, o qual produz o texto “The system advises John” quando submetido a um gerador [Elhadad, 1993].

```
I1 = ((cat clause)
      (tense present)
      (action ((lex "advise")))
      (agent ((lex "system") (proper no)))
      (affected ((lex "John"))))
```

Nesse formalismo, existe apenas uma operação para manipular os FDs: a unificação. Ela combina dois FDs de entrada para gerar um FD de saída que agregue as características de ambos. Poderia ser visto como a união de conjuntos, mas a unificação falha quando os FDs possuem atributos com valores incompatíveis. A geração de linguagem é realizada através da unificação de um FD com a representação linguística desejada, como o mostrado acima, e um FD que representa a gramática coberta pelo sistema.

Segundo Elhadad, a criação do FUF como extensão do FUG foi motivada por:

- Eficiência – com o acréscimo da escolha do léxico e de restrições flutuantes, o mecanismo de controle do FUG mostrou-se demasiado lento para as tarefas. Foram desenvolvidos quatro novos controles no FUF.
- Usabilidade – restrições linguísticas comuns eram difíceis ou mesmo impossíveis de serem expressas no formalismo FUG. Isto foi contornado com a criação do mecanismo de tipos no FUF.
- Modularidade – a interação com fontes externas no processo de geração tornou-se necessária para facilitar a declaração de restrições. O FUF traz uma ferramenta que permite tal interação.

Systemic Unification Realization Grammar of English (SURGE) — A SURGE é uma gramática, mas quando combinada com o FUF, torna-se um gerador de superfície reutilizável em sistemas de geração de linguagem natural, com uma boa cobertura da língua inglesa. Ela foi desenvolvida por Michael Elhadad e Jacques Robin [Elhadad e Robin, 1998].

No processo de geração de linguagem, o SURGE encontra-se após o elemento que realiza a escolha léxica (*lexical chooser*) e antes do módulo que traduz a especificação sintática em texto. (*linearizer/morphology*).

Tal como o FUF, o funcionamento do SURGE é centrado na unificação de FDs, também foi desenvolvido em Common Lisp e pode ser caracterizado por:

- definir uma estrutura sintática
- fornecer um ordenamento das restrições entre os elementos sintáticos
- propagar propriedades de relação entre palavras

Multimodal Functional Unification Grammar (MUG) — O sistema MUG foi desenvolvido em Prolog por David Reitter no *MIT Media Lab Europe* [Reitter, 2004]. Ele é formado por uma série de componentes: formalismo MUG (*Multimodal Functional Unification Grammar*) que descreve a sintaxe para desenvolvimento de gramáticas; o motor MUG (*MUG Engine*) controla a geração de linguagem; por último, o ambiente gráfico é chamado de *MUG Workbench*.

O formalismo MUG é baseado no formalismo FUG, descrito acima. Assim como o FUG, o MUG utiliza a estrutura de dados FD e sua operação de unificação. O motor MUG permite a ligação de fontes externas, como bases de conhecimento, através da disponibilização de interfaces apropriadas.

O ambiente gráfico possui um modo de inspeção da geração realizada que permite uma boa depuração do código. Além disso, o MUG também oferece uma visão procedimental do registo de execução (log), mostrando os passos tomados pelo interpretador ao aplicar uma gramática.

O MUG foi projetado para realizar três tarefas principais:

- Divisão multimodal (*multimodal fission*) – distribuir a saída por vários modos de interação.
- Planeamento de frases – escolher a informação a incluir na saída gerada.
- Realização de linguagem natural e de interface gráfica com o utilizador – gerar a saída, levando em consideração o modo.

3.4.2 Implementação

Dentre as ferramentas analisadas, o MUG foi escolhido para ser utilizado na implementação do agente NLG do Carl devido às seguintes características:

- Utilizar como linguagem o Prolog.
- Permitir múltiplos modos de apresentação da(s) frase(s) gerada(s).
- Possuir mecanismos de atribuição de pontuação (*score*) a cada uma das frases geradas.

Tabela 3.7: Exemplos de utilização do agente NLG

Chamada	nlg(tell, _, _, [name(i77, carl), instance(i77, robot)], Frase)
Frase	“Carl is a robot”
Chamada	nlg(tell, _, _, [pronoun(i82, he), function(i82, manager), association(function_object, i82, i75), name(i75, ieeta)], Frase)
Frase	“He is the manager of Ieeta”
Chamada	nlg(ask, what, X, [name(i72, canada), instance(i72, X)], Frase)
Frase	“What is Canada?”

- Possuir a ferramenta gráfica *MUG Workbench* para inspeção da geração.

O agente NLG foi desenvolvido na linguagem Prolog, tendo como base o MUG⁸. Para seu funcionamento, precisa da semântica da frase a ser gerada. Esta semântica utiliza a linguagem de representação descrita na seção 3.2.2. Este agente registra o seguinte procedimento como capacidade no agente facilitador:

nlg (SpeechAct, SpeechActType, Main, Semantics, Sentence)

Esse procedimento é chamado pelo agente Gestor para iniciar o processo de geração. Aqui, *SpeechAct* é o ato de fala da frase a ser gerada. *SpeechActType* é usado para especificar alguns atos de fala. No caso de *ask*, podemos ter: *where*, *what* ou *who*. *Main* identifica a principal entidade da pergunta. *Semantics* contém as relações semânticas que caracterizam a frase a ser gerada.

Antes de retornar para o agente Gestor a frase gerada em *Sentence*, o agente NLG aciona o agente de síntese de fala com o texto a ser proferido.

A tabela 3.7 mostra alguns exemplos de utilização do agente NLG e das frases geradas. Vale a pena ressaltar que a decisão de utilizar pronomes é tomada pelo gestor de diálogo, com base no contexto armazenado no Estado da Informação. Mais detalhes serão apresentados no capítulo 4, seção 4.8.

⁸Como dito na seção 3.1.2.3, a biblioteca Prolog do OAA não pôde ser usada por incompatibilidade entres versões. Desta forma, este agente utiliza a biblioteca C do OAA através da interface C do SWI-Prolog.

Capítulo 4

O Gestor de Diálogo

Após a implementação da nova arquitetura computacional, do sistema de aquisição e gestão de conhecimento e da transformação dos demais módulos em agentes nessa arquitetura, o último e essencial componente a ser desenvolvido era um novo gestor de diálogo.

Este capítulo mostra as fases de desenvolvimento desse agente, os requisitos levantados para o sistema, sua estrutura geral, a abordagem escolhida para o gestor e também uma descrição de seus componentes.

4.1 Requisitos para o Gestor de Diálogo

O gestor de diálogo deverá permitir ao Carl participar de diálogos como o exemplo apresentado na tabela 4.1, no qual ocorrem elipses, pronomes e clarificações.

Deste cenário extraem-se os seguintes requisitos:

- Contexto do diálogo – é preciso manter um contexto do diálogo para que seja possível resolver referências e selecionar ações mais apropriadas.
- Resolução de referências – para a compreensão de pronomes com base no contexto do diálogo (a utilização de *he* e *you* no turn 5 do Utilizador).
- Resolução de elipses – para a compreensão de frases incompletas, ou seja, frases com uma ou mais palavras omitidas (turn 2 do Utilizador: “Jim”).
- Diálogos de clarificação para resolução de referências – quando a resolução de referências não for capaz de resolver um ou mais pronomes (turn 4 do Carl: “Who do you mean?”).
- Diálogos de clarificação para confirmação – quando o nível de confiança do reconhecimento automático de fala for baixo (turn 7 do Carl: “Did you say...”).

Tabela 4.1: Diálogo fictício com o robô Carl, o qual representa o objetivo/visão para o sistema de diálogo.

Utilizador 1:	Hi!
Carl 1:	Hi! I'm the robot Carl.
Carl 2:	What's your name?
Utilizador 2:	Jim.
Carl 3:	What can I do for you?
Utilizador 3:	Is he in Ieeta?
Carl 4:	Who do you mean?
Utilizador 4:	Peter.
Carl 5:	No, he is in the department.
Utilizador 5:	Does he like you?
Carl 6:	Yes.
Utilizador 6:	Mary lives in Spain.
Carl 7:	Did you say that Mary lives in Spain?
Utilizador 7:	Yes.
Carl 8:	Really? Daniel lives in Spain too!

Carl 9:	Do you work in Ieeta?
Utilizador 8:	Yes.
Carl 10:	What is your project?
Utilizador 9:	CAMBADA
Carl 11:	Did you say CAMBADA?
Utilizador 10:	Yes.
Carl 12:	Oh, so you work with Mark?
Utilizador 11:	Yes.
Carl 13:	Do you want to talk about hobbies?
Utilizador 12:	Yes.
Carl 14:	Sorry, but I'm afraid I'm running out of battery. Please charge me.

- Conhecimento comum – o sistema deve deixar claro que compreendeu o que foi dito pelo utilizador, por forma a criar conhecimento comum, ou seja, conhecimento partilhado pelos dois interlocutores (*grounding*).
- Iniciativa Mista – tanto o utilizador como o robô devem poder ter a iniciativa no diálogo (turn 1 do Utilizador: “Hi!” e turn 9 do Carl: “Do you work in Ieeta?”). Dessa forma, o robô deve ser capaz de iniciar novos diálogos bem como participar em diálogos iniciados pelo utilizador.
- Suporte a vários domínios – o robô deve ser capaz de falar sobre vários domínios e de mudar dinamicamente (turn 13 do Carl: “Do you want...”), assim, o gestor de diálogo não poder ser restrito a um único domínio.

4.2 Arquitetura do Gestor de Diálogo Desenvolvido

Dentre as abordagens apresentadas no capítulo 2, decidimos adotar a de Estado da Informação (*information state*) para gestão do diálogo. Esta abordagem é bem mais flexível e avançada do que as baseadas em estados finitos e do que as baseadas em formulários, ampliando assim, os cenários possíveis de utilização do robô Carl.

Essa escolha foi motivada pela falta de corpora e pela necessidade de ter um sistema não muito complexo sem comprometer a satisfação mínima dos requisitos levantados. Por isso, não foram considerados como opção viável os sistemas de diálogo que utilizam modelos de crenças, desejos e intenções, assim como os sistemas baseados em processos de decisão markovianos.

A abordagem à gestão de diálogo baseada em Estado da Informação, cuja estrutura geral é apresentada na figura 4.1, possui os seguintes componentes principais:

- Estado da Informação – estrutura de dados que contém todas as informações sobre eventos externos e sobre o estado e histórico do diálogo (ver seção 4.3).
- Recepção de eventos e condições que influenciam na habilitação de regras (ver seção 4.4).
- Despacho de ações para execução (sobre as ações, ver seção 4.5).
- Regras de atualização – definem a forma como o Estado da Informação, incluindo a lista de tarefas, é atualizado em cada iteração do ciclo de controle; sobre as tarefas ver a seção 4.6 e sobre as regras ver a seção 4.7.
- Algoritmos de controle – aplicam as regras de atualização pertinentes e extraem da lista de tarefas a próxima ação a ser executada (ver seção 4.8).

Podemos notar, pela figura 4.1, que o sistema de diálogo interage diretamente com os seguintes agentes:

- Navigation (navegação do robô no ambiente) e GTI (interface gráfica e tátil), dos quais recebe eventos, e para os quais despacha ações a executar.
- NLU (compreensão de linguagem natural), do qual recebe eventos.
- NLG (geração de linguagem natural) e ASR (reconhecimento automático de fala), para os quais despacha ações a executar.

Dentro do próprio agente gestor, existe ainda interação direta com o módulo KAM (aquisição e gestão do conhecimento).

Os eventos gerados pelos agentes GTI, NLU e Navigation, assim como eventos temporais gerados no próprio agente gestor, desencadeiam atualizações no Estado da Informação. O

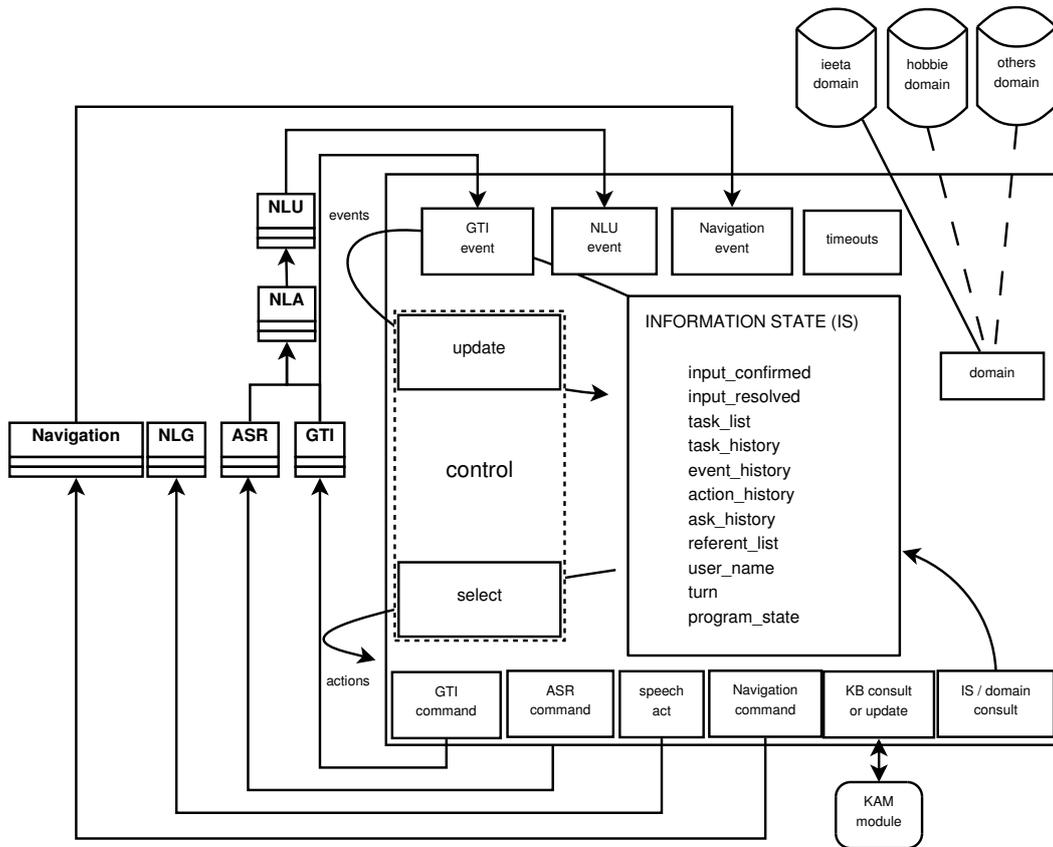


Figura 4.1: Gestor de diálogo e sua interação com os demais agentes da arquitetura computacional

módulo de controle deve escolher quais regras de atualização aplicar. No passo seguinte, deve selecionar as ações a executar.

Assim, cada iteração do ciclo de controle do sistema pode ser resumida em quatro passos principais:

1. Receber os eventos.
2. Atualizar o Estado da Informação.
3. Selecionar as ações.
4. Agir.

O diagrama de classes apresentado na figura 4.2 mostra, utilizando a notação UML, como se relacionam as várias entidades envolvidas no gestor de diálogo. Na leitura deste diagrama, é importante observar a cardinalidade associada aos relacionamentos. Por exemplo, em “evento habilita regra”, temos uma cardinalidade 1..* para 1..*. Com isso, a interpretação deve ser a de que um evento habilita uma ou várias regras e de que uma regra é habilitada por um

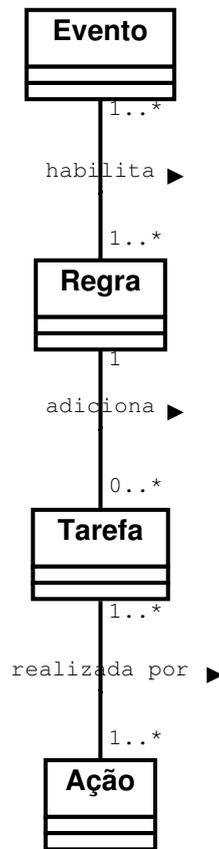


Figura 4.2: Diagrama de classes do gestor de diálogo

ou vários eventos. Em “regra adiciona tarefa” a cardinalidade é de **1** para **0..***. Isto significa que uma regra pode ou não adicionar tarefas à lista de tarefas, e uma tarefa só pode ser adicionada por uma regra. Por último, vemos que as tarefas são executadas através de ações, e estas podem estar em mais de uma tarefa.

4.3 Estado da Informação

O sistema de diálogo do robô Carl vai precisar de gerir tarefas, eventos e ações, bem como manter informações relacionadas, por exemplo quem está de posse da palavra (*turn*). Também é necessário armazenar dados de suporte ao controle do fluxo de processamento dos eventos provenientes do agente de compreensão de linguagem natural (NLU). Além disso, uma lista com os referentes mais recentes ordenados pelo grau de saliência é necessária para a resolução de referências.

Toda essa informação é concentrada no ‘Estado da Informação’. Para um melhor entendimento do leitor, a figura 4.3 mostra o Estado da Informação como uma classe com atributos (ou campos) e métodos de acesso. A tabela 4.2 lista os campos que formam o Estado da In-

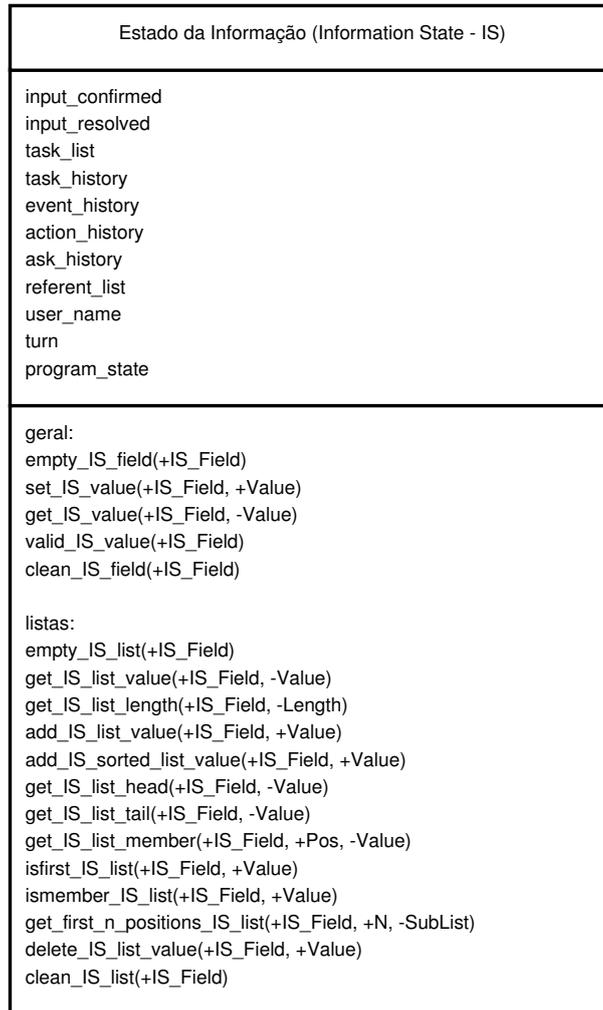


Figura 4.3: Estado da Informação como classe

formação com os seus respectivos tipos indicados e uma breve descrição. Para uma descrição dos métodos de acesso ao Estado da Informação, favor consultar o apêndice A.

4.4 Eventos e Condições

Do ponto de vista do sistema de gestão de diálogo, eventos são acontecimentos que interferem no diálogo, condicionando as decisões tomadas pelo gestor de diálogo. Como se verá na seção 4.7.3, os eventos são levados em consideração nas pré-condições das regras de atualização do Estado da Informação. Podemos dividir os eventos em dois tipos:

1. Mensagens de outros agentes da arquitetura computacional do Carl.
2. Temporizadores do próprio sistema de gestão de diálogo.

Tabela 4.2: Descrição dos campos do Estado da Informação

Campo	Tipo	Descrição
input_confirmed	estrutura	Armazena o resultado da confirmação da semântica de uma frase processada pelo agente NLU. Para mais detalhes, ver seção 4.7.3 e figura 4.4.
input_resolved	estrutura	Armazena o resultado da resolução de pronomes de uma frase já confirmada. Para mais detalhes, ver seção 4.7.3 e figura 4.4.
task_list	lista	Armazena as tarefas a serem realizadas pelo robô (ver detalhes na seção 4.6).
task_history	lista	Mantém um histórico das últimas tarefas realizadas pelo robô.
event_history	lista	Mantém um histórico dos últimos eventos recebidos.
action_history	lista	Mantém um histórico das últimas ações executadas pelo robô.
ask_history	lista	Mantém um histórico das últimas perguntas realizadas pelo robô.
referent_list	lista	Mantém a lista dos referentes das últimas quatro frases para apoio à resolução de pronomes. Para mais detalhes, ver seção 4.7.3.3.
user_name	cadeia de caracteres	Armazena o nome da pessoa que interage com o robô.
turn	cadeia de caracteres	Identifica quem está de posse da palavra; possíveis valores: <i>user</i> e <i>system</i> .
program_state	cadeia de caracteres	Controla a execução do gestor de diálogo; possíveis valores: <i>stop</i> e <i>run</i> .

Os eventos que podem ocorrer encontram-se listados na tabela 4.3. No sistema desenvolvido, o gestor de diálogo recebe mensagens dos agentes NLU, GTI e Navigation. De entre estas, a mais importante é a enviada pelo agente NLU. Esta mensagem indica que um utilizador interagiu com o robô e que foi extraída uma semântica dessa interação. Como foi dito na seção 3.2.4, a representação semântica dessa informação possui a seguinte estrutura: $sem(SA, SA_data, Main, Rels)$.

Além dos eventos, diversas condições internas influenciam a habilitação das regras. Estas condições podem ser agrupadas em:

- Condições sobre campos do Estado da Informação – Algumas regras verificam se determinado campo está vazio, ou se possui um valor específico ou mesmo um valor qualquer. Todas estas condições estão implementadas como métodos do Estado da Informação (ver figura 4.1 e o anexo A). Como exemplo deste tipo de condição, temos:

Tabela 4.3: Descrição dos eventos: mensagens de agentes ou temporizadores

Tipo	Evento	Descrição
mensagem	nlu_event (RecConf, AnalysisType, Semantics)	<i>RecConf</i> indica a confiança do Reconhecimento Automático de Fala (ASR) relativamente a uma dada frase, <i>AnalysisType</i> o tipo de análise, que pode ser <i>deep</i> ou <i>shallow</i> , e <i>Semantics</i> é a semântica da frase.
mensagem	gti_event (GTIEvent)	<i>GTIEvent</i> (cadeia de caracteres) identifica um evento gerado no agente GTI.
mensagem	navigation_event (NavigationEvent)	<i>NavigationEvent</i> (cadeia de caracteres) identifica um evento de navegação.
temporizador	timeout (TimeoutEvent)	<i>TimeoutEvent</i> (cadeia de caracteres) identifica um temporizador que atingiu o limite.

Tabela 4.4: Condições verificadas internamente

Condição	Descrição
resolvable(sem(SA, SA_data, Main, Relations), ReferentList, SemanticsResolved, _)	Verifica se a resolução de pronomes pode ser realizada numa determinada semântica, de acordo com a atual lista de referentes.
able_to_generate_ask_from_kb (AskHistory)	Verifica se é possível gerar uma pergunta ao utilizador com as informação da base de conhecimento atual. Ver procedimento <i>kb_generate_ask</i> na seção 3.3.4
low_confidence_limit_reached (EventHistory, 3)	Testa se o nível de confiança no reconhecimento do ASR foi baixo em 3 das últimas frases reconhecidas.

valid_IS_value(input_resolved).

- Outras condições – a tabela 4.4 apresenta outras condições cuja verificação está suportada no sistema desenvolvido, condições estas relacionadas com a possibilidade de colocar perguntas ao utilizador, ou com a resolução de pronomes, ou ainda com o nível de confiança do reconhecedor.

4.5 Ações

Ações são a resposta do sistema de diálogo a eventos ocorridos. Podem ser classificadas em três tipos:

1. Ações simples – Ações executadas pelo próprio agente Gestor, que podem ser dos seguintes tipos:
 - Operações de manipulação do Estado da Informação (ver seção 4.3)
 - Operações do módulo KAM para atualização e consulta da base de conhecimento do robô (ver seção 3.3.4)
 - Outras ações simples de apoio à gestão do diálogo
2. Ações OAA – Ações executadas por outros agentes na arquitetura OAA, sem necessidade de resposta. Do ponto de vista do agente Gestor, estas ações consistem no envio de mensagens aos demais agentes (ver seção 3.1.2 e tabela 3.1).
3. Ações OAA com espera – Ações executadas por outros agentes na arquitetura OAA, com necessidade de esperar por uma resposta. Neste caso, está o envio da mensagem $asr(+Command, -Value)$, que aguarda o retorno de um comando enviado para o agente ASR (ver também seção 3.1.2 e tabela 3.1).

4.6 Tarefas

O conjunto de tarefas condiciona diretamente o comportamento do robô. A cada tarefa está associada uma prioridade, atribuída estaticamente no momento de sua definição. Cada tarefa contém um plano que rege a sua execução. Desta forma, mais concretamente, uma tarefa é composta pelos seguintes elementos:

- $Tarefa(Arg_1, \dots, Arg_n)$ – $Tarefa$ define o nome da tarefa; Arg_1, \dots, Arg_n , com $n \geq 0$, são os argumentos da tarefa, que podem ser de entrada (indicados com o símbolo '+'), ou de saída (símbolo '-').
- prioridade – um número de 0 a 15 que indica o grau de importância da tarefa, com os números mais altos significando prioridade mais elevada.
- plano – uma sequência de operações que realiza efetivamente a tarefa. Uma lista com uma ou mais das seguintes operações:
 - Tarefa – identificada pelo construtor $task$
 - Ação Simples – identificada pelo construtor $action$
 - Ação OAA – identificada pelo construtor oaa_action
 - Ação OAA com espera – identificada pelo construtor oaa_wait_action
 - Operador $if_then(A,B)$ – onde A é uma condição e B uma sequência de operações

- Operador *if_then_else(A,B,C)* – onde *A* é uma condição, *B* e *C* são sequências de operações
- Operador *wait_condition(Cond, Timeout)* – espera que a condição *Cond* torne-se verdadeira pelo tempo definido em *Timeout*
- Operador *wait_condition(Cond, Timeout, ActionIfTimeout)* – espera que a condição *Cond* torne-se verdadeira pelo tempo definido em *Timeout*. Executa *ActionIfTimeout*, caso o tempo definido seja atingido.

Além dos métodos gerais de acesso ao Estado da Informação, que se encontram descritos no anexo A, foram por conveniência definidos alguns procedimentos específicos para tarefas, que manipulam diretamente o campo *task_list* do Estado da Informação, como por exemplo *task_list(-TaskList)*, equivalente a *get_IS_value(task_list, -TaskList)*.

A lista com todos os procedimentos criados pode ser encontrada no anexo B.

4.6.1 Exemplos de Tarefas

Esta seção apresenta as tarefa mais relevantes para a compreensão do gestor de diálogo. Outras tarefas podem ser vistas no anexo B.

4.6.1.1 Armazenamento de Informação

A tarefa apresentada em 4.1 mostra que para atualizar a base de conhecimento (procedimento *kb_tell*), é preciso saber o nome do utilizador. Isto se deve ao fato do módulo de gestão de conhecimento associar a cada informação o nome de quem a forneceu. Observe que, após acrescentar a informação à base de conhecimento, o procedimento *kb_generate_comment* é chamado para tentar gerar um comentário sobre a mesma. Se isto for possível, o comentário é dito pelo robô, caso contrário, um retorno mais geral é falado para que o utilizador perceba que o robô armazenou a informação.

Tarefa : store_info(+RecConf, +Semantics)

Prioridade : 5

$$\text{Plano : } \left\{ \begin{array}{l} \text{task(get_user_name(UserName))} \\ \text{action(kb_tell(UserName, RecConf, Semantics))} \\ \text{action(kb_generate_comment(Semantics, CommentSemantics,} \\ \quad \text{CommentConfidence, Status))} \\ \text{if_then_else(} \\ \quad \text{Status = ok,} \\ \quad [\text{oaa_action(nlg(observation, release_turn, _, _, Comment,} \\ \quad \quad \text{CommentConfidence))}], \\ \quad [\text{oaa_action(nlg(feedback, release_turn, _, _, _, 1))}] \\ \text{)} \end{array} \right. \quad (4.1)$$

4.6.1.2 Resposta a Questões Gerais

Recorrendo ao procedimento *kb_ask*, a tarefa *answer_question* responde a uma dada pergunta de acordo com a informação que foi adquirida ao longo do diálogo. Note que, além da resposta propriamente dita, este procedimento devolve uma confiança associada à mesma e também um campo de *Status*. Se este campo for igual a *ok*, significa que uma resposta foi encontrada e ela é encaminhada ao agente NLG. Caso contrário, uma mensagem de desculpas é utilizada em seu lugar, como podemos ver em 4.2.

Tarefa : answer_question(+Semantics)

Prioridade : 5

$$\text{Plano : } \left\{ \begin{array}{l} \text{action(kb_ask(Semantics, Reply, Confidence, Status))} \\ \text{if_then_else(} \\ \quad \text{Status = ok,} \\ \quad [\text{oaa_action(nlg(tell, release_turn, _, _, Reply, Confidence))}], \\ \quad [\text{oaa_action(nlg(sorry, release_turn, _, _, [], Confidence))}] \\ \text{)} \end{array} \right. \quad (4.2)$$

4.6.1.3 Resposta a Questões de Confirmação

O plano desta tarefa utiliza o procedimento *kb_ask_if* para responder a uma questão de confirmação (com resposta do tipo sim/não). Se a resposta for positiva, a próxima ação é o envio desta informação ao agente NLG para ser comunicada ao utilizador. Entretanto, se a resposta for negativa, o procedimento *kb_alternate_reply* é chamado para modificar a

pergunta inicial em uma questão aberta que possa ser respondida com as informações da base de conhecimento atual. Se isto for possível, a nova resposta é encaminhada ao agente NLG, caso contrário, a resposta inicial (negativa) é utilizada. A tarefa é mostrada em 4.3.

Tarefa : `answer_yesno_question(+Semantics)`

Prioridade : 5

Plano : {

```

    action(kb_ask_if(Semantics, Reply, Confidence))
    if_then_else(
        Reply = yes,
        [oaa_action(nlg(confirm, release_turn, _, _ , Reply, Confidence))]
        [kb_alternate_reply(Semantics, NewReply, NewConfidence,,
            Status)
        if_then_else(
            Status = ok,
            [oaa_action(nlg(deny, keep_turn, _, _ , _ , Confidence)),
            oaa_action(nlg(tell, release_turn, _, _ , NewReply,,
                NewConfidence))]
            [oaa_action(nlg(sorry, release_turn, _, _ , Reply, Confidence))]
        )]
    )

```

(4.3)

4.6.1.4 Clarificação – Baixa Confiança do Agente de Reconhecimento de Fala Automático

A confirmação de algo que foi escutado com uma baixa confiança do Automatic Speech Recognition (ASR) é tratada pela tarefa 4.4, *confirm_input*. Observe que seu plano consiste em perguntar ao utilizador se a semântica entendida é correta e, de acordo com sua resposta, colocar a informação no campo *input_confirmed* ou adicionar uma tarefa que peça a repetição da informação. As regras que determinam o fluxo global de processamento dos eventos gerados pelo agente NLU serão apresentadas na seção 4.7.

Tarefa : confirm_input(+RecConf, +Semantics)

Prioridade : 5

Plano : {

```
oaa_action(nlg(confirmation, release_turn, _, _, Semantics, _))
wait_condition(nlu_event(RecConfReply, _, sem(SA, _, _, _)), 30)
if_then_else(
  SA = confirm,
  [action(set_IS_value(input_confirmed, input(RecConf, ,
    Semantics)))]
  [task(repeat_info)]
)
```

(4.4)

4.6.1.5 Clarificação – Resolução de Pronomes

A tarefa 4.5 trata da clarificação da resolução de pronomes que não foi possível apenas com o contexto do diálogo. Aqui, uma pergunta é feita para clarificar a quem o utilizador se referia por determinado pronome. O procedimento *generate_disambiguating_ask* é responsável por gerar a semântica dessa pergunta. Logo após, uma mensagem é enviada ao agente NLG, que trata de transformar a semântica em texto e de encaminhá-la ao agente de síntese. A seguir, o plano desta tarefa fica à espera da resposta do utilizador por 30 segundos (*wait_condition...*). De posse desta resposta, é verificado se a semântica já pode ser resolvida – isto é feito pelo procedimento *resolvable_with_reply*. Caso positivo, o campo *input_resolved* é atualizado com a nova informação. Caso contrário, é adicionada uma tarefa que pede a repetição da informação inicial.

Tarefa : clarify_pronouns(+Semantics, +UnresPNoun)

Prioridade : 5

$$\text{Plano : } \left\{ \begin{array}{l} \text{action(generate_disambiguating_ask(UnresPNoun, AskType, Ask))} \\ \text{oaa_action(nlg(clarify, release_turn, AskType, _, Ask, _))} \\ \text{wait_condition(get_IS_value(input_clarified,} \\ \qquad \qquad \qquad \text{input(RecConf, sem(reply, _, _, ReplyRels)), 30)} \\ \text{if_then_else(} \\ \qquad \text{resolvable_with_reply(Semantics, ReplyRels, ,} \\ \qquad \qquad \qquad \text{SemanticsResolved))} \\ \qquad \text{[action(set_IS_value(input_resolved, input(RecConf, ,} \\ \qquad \qquad \qquad \text{SemanticsResolved)))]} \\ \qquad \text{[task(repeat_info)]} \\ \text{)} \end{array} \right. \quad (4.5)$$

4.6.1.6 Tomada de Iniciativa

A tarefa 4.6 gera uma pergunta a partir das informações existentes na base de dados naquele instante. Para evitar repetições, o histórico de perguntas e o utilizador corrente são levados em consideração. Após realizada a pergunta, o plano desta tarefa fica à espera de uma resposta do utilizador por 30 segundos. Observe que, de posse desta resposta, é chamado o procedimento *form_semantics*. Isto é necessário para lidar com elipses, ou seja, com respostas abreviadas, que não chegam a formar uma frase completa. Se o procedimento for capaz de formar uma semântica com a resposta obtida, então esta informação é armazenada através da tarefa *store_info*. Caso contrário, a resposta é rejeitada.

Tarefa : ask(AskHistory)

Prioridade : 3

```
Plano : {
  task(get_user_name(UserName))
  action(kb_generate_ask(UserName, AskHistory, RVar, AskType,
    Ask, Status))
  if_then(
    Status = ok,
    [action(add_IS_list_value(ask_history, ask(UserName, AskType,
      Ask)))
    oaa_action(nlg(ask, release_turn, AskType, _, Ask, _)),
    wait_condition(get_IS_value(input_resolved,
      input(RecConfReply, sem(SA, _, _, ReplyRels),
        _)), 30),
    if_then_else(
      (SA = tell; SA = reply),
      action(form_semantics(Ask, RVar, ReplyRels, FinalRels,
        Status2)),
      if_then_else(
        Status2 = ok,
        [task(store_info(RecConfReply, FinalRels))],
        [task(reject)]
      )],
    [task(reject)]
  )
}
```

(4.6)

4.7 Regras de Atualização

Como mostrado anteriormente (subseção 2.3.5 e seção 4.2), os sistemas de diálogo baseados no Estado da Informação atualizam a sua estrutura de dados central de acordo com determinadas regras. Aqui, iremos primeiramente definir o conceito de regra, tal como adotado no presente trabalho, e depois apresentaremos exemplos.

4.7.1 Estrutura de uma Regra

Uma regra é uma estrutura utilizada para controlar as atualizações do Estado de Informação e é constituída por três partes definidas abaixo:

1. Regra – define o nome da regra.
2. Lista de Condições (LC) – indica as condições necessárias para que a regra possa ser aplicada, geralmente associadas a eventos ou valores do Estado da Informação.
3. Lista de Efeitos (LE) – lista de procedimentos que atualizam o Estado da Informação. Tais procedimentos podem ser:

- `add_task(Task)` – adiciona uma tarefa à lista de tarefas.
- `delete_task(+Task)` – remove uma tarefa da lista de tarefas.
- `set_IS_value(+IS_Field, +Value)` – atribui um valor a um campo do Estado da Informação.
- `clean_IS_field(+IS_Field)` – limpa um campo do Estado da Informação.

4.7.2 Exemplos de Regras Gerais

Esta seção apresenta as regras gerais consideradas mais importantes para o entendimento do gestor de diálogo. Para outras regras, favor ver o anexo B, seção B.2.

4.7.2.1 Tomada de Palavra

A regra 4.7 atribui a palavra ao sistema. É aplicada quando o utilizador possuía a palavra e disse alguma coisa.

Regra : Turn

$$\begin{aligned} \text{LC : } & \left\{ \begin{array}{l} \text{nlu_event(RecConf, Analysis, Semantics)} \\ \text{get_IS_value(turn, TurnValue)} \\ \text{TurnValue = user} \end{array} \right. & (4.7) \\ \text{LE : } & \left\{ \text{set_IS_value(turn, system)} \right. \end{aligned}$$

4.7.2.2 Armazenamento de Informação

A regra *storeInfo* é aplicada quando existe um valor válido no campo *input_resolved* cuja semântica possui um ato de fala *tell*. Tem como efeito adicionar a tarefa *store_info* à lista de tarefas, como vemos em (4.8).

Regra : storeInfo

$$\begin{aligned}
 \text{LC} : & \left\{ \begin{array}{l} \text{valid_IS_value(input_resolved)} \\ \text{get_IS_value(input_resolved, input(RecConf, sem(SA, SA_data, _, Rels)))} \\ \text{SA = tell} \end{array} \right. \\
 \text{LE} : & \left\{ \text{add_task(store_info(RecConf, Rels)} \right.
 \end{aligned}
 \tag{4.8}$$

4.7.2.3 Iniciativa do Diálogo

A tomada de iniciativa pode acontecer em vários pontos do diálogo. Aqui mostraremos a regra que gera pergunta quando o diálogo encontra-se inativo. Estas perguntas são geradas a partir do conhecimento adquirido ao longo do diálogo.

A regra 4.9 mostra uma atualização que é aplicada quando um limite de tempo é superado na interação, a lista de tarefas encontra-se vazia e o sistema é capaz de gerar uma pergunta com o conhecimento adquirido. Esta última restrição é verificada pelo procedimento *able_to_generate_ask_from_kb*. Neste caso, a regra é aplicada e a tarefa *ask* adicionada à lista de tarefas. Note que o histórico de perguntas é passado neste momento.

Regra : ask

$$\begin{aligned}
 \text{LC} : & \left\{ \begin{array}{l} \text{timeout(interaction)} \\ \text{empty_IS_field(task_list)} \\ \text{get_IS_value(ask_history, AskHistory)} \\ \text{able_to_generate_ask_from_kb(AskHistory)} \end{array} \right. \\
 \text{LE} : & \left\{ \text{add_task(ask(AskHistory))} \right.
 \end{aligned}
 \tag{4.9}$$

4.7.2.4 Ambiente Ruidoso

A regra 4.10 identifica quando o desempenho do ASR é baixo no evento atual e também foi baixo nos últimos 3 eventos. Esta situação é típica de ambiente ruidoso, por isso é adicionada uma tarefa para sugerir a mudança de ambiente.

Regra : noisyEnvironment

$$\begin{aligned}
 \text{LC : } & \left\{ \begin{array}{l} \text{nlu_event(RecConf, Analysis, Semantics)} \\ \text{get_IS_value(event_history, EventHistory)} \\ \text{RecConf} < 70 \\ \text{low_confidence_limit_reached(EventHistory, 3)} \end{array} \right. & (4.10) \\
 \text{LE : } & \left\{ \text{add_task(change_place)} \right.
 \end{aligned}$$

4.7.3 Regras para Diálogos de Clarificação

A figura 4.4 mostra o fluxo das atualizações possíveis no Estado da Informação em decorrência de um evento gerado pelo agente Natural Language Understanding (NLU). As tarefas podem ser encontradas na seção 4.6 e as regras são apresentadas em seguida. Se a confiança no reconhecimento devolvida pelo agente ASR for muito baixa (menos de 30%), a informação é rejeitada e a tarefa *reject* trata desta situação. A extração da semântica pode ser feita de forma superficial (*shallow*), usando o Tilburg Memory Based Learner (TiMBL), ou de forma aprofundada (*deep*), usando o LCFlex (mais detalhes no capítulo 3, seção 3.2). Se a análise tiver sido feita de forma superficial e a confiança do ASR estiver acima de 30%, então é adicionada uma tarefa de clarificação de semântica superficial. Se a análise tiver sido feita de forma aprofundada e a confiança do ASR estiver entre 30% e 70%, é adicionada uma tarefa de confirmação para que a informação possa chegar até ao campo *input_confirmed*. Se a confiança estiver acima dos 70%, a informação é passada diretamente para esse campo.

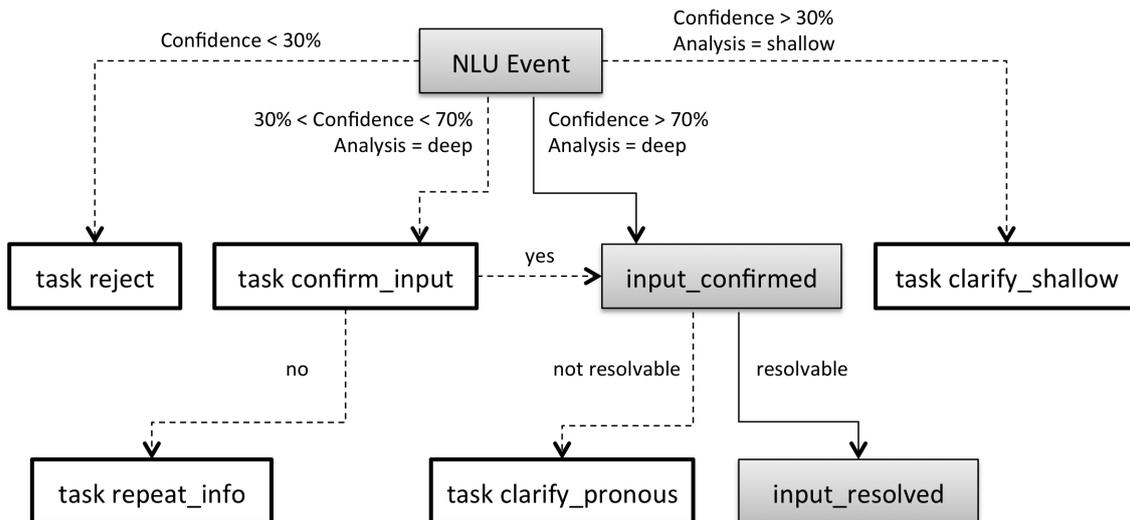


Figura 4.4: Fluxo das atualizações

Quando existe uma informação válida no campo *input_confirmed*, é invocada a resolução

de pronomes. Se esta correr sem problemas, a informação chega até ao campo *input_resolved*. Entretanto, a resolução de pronomes pode falhar por dois motivos: a lista de referentes do Estado da Informação não possui elementos compatíveis ou possui mais de um candidato. Nestes casos, a tarefa de clarificação de pronomes é adicionada.

A tabela 4.5 apresenta as regras relacionadas diretamente a um evento gerado pelo agente NLU (estas regras podem ser vistas também na figura 4.4). A regra *asrLowConfidence* identifica quando o nível de confiança do agente de reconhecimento automático de fala se encontra muito baixo, abaixo de 30%, e adiciona a tarefa de rejeição.

Tabela 4.5: Regras relacionadas aos eventos gerados pelo agente NLU

Regra	Lista	Condições/Efeitos
asrVeryLowConfidence	LC	nlu_event(RecConf, Analysis, Semantics) RecConf < 30
	LE	add_task(reject)
asrLowConfidence	LC	nlu_event(RecConf, Analysis, Semantics) Analysis = deep RecConf ≥ 30 RecConf > 70
	LE	add_task(confirm_input(RecConf, Semantics))
asrHighConfidence	LC	nlu_event(RecConf, Analysis, Semantics) Analysis = deep RecConf ≥ 70
	LE	set_IS_value(input_confirmed, input(RecConf, Semantics))
shallowAnalysis	LC	nlu_event(RecConf, Analysis, Semantics) RecConf ≥ 30 Analysis = shallow
	LE	add_task(clarify_shallow(Semantics))

A regra *asrLowConfidence* identifica quando a análise da semântica é aprofundada e o nível do reconhecedor encontra-se abaixo de um determinado limiar, mas não é tão baixo a ponto de ser rejeitado diretamente, isto é, encontra-se entre 30% e 70%. Neste caso, a tarefa de confirmação é adicionada à lista de tarefas.

A regra *asrHighConfidence* atualiza o Estado da Informação quando a análise da semântica tiver sido feita forma aprofundada (*deep*) e o nível de confiança do reconhecedor estiver acima de um determinado limiar, neste caso 70%. O campo do Estado da Informação modificado por esta regra é o *input_confirmed*.

A regra *shallowAnalysis* identifica quando a análise semântica é realizada de forma super-

ficial (*shallow*) e adiciona a tarefa *clarify_shallow(Semantics)* para clarificá-la.

4.7.3.1 Resolução de Pronomes Anafóricos

Pronome anafórico é aquele que faz referência a uma entidade que foi previamente apresentada no diálogo. Assim, o procedimento *resolvable* verifica se a resolução de pronomes anafóricos pode ser realizada numa determinada semântica, levando em consideração a atual lista de referentes recentes.

A regra *semanticsResolvable* é aplicada quando existe um valor válido no campo *input_confirmed* cuja semântica possa ser resolvida, exceto quando o tipo de ato de fala é *reply* (este caso é tratado por regras que lidam com a resposta de clarificação de pronomes). Como efeitos, atribui um valor ao campo *input_resolved* e torna vazio o campo *input_confirmed*. Esta regra pode ser vista em detalhes em 4.11, mas é possível ter uma visão global na figura 4.4, mostrada anteriormente.

Regra : *semanticsResolvable*

$$\begin{array}{l}
 \text{LC : } \left\{ \begin{array}{l}
 \text{valid_IS_value(input_confirmed)} \\
 \text{get_IS_value(input_confirmed, input(RecConf, sem(SA, SA_data,} \\
 \text{Main, Rels))} \\
 \text{get_IS_value(referent_list, ReferentList)} \\
 \text{SA} \neq \text{reply} \\
 \text{resolvable(sem(SA, SA_data, Main, Relations), ReferentList,} \\
 \text{SemanticsResolved, _)}
 \end{array} \right. \quad (4.11) \\
 \text{LE : } \left\{ \begin{array}{l}
 \text{set_IS_value(input_resolved, input(RecConf, SemanticsResolved))} \\
 \text{clean_IS_field(input_confirmed)}
 \end{array} \right.
 \end{array}$$

4.7.3.2 Clarificação de Pronomes Anafóricos

Apresentamos agora a regra *semanticsNotResolvable*, que lida com o caso de não ser possível resolver os pronomes anafóricos com a lista de referentes atual. Esta regra é aplicada quando existe um valor válido no campo *input_confirmed* cuja semântica não possa ser resolvida. Seus efeitos são o de tornar vazio o campo *input_confirmed* e o de adicionar a tarefa de clarificação de pronomes à lista de tarefas, como vemos em 4.12.

Regra : semanticsNotResolvable

$$\begin{array}{l}
 \text{LC : } \left\{ \begin{array}{l}
 \text{valid_IS_value(input_confirmed)} \\
 \text{get_IS_value(input_confirmed, input(RecConf, sem(SA, SA_data,} \\
 \text{Main, Rels))} \\
 \text{get_IS_value(referent_list, RList)} \\
 \text{SA} \neq \text{reply} \\
 \text{not(resolvable(sem(SA, SA_data, Main, Rels), RList, SemResolved,} \\
 \text{UnresPNoun))}
 \end{array} \right. \quad (4.12) \\
 \\
 \text{LE : } \left\{ \begin{array}{l}
 \text{add_task(clarify_pronouns(SemResolved, UnresPNoun))} \\
 \text{clean_IS_field(input_confirmed)}
 \end{array} \right.
 \end{array}$$

4.7.3.3 Lista de Referentes

A regra *updateReferentList* atualiza a lista de referentes e é aplicada quando existe um valor válido no campo *input_resolved*. Tem como efeito a chamada do procedimento *update_referent_list* e a atualização do campo *referent_list* no Estado da Informação, como pode ser observado em (4.13).

Regra : updateReferentList

$$\begin{array}{l}
 \text{LC : } \left\{ \begin{array}{l}
 \text{valid_IS_value(input_resolved)} \\
 \text{get_IS_value(input_resolved, input(RecConf, sem(SA, SA_data,} \\
 \text{_ , Rels))} \\
 \text{get_IS_value(referent_list, ReferentList)}
 \end{array} \right. \quad (4.13) \\
 \\
 \text{LE : } \left\{ \begin{array}{l}
 \text{update_referent_list(ReferentList, sem(SA, SA_data, _ , Rels,} \\
 \text{NewReferentList))} \\
 \text{set_IS_value(referent_list, NewReferentList)}
 \end{array} \right.
 \end{array}$$

O procedimento *update_referent_list*, mostrado no algoritmo 2, é uma versão simplificada do algoritmo descrito em [Lappin e Leass, 1994].

Cada referente possui um valor de saliência associado que é dado pela tabela 4.6. A lista deve ser ordenada em função destes valores, sendo o maior o mais representativo.

Estes valores são reduzidos à metade quando uma nova frase é considerada. Isto é feito para dar prioridade aos referentes mais recentes.

Outra característica deste procedimento é que a lista contém apenas os referentes mencionados nas últimas quatro frases.

O procedimento que efetivamente faz a resolução de pronomes é descrito pelo algoritmo 3, na página 102. Esse procedimento recebe uma semântica e uma lista de referentes, e, com

Algoritmo 2: update_referent_list(ReferentList, Semantics)

```
início
  para cada referent[i] em ReferentList faça
    se referent[i].age = 3 então
      delete(ReferentList, referent[i]) // remove referência antiga
    senão
      referent[i].age = referent[i].age + 1 // atualiza idade
      referent[i].salience = referent[i].salience/2 // reduz saliência
  TempList ← extract_referents(Semantics)
  NewReferentList ← merge(ReferentList, TempList)
  NewReferentList ← sort_by(NewReferentList, salience)
  retorna NewReferentList
fim
```

Tabela 4.6: Fatores de saliência utilizados pelo algoritmo 2

Fator	Valor
Frase recente	100
Sujeito	80
Objeto direto	50
Objeto indireto	40

isso, é capaz de substituir os pronomes não resolvidos por elementos da lista compatíveis.

Algoritmo 3: resolve(Semantics, ReferentList)

```
início
  UnresolvedList ← extract_unresolved(Semantics)
  para cada referring[i] em UnresolvedList faça
    TempReferentList ← remove_unmatched(ReferentList, referring[i])
    se TempReferentList = null então
      retorna null
    senão
      Referent ← get_first(TempReferentList)
      Semantics ← replace(Semantics, referring[i], Referent)
  retorna Semantics
fim
```

4.8 Controle do Gestor de Diálogo

Como dito no início deste capítulo, o módulo de controle é responsável por registrar os eventos gerados pelos agentes GTI, NLU e Navigation, assim como eventos temporais gerados internamente. A seguir, este módulo de controle aplica todas as regras de atualização possíveis de acordo como atual Estado da Informação. Por fim, ele seleciona a próxima ação a ser executada.

4.8.1 Módulo de Controle

O comportamento do módulo de controle global do gestor de diálogo é descrito pelo algoritmo 4. Seu primeiro passo é receber os eventos e adicioná-los em um histórico (mais detalhes na seção 4.4). Após isso, o Estado da Informação é atualizado pelas regras que tiverem sido habilitadas – a seção 4.8.2 apresenta este tópico. O próximo passo do algoritmo é um laço que se repete enquanto o sistema possuir a palavra e, claro, tiver ações a executar. Nesse laço, a primeira tarefa da lista de tarefas é considerada, conforme apresentado na seção 4.6, a lista está ordenada por prioridade. Dessa primeira tarefa, extrai-se a próxima ação a ser executada – uma descrição completa desse ponto é mostrada na seção 4.8.3. Se a ação não for nula, o algoritmo a executa e pega o valor atualizado de quem possui a palavra, *TurnValue*. Se a palavra for do utilizador – *TurnValue = user*, o laço é interrompido. Outra condição de parada do laço é a falta de ações a serem executadas – próxima ação nula (*NextAction = null*) e lista de tarefas vazia. Ao final desse laço, numa última etapa, o campo do Estado da informação que indica o estado do programa, *program_state*, é avaliado. Se o estado for de parada, *stop*, o algoritmo chega ao fim, senão, é retomado o primeiro passo de recebimento de eventos.

Algoritmo 4: control()

```
início
  repita
    Events ← receive_events() // ver seção 4.4
    add_IS_list_value(event_history, Events)
    perform_state_update() // ver seção 4.8.2
    repita
      get_IS_list_head(task_list, Task)
      NextAction ← get_next_action(Task) // ver seção 4.8.3
      se NextAction ≠ null então
        act(NextAction)
        get_IS_value(turn, TurnValue)
      até TurnValue = user ∨ (NextAction = null ∧ empty_task_list())
      get_IS_value(program_state, ProgramStateValue)
    até ProgramStateValue = stop
fim
```

4.8.2 Atualização do Estado da Informação

A atualização do Estado da Informação é o passo do algoritmo de controle que é executado logo após o recebimento dos eventos. Aqui, as regras de atualização que tenham as suas pré-condições satisfeitas serão aplicadas. O algoritmo 5 mostra como é feita esta atualização. Seu funcionamento é bastante simples, o algoritmo percorre as regras existentes e aplica os efeitos das regras que tenham suas pré-condições satisfeitas. É importante notar que o procedimento garante que todas as atualizações possíveis sejam executadas.

Algoritmo 5: `perform_state_update()`

```
início
  repita
    NumUpdatesApplied ← 0
    para cada rule[i] faça
      se check_conditions(rule[i]) então
        apply_effects(rule[i]) // aplica os efeitos da regra se as
          condições tiverem sido satisfeitas
        NumUpdatesApplied ← NumUpdatesApplied + 1
    até NumUpdatesApplied = 0
fim
```

4.8.3 Ação

O algoritmo 6 é responsável por indicar a próxima ação a ser executada. Para isso, o algoritmo analisa o plano da primeira tarefa na lista de tarefas. Conforme descrito na seção 4.6, um plano pode conter tarefas e também operadores condicionais, por isso cada passo do plano é tratado de acordo com o seu tipo. No início do algoritmo, é verificado se o plano está vazio, indicação de que se chegou ao seu final. Nesse caso, a tarefa é guardada em histórico e apagada. Se o plano não estiver vazio, avalia-se seu próximo passo. Por exemplo, se for um passo do tipo *if_then_else(A,B,C)*, a condição *A* é analisada, se for verdadeira, o novo plano é formado com a lista de operações *B* em seu início. Se a condição não for verdadeira, é colocada a lista *C* no início do novo plano. Note que as tarefas encontradas são adicionadas à lista de tarefas e desta forma podem ser executadas já na próxima chamada de *get_next_action()*. Vale a pena ressaltar que a prioridade da tarefa encontrada é alterada dinamicamente para um valor superior ao da tarefa que a adiciona. Isso garante que ela fique no início da lista de tarefas.

O procedimento *act(Action)* (algoritmo 7) recebe uma ação, executa-a e atualiza o histórico. Quando a ação é um ato de fala (identificado por *nlg*), o algoritmo procura utilizar pronomes (*replace_names_by_pronouns*) de acordo com a lista de referentes daquele momento. Além disso, também atualiza o campo *turn* do Estado da Informação.

Algoritmo 6: `get_next_action(Task)`

início

```
se Task.plan = null então
  // chegou ao final do plano
  add_IS_list_value(task_history, Task);
  delete_task(Task);
  retorna null
senão
  getfirst(Task.plan, Step); // pega próximo passo do plano
  seleccione Step faça
    caso if_then(A,B)
      remove_from_plan(Step, Task.plan);
      se A então
        Task.plan ← append(B, Task.plan); // B no início do plano
      retorna get_next_action(Task)
    caso if_then_else(A,B,C)
      remove_from_plan(Step, Task.plan);
      se A então
        Task.plan ← append(B, Task.plan); // B no início do plano
      senão
        Task.plan ← append(C, Task.plan); // C no início do plano
      retorna get_next_action(Task)
    caso wait_condition(Cond, Timeout, ActionIfTimeout)
      se Cond então
        remove_from_plan(Step, Task.plan);
        retorna get_next_action(Task)
      senão se reached_timeout(Timeout) então
        se ActionIfTimeout ≠ null então
          remove_from_plan(Step, Task.plan);
          retorna ActionIfTimeout // retorna ação para timeout
        senão
          delete_task(Task);
          retorna null
      retorna null
    caso task(NewTask)
      NewTaskOriginalPriority ← NewTask.Priority;
      TaskPriority ← Task.Priority;
      set_task_priority(NewTask, TaskPriority+1); // prioridade maior
      add_task(NewTask);
      set_task_priority(NewTask, NewTaskOriginalPriority);
      retorna null
    senão
      remove_from_plan(Step, Task.plan);
      retorna Step
```

fim

Algoritmo 7: act(Action)

início**selecione Action faça****caso** *action(oaa_action(nlg(SA, SA_data, release_turn, Main, Rels, Confidence)))*

| get_IS_value(referent_list, ReferentList)

| replace_names_by_pronouns(ReferentList, Rels, PronounRels)

| execute_action(nlg(SA, SA_data, release_turn, Main, PronounRels, Confidence))

| set_IS_value(turn, user)

senão

| execute_action(Action)

add_IS_list_value(action_history, Action)

fim

Capítulo 5

Avaliação

Com o desenvolvimento e integração do novo Gestor de Diálogo na arquitetura do Carl, descrito no capítulo 4, o sistema para interação com base em linguagem natural falada do Carl ficou completo. O próximo passo era a avaliação desse sistema e de seus componentes, objetivo principal deste capítulo.

Foram realizados dois tipos de avaliação. Primeiramente, foram analisadas de forma isolada diversas funcionalidades do sistema: a compreensão de frases declarativas; a aquisição de conhecimento e capacidade de resposta a perguntas sobre esse conhecimento; e a resolução de pronomes. Numa segunda fase, foi realizada uma avaliação global da interação com o robô, com ênfase na interação falada. Finalmente, foi também feita uma avaliação da usabilidade do robô com base nas respostas dadas a um questionário por um grupo de utilizadores.

5.1 Avaliação de Funcionalidades

Não sendo o foco da tese as tecnologias de base, como o reconhecimento de fala ou a síntese, avaliações a estas não foram incluídas neste capítulo. Informações sobre a avaliação do reconhecedor utilizado podem ser encontradas em [Seabra Lopes *et al.*, 2003b].

As avaliações descritas a seguir foram realizadas no robô Carl em um típico laboratório de pesquisa, portanto um ambiente com algum ruído de fundo.

5.1.1 Compreensão de Frases Declarativas

Neste primeiro conjunto de experiências, o sistema de compreensão de linguagem falada (SLU, de Spoken Language Understanding), que consiste nos agentes ASR, NLA e NLU combinados, e a capacidade de clarificação semântica do gestor de diálogo foram avaliados na compreensão de frases declarativas [Quinderé *et al.*, 2007b]. Esse tipo de frase foi escolhido por ser essencial para avaliação da aquisição de conhecimento por parte do robô.

5.1.1.1 Método

Um orador leu 41 frases, repetindo 3 vezes cada uma, perfazendo um total de 123 locuções com 312 relações. As frases foram selecionadas de maneira a gerar um mínimo de 20 ocorrências de cada tipo de relação semântica (*subtype, instance, name, function, association, composition*). “Tony is in Canada” e “The professor of Mary likes the company president” são dois exemplos de frases utilizadas. A lista completa encontra-se no Apêndice C. O leitor pode observar que elas são assertivas e concisas. O número médio de palavras por frase é de 5,07 e o número médio de relações semânticas que corretamente representam a semântica das frases é de 2,54. O Nuance 8.0 foi utilizado como sistema de reconhecimento automático de fala, com um modelo de linguagem baseado em trigramas, adaptado do modelo desenvolvido pelas universidades Carnegie Mellon e Cambridge. Esse modelo, originalmente construído a partir de um conjunto vasto de fontes e com uma abrangência de 64000 palavras, foi restringido às palavras contidas na experiência para diminuir o espaço de procura do reconhecedor e, com isso, obter um melhor desempenho do reconhecimento de fala.

As duas métricas de desempenho utilizadas, a precisão e a cobertura, são definidas em função dos conjuntos de relações relevantes e extraídas (ver figura 5.1), que formam, então, quatro subconjuntos: relevantes extraídas (RE), relevantes não extraídas (RN), não relevantes extraídas (NE) e não relevantes não extraídas (NN).



Figura 5.1: Conjuntos de relações extraídas e não extraídas, bem como relevantes e não relevantes

Precisão é a razão entre o número de relações corretamente extraídas pelo sistema (isto é, relações extraídas que são relevantes) e o número total de relações extraídas (equação 5.1).

$$Precisão = \frac{RE}{RE + NE} \quad (5.1)$$

Por sua vez, a *cobertura (recall)* é a razão entre o número de relações corretamente extraídas e o número de relações relevantes (equação 5.2).

$$Cobertura = \frac{RE}{RE + RN} \quad (5.2)$$

5.1.1.2 Resultados

Esta experiência foi realizada em dois cenários. No primeiro, foi usado o gestor de diálogo anterior, sem diálogos de confirmação. O segundo cenário utilizou o novo gestor com diálogos de confirmação.

A tabela 5.1 apresenta os resultados do primeiro conjunto de experiências num cenário sem diálogos de confirmação, utilizando o gestor de diálogo originalmente apresentado em [Seabra Lopes, 2002] (com resultados apresentados em [Seabra Lopes *et al.*, 2005b]). Mais informações sobre este gestor de diálogo encontram-se no capítulo 3, seção 3.1.1. Nesse cenário, o sistema obteve um valor médio de 84,3% para *Cobertura* e de 80,9% para *Precisão*. Note que o desempenho mais fraco para a precisão foi obtido para a relação *subtype*, ao passo que a pior cobertura foi da relação *composition*.

Tabela 5.1: Resultado da análise semântica de frases declarativas (SLU/DM) – sem confirmação

Relação	Relevantes	Corretas	Erradas	Precisão	Cobertura
Association	87	74	19	79,6%	85,1%
Composition	21	7	0	100,0%	33,3%
Name	93	82	1	98,8%	88,2 %
Instance	39	36	39	48,0%	92,3%
Subtype	21	21	1	95,5%	100,0%
Function	51	43	2	95,6%	84,3%
Total	312	263	62	80,9%	84,3%

A tabela 5.2 apresenta os resultados da análise semântica das frases num cenário com diálogos de confirmação, utilizando o gestor de diálogo descrito no capítulo 4 e originalmente apresentado em [Quinderé *et al.*, 2007b].

A tabela 5.3 mostra um resumo do desempenho da análise semântica dos dois gestores de diálogo avaliados. Como pode ser observado, o uso de diálogos de confirmação permitiu uma evolução de 6 pontos percentuais (p.p.) na *Cobertura* e 7 p.p. na *Precisão* das relações semânticas extraídas. Comparando as tabelas 5.1 e 5.2, pode-se ver que a cobertura da relação *composition* melhorou 24 p.p. e a precisão da relação *instance*, 12 p.p.

Tabela 5.2: Resultado da análise semântica de frases declarativas (SLU/DM) – com confirmação

Relação	Relevantes	Corretas	Erradas	Precisão	Cobertura
Association	87	79	10	88,8 %	90,8%
Composition	21	12	1	92,3%	57,1%
Name	93	89	2	97,8%	95,7%
Instance	39	38	25	60,3%	97,4%
Subtype	21	21	0	100,0%	100,0%
Function	51	44	0	100,0%	86,3%
Total	312	283	38	88,2%	90,7%

Tabela 5.3: Resumo da análise semântica de frases declarativas (SLU/DM)

Gestor de Diálogo	Relevantes	Corretas	Erradas	Precisão	Cobertura
Sem confirmação	312	263	62	80,9%	84,3%
Com confirmação	312	283	38	88,2%	90,7%
Diferença	-	+20	-24	+7,3 p.p.	+6,4 p.p.

5.1.2 Aquisição de Conhecimento e Resposta a Perguntas

Outra experiência foi realizada para avaliar o ciclo completo formado pela compreensão de linguagem falada, a gestão de diálogo, a aquisição de conhecimento e resposta a perguntas.

5.1.2.1 Método

Um conjunto de 41 questões (por exemplo “*Is Tony in Canada?*”) foi definido baseado nas frases declarativas utilizadas na experiência anterior.

As respostas dadas pelo robô, que utilizou a base de conhecimento adquirida através da experiência anterior, foram comparadas a uma situação “utópica”. O estado “utópico” da base de conhecimento foi criado utilizando-se a semântica correta das 41 frases declarativas do primeiro teste. A seguir, a semântica correta das 41 questões correspondentes foi submetida ao sistema de gestão de conhecimento. Assim, respostas “utópicas” foram obtidas. Obviamente, o sistema conseguiu responder a todas as questões, mas como a confiança na resposta depende da quantidade de afirmações que a suportam (ver secção 3.3.3, eq. 3.2, na página 72), a confiança média ficou consideravelmente abaixo de 100%.

Na análise dos resultados, é utilizada uma medida de eficiência que compara o desempenho do sistema desenvolvido com o desempenho do sistema em um cenário “utópico”. Esta medida consiste na razão entre os dois valores.

5.1.2.2 Resultados

A tabela 5.4 apresenta os resultados desta comparação entre realidade e utopia, novamente em dois cenários: sem diálogos de confirmação, usando o antigo gestor de diálogo do Carl; e com diálogos de confirmação, usando o atual gestor de diálogo. A coluna desempenho indica o percentual de questões que puderam ser respondidas. A razão do sistema não ser capaz de responder a algumas questões é consequência do fato das respectivas frases declarativas não terem sido reconhecidas corretamente pelo ASR. Podemos verificar que o sistema com o novo gestor de diálogo foi superior ao anterior, 97,6% versus 90,2% ¹. Se tomarmos o resultado da confiança obtido no cenário com confirmação, 37,3%, e o dividirmos pelo respectivo resultado da utopia, 45,5%, temos uma boa medida da eficiência do sistema em sua versão mais recente, 82,1%. Quando comparado ao sistema sem confirmação, o atual sistema apresentou uma melhoria de 14,9 p.p..

Tabela 5.4: Resultados do módulo de gestão e aquisição de conhecimento - utopia *vs.* realidade.

Cenário	Questões	Respostas	Desempenho	Confiança	Eficiência
Utopia	41	41	100,0%	45,5%	
Sem confirmação	41	37	90,2%	29,4%	67,2%
Com confirmação	41	40	97,6%	37,3%	82,1%

5.1.3 Resolução de Pronomes

Outra experiência foi realizada para analisar o fluxo interno de atualizações após um evento NLU ser recebido pelo gestor de diálogo (figura 4.4) e também para testar a sua capacidade de resolução de pronomes.

5.1.3.1 Método

Para esta experiência, 10 pares de frases foram selecionados. Como não era nossa intenção realizar uma avaliação exaustiva, essa quantidade foi suficiente para testar a resolução de pronomes. Em metade dos pares, as primeiras frases continham dois nomes de pessoas ou lugares. Nos restantes pares, a primeira frase possuía um único nome. Nas segundas frases dos pares, havia sempre um pronome referenciando um nome da primeira frase. O exemplo abaixo mostra um par cuja primeira frase possui um único nome:

Paul is a lawyer. He is in Canada.

¹Parte da melhora se deve a correção de um erro no programa que analisou os resultados. Sem esse erro, a performance do sistema anterior foi de 92,7%.

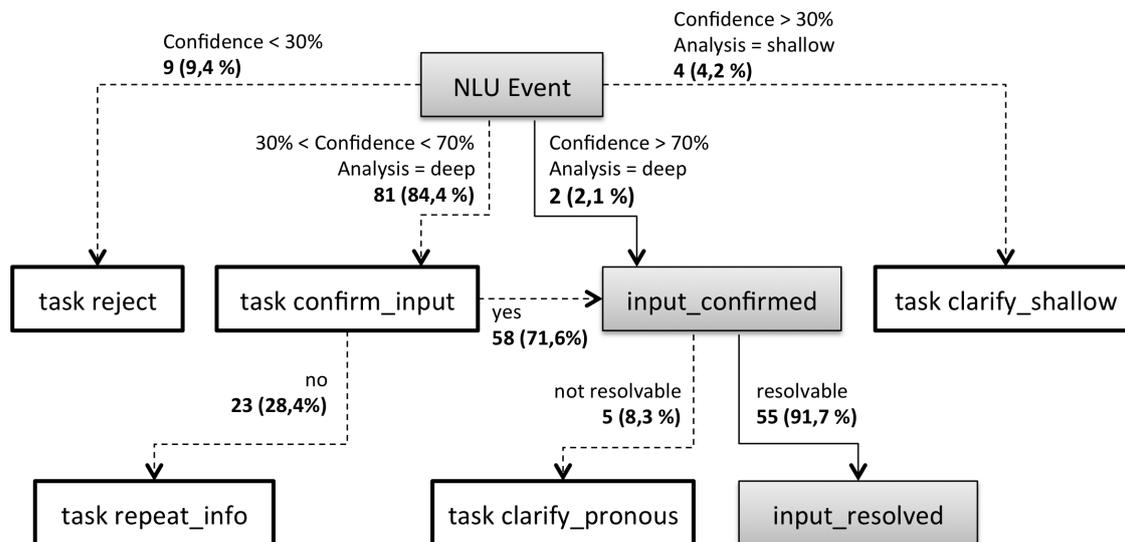


Figura 5.2: Fluxo das atualizações: resultado da experiência

A lista completa das frases utilizadas encontra-se no Apêndice D. As frases foram repetidas até que elas alcançassem por três vezes² o campo *input_confirmed*. O desempenho na resolução de pronomes é aqui avaliado pela percentagem de pronomes corretamente resolvidos.

5.1.3.2 Resultados

A figura 5.2 mostra as etapas entre ocorrência de um evento NLU até o preenchimento do campo *input_resolved* do Estado da Informação (ver seção 4.3, subseção 4.7.3 e figura 4.4). Podemos observar que a tarefa de confirmação de entrada (*confirm_input*) é chamada em 84,4% dos casos. Em 9,4% das vezes, a informação é rejeitada por baixa confiança, e em outros 4,2%, a tarefa *clarify_shallow* é acionada. Em apenas 2,1% dos eventos, a informação vai diretamente para o campo *input_confirmed*, casos nos quais a confiança no reconhecimento foi superior a 70%. Isso revela que foram adotados limites de confiança conservativos para a passagem direta ao campo *input_confirmed*, protegendo a base de conhecimento de frases mal reconhecidas, mas provocando maior uso da tarefa *confirm_input*. Nessa tarefa, a informação foi confirmada em 71,6% das vezes e negada em 28,4%. Do campo *input_confirmed* para o *input_resolved*, 91,7% (55/60) do fluxo passa diretamente e, o restante, através da tarefa *clarify_pronouns*. Nesta experiência, o reconhecimento de fala, no agente o ASR obteve uma taxa média de erro de palavras (*word error rate*) de $11,26\% \pm 27,82\%$.

Como o foco desta experiência era a resolução de pronomes, as frases eram mais simples do que na experiência anterior, permitindo uma extração semântica perfeita para todas as relações presentes nas frases. Na tabela 5.5 encontra-se a análise do sistema composto pelos

²Este número foi escolhido para manter uma consistência com a experiência anterior, na qual foram utilizadas 3 repetições.

módulos ASR, NLU, Dialogue Manager (DM) e Knowledge Acquisition and Management (KAM) na resolução de pronomes. Pode-se ver que ele obteve um excelente grau de eficiência em desempenho e confiança.

Tabela 5.5: Resolução de pronomes.

Dados	Questões	Respostas	Desempenho	Confiança
Utopia	20	20	100,0%	48,9%
Realidade	20	20	100,0%	44,1%
Eficiência			100,0%	90,3%

Apesar do pequeno tamanho do teste, os resultados mostram que o sistema tem potencial para dar suporte à aquisição de conhecimento em um robô móvel inteligente.

5.2 Avaliação Global do Sistema

Esta seção apresenta os resultados de uma avaliação global feita ao sistema de interação baseado em fala, com a participação de vários utilizadores. Essa avaliação foi conduzida na Universidade de Aveiro, no dia 10 de Maio de 2007.

5.2.1 Objetivos

O objetivo decorre de nossa hipótese inicial, de que é possível desenvolver um sistema de diálogo baseado em linguagem natural falada que resolva o problema de encontrar um modo acessível e adaptável de interagir eficientemente com robôs, permitindo a transmissão de conhecimento em ambos sentidos. Assim, pretendemos avaliar nesta seção o desempenho do robô na execução tarefas de interação desencadeadas por diferentes utilizadores. Ao final da avaliação, deveremos responder a perguntas como:

- É possível interagir com o robô e realizar tarefas de interação usando diálogo falado?
- Como é essa interação? É fácil, rápida?
- Há falhas? Onde se concentram?
- Qual a opinião dos utilizadores sobre a interação? Utilizável? Aceitável?

5.2.2 Método

5.2.2.1 Treino do Reconhecedor Automático de Fala

O primeiro desafio aqui foi o treino do modelo de linguagem a ser utilizado pelo agente ASR nesta avaliação. O ideal seria possuir um grande conjunto de frases que tivessem sido

utilizadas por pessoas em interações semelhantes com o robô. Na ausência de tal conjunto, optou-se por usar as frases que haviam sido processadas pelo robô nos meses anteriores e que se encontravam armazenadas em arquivo, um total de 560 frases, e acrescentar as necessárias para esta avaliação. Por exemplo, “I don’t know” não havia sido falado por nenhum utilizador até então, mas passa a ser uma frase válida porque agora o robô também pode fazer perguntas. Além dessas, também foram acrescentadas as frases e os nomes encontrados no site do IEETA³, com objetivo de ampliar o reconhecimento de palavras.

Como ainda assim o conjunto de frases era insuficiente para geração de um modelo de linguagem, optou-se por utilizar regras de classe. Esta técnica tem por objetivo ampliar o conjunto das frases que podem ser reconhecidas pelo sistema e consiste em substituir palavras por suas respectivas classes. Com isso, além da palavra originalmente presente, qualquer outra da mesma classe passa a ser considerada válida para aquela frase.

Foram criadas as seguintes regras de classe para: nome de pessoa (*PersonName*), nome de lugar (*PlaceName*), substantivo comum (*CommonNoun*), função (*Function*), verbo ser/estar (*BeVerb*) e comando (*Command*).

A seguir, essas regras de classe foram utilizadas para gerar um pacote de etiquetagem com a ferramenta **nuance-compile**. Este pacote serviu de parâmetro para o utilitário **nl-tag-tool** etiquetar o conjunto de treino. Concretamente, o *nl-tag-tool* percorreu todo o arquivo de texto inicial trocando cada ocorrência específica por sua classe correspondente, por exemplo, trocar *Portugal* por *PlaceName*.

Com base no conjunto de treino etiquetado, um modelo estatístico foi gerado com a ferramenta **train-slm**. Por fim, uma gramática foi definida e compilada através da ferramenta **nuance-compile**. Todas as ferramentas utilizadas fazem parte do pacote da ferramenta Nuance 8.0 [Nuance, 2001].

5.2.2.2 Protocolo de Avaliação

Dez alunos de graduação e pós-graduação da Universidade de Aveiro, em sua maioria portugueses, participaram na avaliação. Cada um deles recebeu as seguintes instruções:

You have to perform 5 tasks, two of them being navigation tasks.

The tasks are randomly chosen.

Pay attention because the robot can ask you something in the middle of the dialogue, for instance: “what is Lisbon?”. You could answer in many ways, like “a city”, “it is a city”, “Lisbon is a city”, “I don’t know”, “a place”, etc.

As tarefas foram distribuídas por sorteio, de uma lista com 5 tarefas declarativas, 5 tarefas interrogativas e 5 tarefas de navegação. A lista completa encontra-se na tabela 5.6. A tarefa

³Site: www.ieeta.pt, acessado em abril de 2007.

nº 10 (“Ask where is he”) permite avaliar a resolução de pronomes. A atribuição desta tarefa aos participantes implicou também a atribuição da tarefa nº 5 (“Inform that Almeida is the manager of IEETA”) na qual aparece o referente pretendido para o pronome da tarefa nº 10.

O robô foi previamente alimentado com algumas informações, permitindo-lhe responder a uma parte das questões desde o primeiro diálogo e também gerar comentários sobre alguns fatos. Por exemplo, foi adicionada a informação de que Almeida estava no Canadá para permitir um comentário sobre Almeida na realização da tarefa 5, assim como responder a tarefa 10. Também foram adicionadas informações acerca do IEETA, para que a pergunta 7 fosse respondida com mais detalhes. O conhecimento para resposta de outras questões foi sendo adquirido ao longo da avaliação através da execução de tarefas com objetivo de informar algo ao robô. Por exemplo, as tarefas 1 e 2 servem de base para a pergunta 6. É também importante notar que as tarefas de navegação (11 a 15) foram apresentadas aos pares, como nos exemplos seguintes:

- Command the robot to move backward, then stop it when you want.
- Command the robot to move forward, then stop it when you want.
- Command the robot to turn left, then command the robot to turn right.

Tabela 5.6: Tarefas de interação usadas na avaliação

N	Tipo	Tarefa
1	Declarativa	Inform that Mario is a student.
2	Declarativa	Inform that Mario is a professor.
3	Declarativa	Inform that Carl is a robot .
4	Declarativa	Inform that a robot has battery.
5	Declarativa	Inform that Almeida is the manager of IEETA.
6	Interrogativa	Ask if Mario is a doctor.
7	Interrogativa	Ask what is IEETA.
8	Interrogativa	Ask what is Carl (the pronoun “you” can be used).
9	Interrogativa	Ask who is the manager of IEETA.
10	Interrogativa	Ask where is he (Almeida, but use the pronoun).
11	Navegação	Command the robot to move forward.
12	Navegação	Command the robot to move backward.
13	Navegação	Command the robot to turn left.
14	Navegação	Command the robot to turn right.
15	Navegação	Command the robot to stop.

Durante a avaliação, todos os agentes da arquitetura OAA do Carl produziram relatórios detalhados (*log*) de suas atividades. Esses relatórios foram compilados numa folha de cálculo para que o comportamento do sistema como um todo fosse avaliado em detalhes (ver

subseção 5.2.3). Além disso, o agente de Reconhecimento Automático de Fala também gerou ficheiros com todo o áudio captado durante a avaliação.

No final da avaliação, foi entregue a cada participante um questionário com o objetivo de aferir o grau de satisfação com a interação (ver subseção 5.3).

5.2.3 Resultados

Nesta subseção, apresenta-se uma análise do desempenho global do robô baseada nos relatórios detalhados dos agentes da arquitetura OAA. Primeiramente, iremos analisar se foi possível interagir com o robô e realizar as tarefas. A seguir, mostraremos como foi a interação com o robô e veremos se foi rápida e fácil; iremos identificar os tipos de elocuições utilizadas pelos participantes para realizar as tarefas; apresentaremos o desempenho das perguntas realizadas pelo robô; e, por último, veremos as tarefas criadas pelos participantes. Na procura por falhas no sistema, iremos analisar as elocuições que foram rejeitadas pelo gestor de diálogo, o comportamento do agente NLU e as falhas de interação do robô.

5.2.3.1 Realização das Tarefas

É importante começar a análise dos resultados investigando se foi de fato possível realizar as tarefas. A tabela 5.7 mostra que as tarefas foram executadas com sucesso de uma forma geral.

Tabela 5.7: Sucesso global na realização das tarefas

-	Participantes	Tarefas	Tentativas	Tentativas/ Participante	Tentativas/ Tarefa
Total	10	58	66	6,60	1,14
OK		47	52	5,20	1,11
OK(%)		81%	79%	79%	-

De um total de 58 tarefas, 47 foram realizadas com sucesso, o que representa 81% das tarefas. O leitor mais atento deve estar intrigado com o número de 58 tarefas. Se temos 10 participantes e 5 tarefas para cada um deles, deveríamos ter apenas 50 tarefas. Alguns participantes, porém, realizaram mais tarefas do que as inicialmente atribuídas! Oito destas tarefas adicionais faziam parte do grupo proposto, mostrado na tabela 5.6, por isso são contabilizadas aqui. Além dessas 8, outras 14 foram criadas pelos próprios participantes e serão detalhadas no final desta seção.

Se, após ter sucesso, o participante tentou executar novamente a mesma tarefa, esta foi contabilizada como uma nova tentativa. Assim, temos um total de 58 tarefas e 66 tentativas, dessas tentativas, 79% foram realizadas com sucesso.

A tabela 5.8 apresenta informação similar discriminada por participante.

Tabela 5.8: Sucesso na realização das tarefas de cada participante

Participante	Tarefas	Tentativas		
		n°	OK	OK (%)
1	5	5	4	80%
2	5	5	4	80%
3	5	5	1	20%
4	5	11	8	73%
5	6	6	6	100%
6	8	8	7	88%
7	8	10	8	80%
8	6	6	5	83%
9	5	5	4	80%
10	5	5	5	100%
Média	5,8	6,6	5,2	78%
Desvio padrão	1,2	2,2	2,0	21%

Pode-se notar que o participante 3 teve um desempenho abaixo dos demais, com apenas 20% das tentativas executadas com sucesso. De um outro lado, vemos que os participantes 5 e 10 completaram todas as suas tentativas. Também podemos observar que os participantes 6 e 7 tentaram realizar 3 tarefas adicionais que pertenciam ao grupo inicialmente proposto, num total de 8 tarefas. Os participantes 5 e 8, por sua vez, tentaram realizar 1 tarefa adicional, também do conjunto definido para a avaliação. Os demais se mativeram nas 5 tarefas que lhes foram inicialmente atribuídas.

Tomando como unidade básica a tentativa, temos uma média de 6,6 por participante (com desvio padrão de 2,2⁴). Dessas, uma média de 5,2 ($\pm 2,0$) foram realizadas com sucesso.

Se agruparmos as tarefas de acordo com o seu tipo (tabela 5.9), podemos notar que as de navegação possuem o mais alto índice de realização, 90%, seguidas pelas tarefas interrogativas, com 74%. As tarefas declarativas foram as realizadas com menos sucesso, 58% de um total de 12 tentativas.

Tabela 5.9: Sucesso na realização dos diferentes tipos de tarefa

Tipo	Tarefas	Tentativas	OK	OK (%)
Declarativa	11	12	7	58%
Interrogativa	19	23	17	74%
Navegação	28	31	28	90%

A tabela 5.10 apresenta ainda informação similar, mas agora discriminada por tarefa. Pode

⁴Para facilitar a leitura, de agora em diante indicaremos o desvio padrão com o sinal \pm .

ver-se que só uma delas não foi realizada – a de informar que Mário é um estudante. Vale ressaltar que essa tarefa foi a única com apenas uma tentativa de execução. No outro extremo, também podemos observar que 5 tarefas, 1/3 do total, foram concluídas com sucesso em todas as tentativas.

Tabela 5.10: Sucesso na realização das diferentes tarefas

N	Tarefa	Tentativas	OK	OK (%)
1	Inform that Mario is a student	1	0	0%
2	Inform that Mario is a professor	2	2	100%
3	Inform that Carl is a robot	3	2	67%
4	Inform that a robot has battery	2	1	50%
5	Inform that Almeida is the manager of IEETA	4	2	50%
6	Ask if Mario is a doctor	3	3	100%
7	Ask what is IEETA	4	3	75%
8	Ask what is Carl	8	5	63%
9	Ask who is the manager of IEETA	5	4	80%
10	Ask where is he (Almeida, but use the pronoun)	3	2	67%
11	Command the robot to move forward	6	5	83%
12	Command the robot to move backward	5	5	100%
13	Command the robot to turn left	7	7	100%
14	Command the robot to turn right	6	4	67%
15	Command the robot to stop	7	7	100%
	Média	4,1	3,2	72%
	Desvio padrão	2,0	2,0	26%

5.2.3.2 Aspectos Relativos à Interação

Em seguida, são analisados vários aspectos relacionados com a interação entre o robô e os seus interlocutores, nomeadamente, as tomadas de palavra, os tipos de elocuições utilizadas pelos participantes, as perguntas colocadas pelo robô e as tarefas criadas pelos participantes.

Tomadas de palavra — Para facilitar a compreensão dos dados apresentados em seguida, é importante relembrar o conceito de *tomada de palavra*, a transferência da posse da palavra entre participantes num diálogo. Sobre as noções de ‘posse da palavra’ e ‘tomada da palavra’, ver a seção 2.1.1.

Na tabela 5.11 são apresentadas algumas informações sobre as tomadas de palavra necessárias à execução das tarefas. Na interpretação dos dados contidos nesta tabela, deve-se atentar para o fato de que os números representam a soma das *tomadas de palavra* do robô e dos participantes. Portanto, quando vemos que o número médio de *tomadas de palavra* por tentativa é de 12,8, isso significa que, para cada participante, houve 6,4 *tomadas de palavra*,

em média, em cada tentativa de executar uma tarefa. Da mesma forma, na interação com cada um dos 10 participantes na avaliação, registaram-se em média 84,5 *tomadas de palavra*.

Tabela 5.11: Tomada de palavra

-	Tarefas	Tentativas	Tomadas	Tomadas/ Participante	Tomadas/ Tentativa
Total	58	66	845	84,5	12,8
OK	47	52	582	58,2	11,19
OK(%)	81%	79%	69%	69%	87%

A figura 5.3 mostra um resumo das *tomadas de palavra* por participante. Podemos observar que a quantidade de tomadas de palavra para cada participante varia significativamente, com alguns participantes atingindo mais de 140 *tomadas de palavra* em contraste com outros na faixa de 40 *tomadas*. O mesmo pode ser observado para as *tomadas* por tentativa, com valores acima de 20 para alguns e na faixa de 5 para outros.

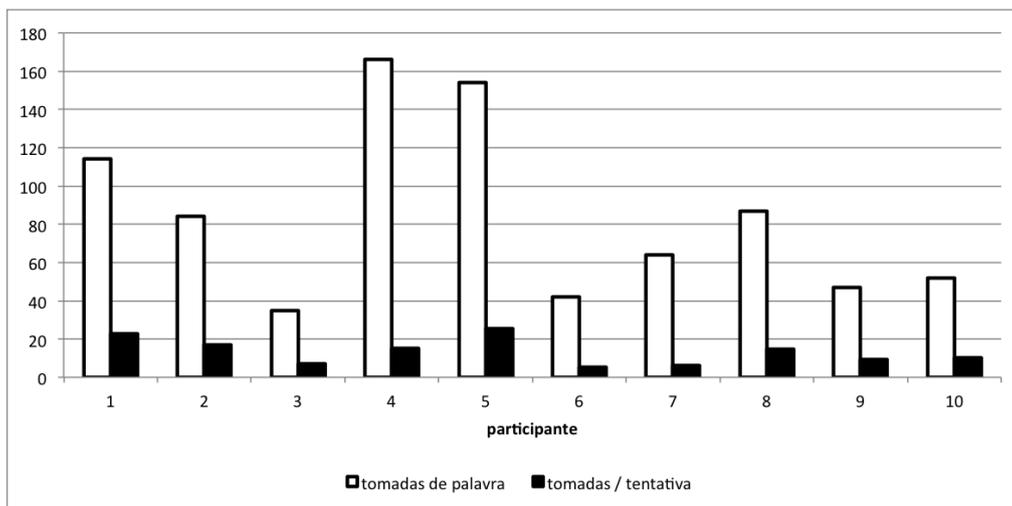


Figura 5.3: Tomadas de palavra por participante

Quando subdividimos esta informação segundo os diferentes tipos de tarefas, verificamos que as tarefas de navegação consomem menos *tomadas de palavra* por tentativa (9,5 no total e 10,1 para as tarefas realizadas com sucesso) do que as tarefas declarativas e interrogativas. Isto pode ser explicado pelo fato das locuções necessárias para a realização das tarefas de navegação serem mais curtas do que as necessárias para a realização dos demais tipos de tarefas. Os dados completos encontram-se na tabela 5.12.

A figura 5.4 mostra a quantidade de *tomadas de palavra* por tarefa. É fácil notar uma grande variação tanto nas *tomadas de palavra* como nas *tomadas de palavra* por tentativa. Por exemplo, enquanto a tarefa 8 registrou mais de 140 *tomadas*, a tarefa 6 marcou apenas 6

Tabela 5.12: Tomadas de palavra por tipo de tarefa

Tipo de Tarefa	Tarefas	Tentativas	Tomadas	Tomadas/ Tentativa	
Declarativa	Total	11	12	214	17,8
	OK	7	7	91	13,0
	OK(%)	64%	58%	43%	-
Interrogativa	Total	19	23	335	14,6
	OK	15	17	209	12,3
	OK(%)	79%	74%	62%	-
Navegação	Total	28	31	296	9,5
	OK	25	28	282	10,1
	OK(%)	89%	90%	95%	-

tomadas de palavra. Variação semelhante pode ser observada em relação às *tomadas de palavra* por tentativa.

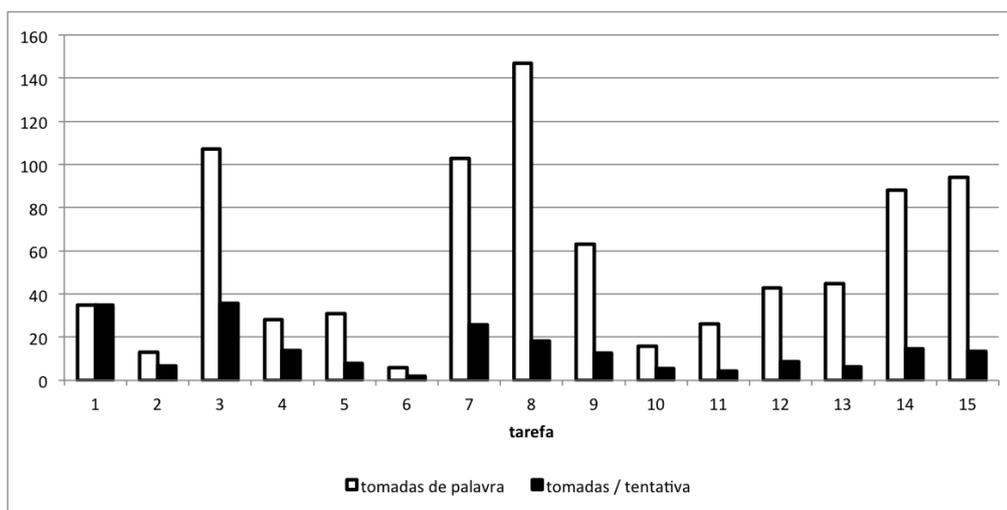


Figura 5.4: Tomadas de palavra por tarefa

Tipo de Elocuções — Aqui, iremos analisar o tipo de elocuições utilizadas pelos participantes (excluindo portanto as frases proferidas pelo robô Carl) na realização das tarefas. Foram identificados quatro tipos:

1. Realização de tarefas – elocuições com o objetivo de realizar uma tarefa.
2. Confirmação – elocuições com o objetivo de responder a um pedido de confirmação de uma informação anteriormente compreendida pelo robô, tipicamente com uma baixa confiança do ASR.

3. Manutenção de diálogo – elocuições sem objetivo relacionado com as tarefas, proferidas apenas para manter o fluxo do diálogo. Por exemplo: *Ok. Carl. Ah...*
4. Resolução de pronomes – elocuições em resposta a um pedido de clarificação relativo ao referente de um pronome dito anteriormente.

Como pode ser observado na tabela 5.13, a maior parte das elocuições tem o objetivo de realizar tarefas, 331, ou 66% do total de 498; uma boa parte delas, 32% (161/498), foi dispendida com a confirmação; e apenas uma quantidade insignificante com manutenção de diálogo e resolução de pronomes.

Tabela 5.13: Tipos de elocuições dos participantes

	Realização de tarefa	Confirmação	Manutenção de diálogo	Resolução de pronomes	Todas as elocuições
Total	331	161	5	1	498
OK	227	102	4	1	334
OK (%)	69%	63%	80%	100%	67%

A figura 5.5 mostra os tipos de elocuições por participante. Podemos ver que os participantes 4 e 5 proferiram mais frases para realização de tarefas. As elocuições de manutenção de diálogo se concentraram no participante 8, que também apresentou, proporcionalmente, um alto número de elocuições de confirmação. Os participantes 6 e 10 utilizaram poucas elocuições de confirmação.

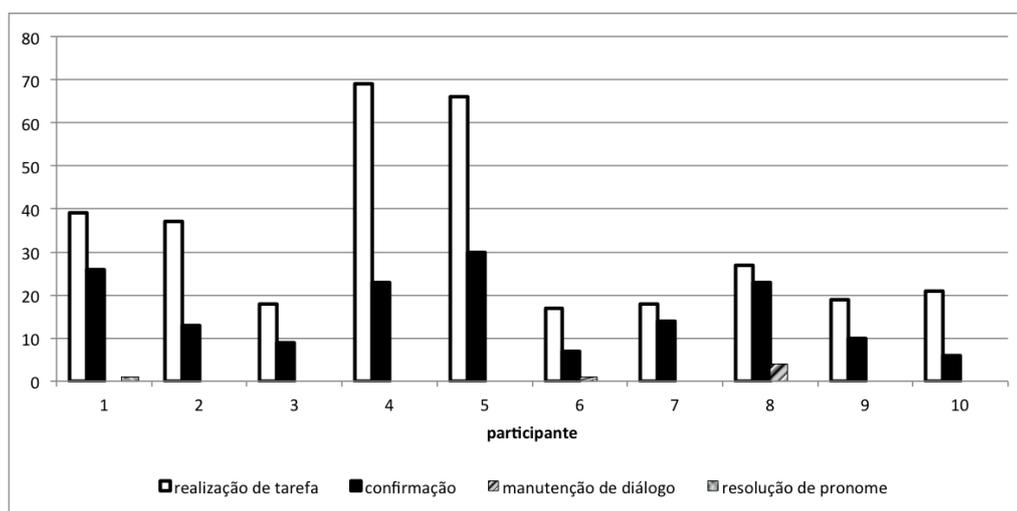


Figura 5.5: Tipo de elocuições utilizadas por participante

Proporcionalmente, houve mais elocuições de confirmação para as tarefas de navegação, 38,9% (69/177), do que para as tarefas declarativas e interrogativas, 29,0% (38/131) e 28,4%

(54/190), respectivamente, como pode ser observado na tabela 5.14.

Tabela 5.14: Tipo de elocuções utilizadas por tipo de tarefa

Tipo de Tarefa		Realização de tarefa	Confirmação	Manutenção de diálogo	Resolução de pronomes	Todas as elocuções
Declarativa	Total	92	38	1	0	131
	OK	43	10	0	0	53
	OK(%)	47%	26%	0%		40%
Interrogativa	Total	135	54	0	1	190
	OK	91	26	0	1	118
	OK(%)	67%	48%	-	100%	62%
Navegação	Total	104	69	4	0	177
	OK	93	66	4	0	163
	OK(%)	89%	96%	100%		92%

Analisando por tarefa, na figura 5.6 vemos que a sexta (*ask if Mario is a doctor*) não precisou de confirmação para ser realizada. Entretanto, na tarefa 12 (*command the robot to move backward*), houve mais elocuções de confirmação do que de execução de tarefa.

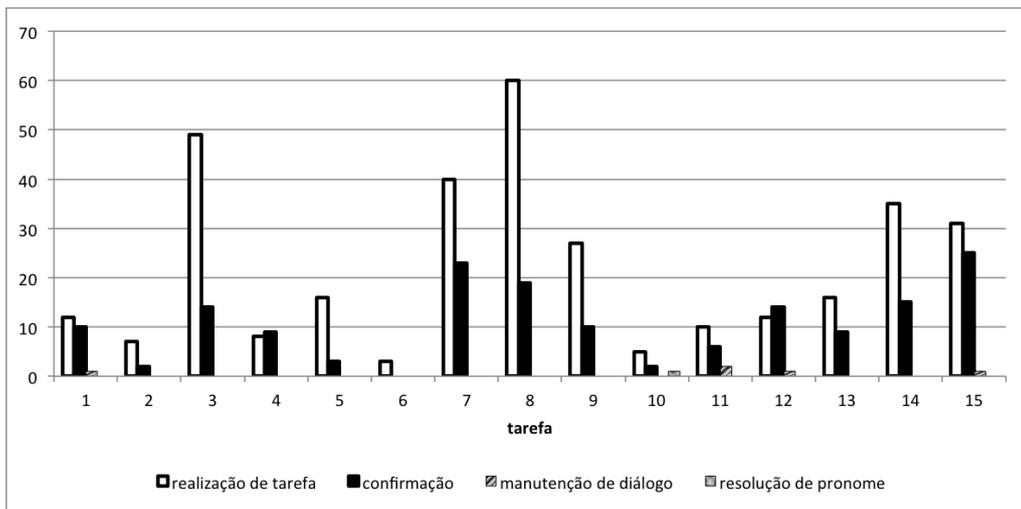


Figura 5.6: Tipo de elocuções utilizadas por tarefa

Perguntas Realizadas pelo Robô — Apesar do aviso incluído nas instruções inicialmente entregues aos participantes na avaliação, segundo o qual o robô poderia realizar perguntas ao longo do diálogo, metade dessas perguntas foi simplesmente ignorada pelos participantes (ver tabela 5.15).

Em 30% das perguntas colocadas pelo robô, o participante não forneceu o tipo de resposta

Tabela 5.15: Resposta a perguntas realizadas pelo robô

Reação do utilizador a perguntas do robô	Quantidade	(%)
Pergunta ignorada	5	50%
Resposta inadequada	3	30%
Resposta adequada, mas descartada pelo robô	1	10%
Resposta adequada, mas mal reconhecida pelo robô	1	10%
Total	10	-

que o robô esperava. Nesses casos, o robô perguntou sobre o tipo de um objeto encontrado na base e os participantes responderam a profissão desse objeto.

O mau desempenho do ASR prejudicou uma resposta “I don’t know”, que apesar de corretamente reconhecida, foi descartada por estar abaixo nível de confiança mínimo estipulado.

Na única resposta que foi aceite pelo robô, houve uma troca de palavras no módulo ASR que acabou sendo confirmada pelo participante. A tabela 5.16 mostra o trecho do diálogo correspondente.

Tabela 5.16: Pergunta respondida pelo participante

Carl:	What is he?
Participante:	A doctor.
Carl:	Did you say Alexander?
Participante:	Mario is a doctor.
Carl:	Please say yes or no.
Participante:	Yes.
Carl:	All right!

Tarefas Criadas pelos Participantes — Um dado muito importante dessa avaliação é o de que 80% dos participantes tentaram realizar alguma tarefa adicional (tabela 5.17). O participante número 8 chegou a criar 5 novas tarefas, enquanto o participante número 7, criou outras 3 tarefas. Isto demonstra o interesse despertado pela interação com o robô. Foram criadas um total de 14 tarefas, das quais 7 foram realizadas com sucesso.

A tabela 5.18 mostra algumas tarefas criadas pelos participantes. Podemos ver que existem desde tarefas simples como cumprimentos e agradecimentos a tarefas interrogativas e declarativas. Na primeira delas, houve falha de reconhecimento e o robô simplesmente deu continuidade ao diálogo. Na segunda, há compreensão, mas o robô não possui a informação pedida. A seguir, observe que o robô responde corretamente utilizando o pronome *he*. Na última tarefa apresentada, há um diálogo de confirmação.

Tabela 5.17: Resumo das tarefas criadas pelos participantes

Participante	Tarefas Criadas	OK	OK(%)
1	0	0	-
2	0	0	-
3	1	0	0%
4	1	0	0%
5	1	0	0%
6	1	1	100%
7	3	2	67%
8	5	3	60%
9	1	0	0%
10	1	1	100%
Total	14	7	41%
Média	1,40	0,70	41%
Desvio padrão	1,43	1,00	43%

Tabela 5.18: Algumas tarefas criadas pelos participantes

Participante 4:	Mario is a scientist.
Carl:	I'm sorry?
Participante 6:	Who is paul?
Carl:	I'm afraid I cannot help you with that.
Participante 7:	Who is almeida?
Carl:	He is the manager of Ieeta.
Participante 8:	Hi!
Carl:	Did you say Hello!?
Participante 8:	Yes.
Carl:	Hi there!

5.2.3.3 Análise de Falhas

Em seguida, iremos analisar as falhas do sistema, nomeadamente, as elocuições rejeitadas pelo sistema, o reconhecimento de atos de fala e as falhas de interação de uma maneira geral.

Rejeição de elocuições — Analisamos aqui um pouco do comportamento do gestor de diálogo, mostrando das frases efetivamente ditas pelos participantes, quantas foram rejeitadas e qual o motivo.

A tabela 5.19 mostra o número de elocuições rejeitadas para todas as tarefas e para as tarefas realizadas (OK). Podemos observar que o principal motivo de rejeição, ocorrido em 53 elocuições, foi a impossibilidade de extração de informação pelo agente NLU, ou seja, não foi

possível extrair uma semântica a partir da saída do agente NLA. Em 41 elocuições, o agente NLA não foi capaz de gerar uma análise sintática com o que recebeu do agente ASR. E em 4 elocuições a informação semântica extraída foi rejeitada pela baixa confiança do ASR. Para as tarefas realizadas (OK), observe que há, proporcionalmente, um aumento nas rejeições ocorridas em função do agente NLA, mas que o agente NLU continua a ter mais ocorrências.

Tabela 5.19: Elocuições dos participantes rejeitadas pelo gestor de diálogo

	Tentativas	Elocuições	Rejeitadas	Motivo da rejeição		
				Confiança ASR	NLA	NLU
Total	66	498	98 (19,7%)	4 (0,8%)	41(8,2%)	53 (10,6%)
OK	52	334	68 (20,4%)	2 (0,6%)	32 (9,6%)	34 (10,2%)

Reconhecimento de Atos de Fala — Iremos analisar em mais detalhes o reconhecimento dos atos de fala pelo agente NLU. A tabela 5.20 mostra o desempenho do NLU na identificação dos atos de fala. Pode-se notar que, em alguns casos, há mais erros do que acertos. Este é um dado importante porque a gestão do diálogo é realizada em função do ato de fala reconhecido. Uma falha aqui compromete todo o fluxo subsequente do diálogo.

Tabela 5.20: Desempenho do robô na identificação dos atos de fala

	Tentativas	Atos de fala dos participantes	Atos de fala corretos	Atos de fala incorretos	Precisão
OK	52	334	162	172	48,5%

A matriz de confusão de atos de fala no agente NLU, na tabela 5.21, mostra que ato de fala foi reconhecido para cada ato de fala proferido pelos participantes. Na primeira coluna temos que o ato de fala `tell` foi reconhecido corretamente 37 vezes num total de 57 e que em 15 vezes foi entendido como `ask_if`, 2 vezes como `achieve`, 1 vez como `describe` e 2 como `reply`.

Ainda nessa tabela, podemos ver que os atos de fala `reply`, `confirm` e `deny` tiveram uma boa taxa de acerto, cerca de 85%, ao passo que os atos de fala `ask` e `describe` ficaram com somente 43% e 31%, respectivamente. Os atos de fala `tell` e `achieve` tiveram um desempenho razoável de 65% e 56%, respectivamente. No geral, o reconhecimento correto dos atos de fala foi 66%.

Tabela 5.21: Matriz de confusão de atos de fala no agente NLU. Para facilitar a leitura, os acertos estão em negrito e a principal troca, sublinhada.

Ato de fala compreendido	Ato de fala proferido									
	tell	ask	ask_if	achieve	describe	greet	reply	confirm	deny	sorry
tell	37	2	0	6	<u>19</u>	0	0	2	0	0
ask	0	10	0	0	12	0	0	0	0	0
ask_if	<u>15</u>	1	3	0	1	0	0	0	0	0
achieve	2	0	0	42	5	0	<u>3</u>	0	1	0
describe	1	0	0	1	19	0	0	0	0	0
greet	0	0	0	2	1	1	0	1	1	0
reply	2	<u>10</u>	<u>1</u>	<u>24</u>	4	<u>1</u>	16	<u>5</u>	<u>13</u>	0
confirm	0	0	0	0	0	0	0	50	0	0
deny	0	0	0	0	0	0	0	0	89	0
sorry	0	0	0	0	0	0	0	0	0	1
Total	57	23	4	75	61	2	19	58	104	1
Acerto (%)	65%	43%	75%	56%	31%	50%	84%	86%	86%	100%

Falta de Resposta do Robô — Num total de 578 interações, 96 falharam. Aqui, o termo ‘interação’ deve ser entendido como uma locução de um participante que necessitava de resposta do robô. Essas interações não tem uma relação fixa com as tomadas de palavra. Na maior parte dos casos, cada interação corresponde a 2 tomadas de palavra, mas isso pode ser diferente. Considere a situação hipotética na qual um participante fala 3 vezes e o robô não responde a nenhuma das vezes. Nesse caso, seriam contabilizados 1 tomada de palavra (apenas o participante falou nesse exemplo) e 3 interações (o participante tentou por 3 vezes perguntar ou dizer algo ao robô).

Tabela 5.22: Falhas de interação do robô

Agente	Interações	Falhas	Falhas (%)
Reconhecimento automático de fala (ASR)		44	8%
Análise sintática de linguagem natural (NLA)		10	2%
Compreensão de linguagem natural (NLU)		1	0%
Gestor (dialogue manager)		41	7%
Geração de linguagem natural (NLG)		0	0%
Total	578	96	17%

A tabela 5.22 identifica a responsabilidade na falta de resposta de cada um dos agentes da arquitetura OAA do Carl. As falhas do ASR estão concentradas em retornos do tipo ABORT ou SPEECH_TOO_EARLY, estes últimos talvez por falta de cuidado do participante em

falar ao mesmo tempo do robô. No caso do agente DM, uma falha comum foi a recebimento de um ato de fala do tipo **reply** sem que o robô tivesse feito pergunta ao participante. Nestes casos, o DM não gerou retorno. Da mesma forma, confirmações/negações (Yes/No) sem uma pergunta prévia do robô, ficaram sem resposta. As dez falhas do NLA ocorreram no final de uma sessão, na qual esse agente bloqueou após uma chamada ao componente LCFlex.

5.2.3.4 Aquisição de Conhecimento

A lista de tarefas foi elaborada com a intenção de provocar situações que evidenciassem a aquisição de conhecimento pelo robô. Como mostrado anteriormente na tabela 5.6, algumas tarefas questionavam informações que eram fornecidas por frases declarativas da própria avaliação.

A tabela 5.23 mostra trechos do diálogo com três participantes que evidenciam a aquisição de conhecimento pelo robô. No primeiro trecho, o robô é questionado sobre a profissão de Mário e, não tendo referência em sua base, responde que não pode ajudar o participante. No seguinte, o robô é informado de que Mário seria um professor. Por último, quando questionado novamente se Mário seria um médico, o robô nega e complementa a resposta indicando que ele é um professor.

Tabela 5.23: Evidências de aquisição de conhecimento

Participante 4:	Is Mario a doctor?
Carl:	I'm afraid I cannot help you with that.
Participante 5:	Mario is a professor.
Carl:	Did you say that Mario is a professor?
Participante 5:	Yes.
Participante 6:	Is Mario a doctor?
Carl:	No.
Carl:	He is a professor.

5.3 Avaliação de Usabilidade

Como já referido, no final da avaliação global descrita na seção anterior, foi entregue a cada participante um questionário com o objetivo de aferir o grau de satisfação com a interação (ver questionário na tabela 5.24). Essas respostas, dadas numa escala de 1 (pior desempenho) até 5 (melhor desempenho), serão analisadas em seguida. A figura 5.7 mostra um resumo dessa avaliação.

Pode-se observar que os participantes consideraram fácil compreender o robô (figura 5.7(a)), mas tiveram alguma dificuldade em se fazerem entender (figura 5.7(b)).

Tabela 5.24: Questionário sobre satisfação global

Perguntas	Tradução
Was the robot response easy to understand?	Foi fácil compreender o robô?
Did the robot understand what you said?	O robô compreendeu o que você falou?
Was it easy to find the answer/provide information you wanted?	Foi fácil encontrar a resposta ou fornecer a informação que você queria?
Was the pace of interaction with the robot appropriate?	O ritmo da interação com o robô foi adequado?
Did you know what you could say at each point of the dialogue?	Você sabia o que poderia dizer em cada ponto do diálogo?
Did the robot work the way you expected it to?	O robô se comportou da maneira que você esperava?

A figura 5.7(c) indica que o ritmo da interação com o robô foi apropriado. Enquanto alguns participantes consideraram fácil a troca de informações com o robô, outros tiveram mais dificuldades nesse aspecto, conforme figura 5.7(d).

De acordo com a figura 5.7(e), os participantes sabiam o que poderiam dizer a cada ponto do diálogo. A figura 5.7(f) mostra que o robô funcionou como esperado para a maior parte dos participantes.

5.4 Discussão

Diante dos resultados apresentados, podemos afirmar que foi possível interagir com o robô, realizar as tarefas propostas e outras criadas espontaneamente pelos participantes da avaliação, mostrando, assim, que o sistema de diálogo baseado em linguagem natural falada desenvolvido foi capaz de gerenciar a interação de modo eficiente e permitiu a transmissão de conhecimento em ambos sentidos.

Algumas pessoas tiveram mais dificuldades na comunicação com o robô e acabaram por completar menos tarefas, outras, sentiram-se tão confortáveis na interação que chegaram a realizar todas as tarefas. De uma maneira geral, 4 em cada 5 tarefas foram realizadas com sucesso. Na tentativa de executar uma tarefa, os participantes precisaram de 6,4 *tomadas de palavra*, em média, o que corresponde a um diálogo com 12,8 *tomadas de palavra* no total. Um dado que explica essa demora é o elevado número de rejeições das confirmações pedidas pelo robô. Em 45,3% das vezes, a locução foi rejeitada pelos participante. Ou seja, em quase metade dos casos, o robô compreendeu algo que os participantes não disseram. A causa principal disso foi o mau reconhecimento de fala pelo agente ASR. Também devemos destacar a dificuldade em detectar corretamente o ato de fala proferido. Para alguns atos de fala, o agente NLU chegou a errar mais do que acertar. No geral, a precisão foi de 66%, que se deve

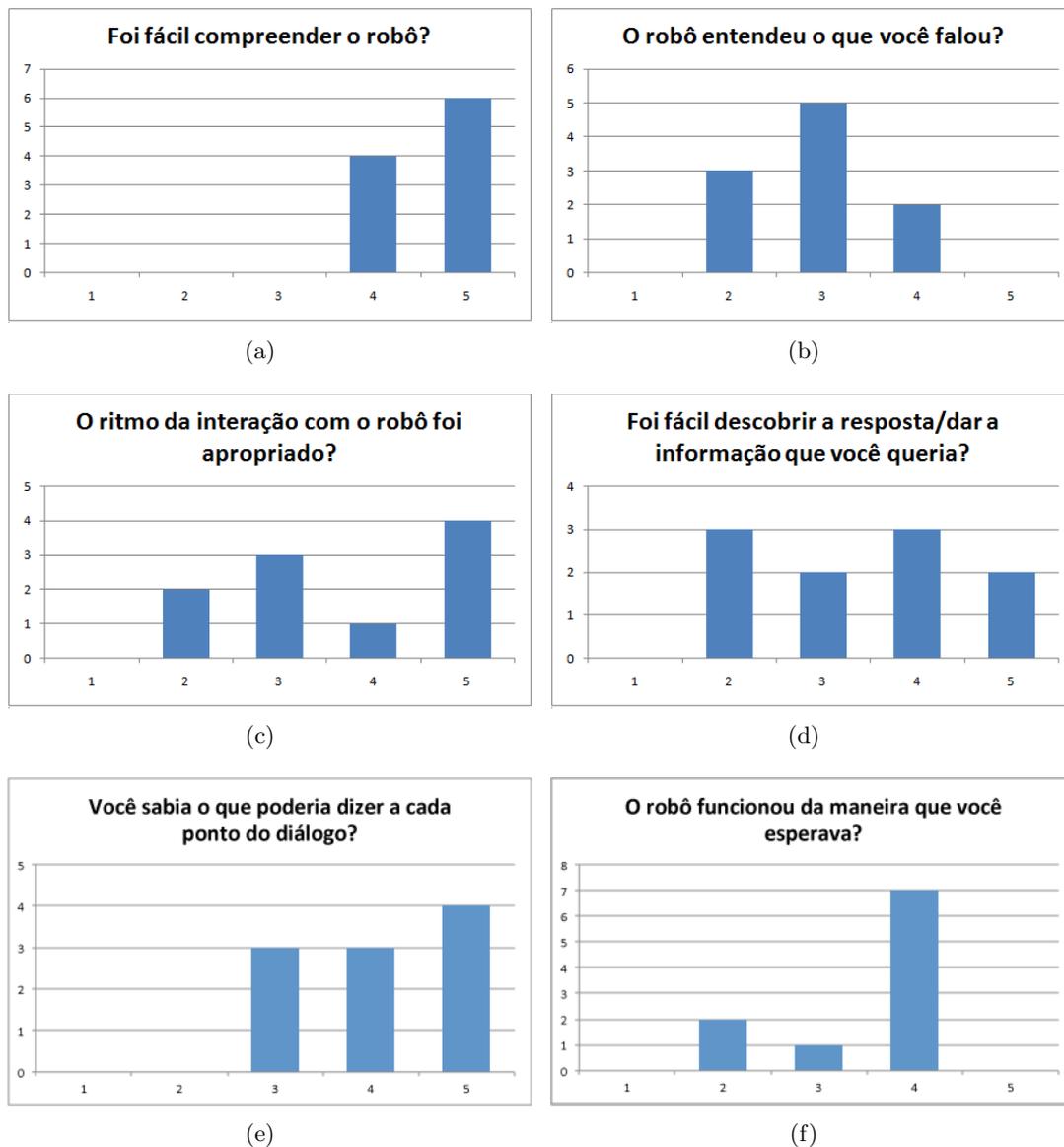


Figura 5.7: Avaliação dos participantes

considerar baixa. Metade das perguntas realizadas pelo robô foi ignorada pelos participantes. Na ânsia de realizar as suas tarefas, eles não perceberam que o robô havia perguntado algo. Uma alternativa para minimizar isso no futuro, seria uma mudança na abordagem do robô. Ao invés de perguntar diretamente, poderia haver um pedido prévio, algo como “*May I ask you something?*”. Houve falhas em vários níveis. Situações não previstas nas regras do gestor de diálogo deixaram o robô sem resposta algumas vezes, por exemplo, quando o participante confirmava ou negava algo sem que uma pergunta anterior tivesse sido feita. Outros agentes também tiveram pequenas falhas, no entanto, o que mais impactou o andamento do diálogo foram os problemas de reconhecimento do ASR.

Entretanto, mesmo com os problemas relatados acima, os participantes demonstraram interesse pela interação. Uma prova disso é que 80% deles tentaram realizar tarefas fora do que havia sido pedido. De maneira geral, os participantes consideraram fácil compreender o robô, tiveram certa dificuldade em se fazerem entender, mas consideraram que o robô se comportou conforme o esperado.

Ao se comparar o desempenho de outros robôs em tarefas que envolvem diálogo, perceberemos números semelhantes aos obtidos pelo Carl. A implementação de um árbitro para o jogo “rock-paper-scissors”, com o robô Honda ASIMO como plataforma, teve um percentual de 70% das tarefas realizadas [Nakadai *et al.*, 2008], um pouco inferior aos 79% do Carl, relatados aqui. Embora, nessa implementação, o ASIMO tenha utilizado reconhecimento de fala simultâneo, pelas características do jogo em questão, a gestão de diálogo foi baseada em uma máquina de estados finitos. Ou seja, se por um lado o reconhecimento de fala foi mais complexo do que o do Carl, a gestão de diálogo foi mais simples. A avaliação do robô PHOPE [Oh *et al.*, 2009], projetado para servir de guia em ambientes fechados, foi realizada com um utilizador simulado e os resultados foram apresentados em função da taxa de erro de reconhecimento de fala (WER): para reconhecimentos perfeitos, 90% das tarefas foram completadas; e, para uma taxa de erro de 30%, o percentual de tarefas concluídas com sucesso foi de 79%. Esse robô utiliza um módulo de compreensão de linguagem natural diferente do Carl, no qual conceitos são extraídos e armazenados em formulários; também diferente é sua gestão de diálogo, baseada em exemplos; e, por último, ao contrário do Carl, o robô PHOPE não possui capacidade de aprendizagem.

Capítulo 6

Conclusões

Este capítulo apresenta um resumo do trabalho realizado pelo autor, destacando os principais resultados alcançados. Por último, algumas sugestões para continuação do trabalho são apontadas, concluindo a tese.

6.1 Resumo do Trabalho Realizado

6.1.1 Arquitetura Computacional

Uma das principais tarefas, fundamental para o trabalho posterior, consistiu na definição e implementação de uma nova arquitetura computacional no robô Carl, com base na Open Agent Architecture (OAA) [Martin *et al.*, 1999], uma infraestrutura, disponível no domínio público, que pode ser utilizada para a integração de uma comunidade de agentes heterogêneos. Com a adoção da OAA, a arquitetura do Carl ficou organizada em torno dos seguintes agentes: Interface Gráfica e Tátil, Reconhecimento Automático de Fala, Análise Sintática de Linguagem Natural, Compreensão de Linguagem Natural, Gestão Central, Geração de Linguagem Natural, Síntese de Fala, e Navegação. A execução destes agentes está distribuída pelos dois computadores existentes no Carl.

Embora trabalhosa e sem resultados imediatos em termos de novas funcionalidades para o robô, essa mudança de arquitetura foi essencial para os novos desenvolvimentos, pois a troca de componentes passou a ser viável com a sua adoção. Tudo o que é preciso para substituir um agente, é garantir que o novo forneça os mesmos serviços do agente anterior.

A principal dificuldade encontrada foi a extensão da mudança: todos os componentes do sistema tiveram de ser alterados para utilizar a nova arquitetura.

6.1.2 Representação do Conhecimento e Raciocínio

Sendo a aquisição de conhecimentos pelo robô central neste trabalho, foi desenvolvido um módulo de representação de conhecimento e raciocínio, o qual ficou integrado no agente de

Gestão Central da arquitetura do Carl. Para isso, foi definida uma linguagem de representação de conhecimento e foram empregados mecanismos de inferência [Seabra Lopes *et al.*, 2005a].

A linguagem de representação criada baseia-se em conceitos importados das redes semânticas e dos diagramas de classes e objetos da *Unified Modelling Language* (UML). Seus principais predicados declaram tipos, objetos, nomes de objetos, atributos de objetos e diversas relações entre tipos e objetos.

Os dois procedimentos principais deste módulo são o *tell* e o *ask*. No primeiro, a informação de um interlocutor é armazenada na base. No segundo, os mecanismos de inferência por dedução, indução e analogia são utilizados para fornecer a melhor resposta possível a partir das informações contidas na base de conhecimento em cada momento, informações estas que podem ser contraditórias.

Esse é um componente chave de todo o sistema porque é responsável por armazenar a informação colhida ao longo dos diálogos e de utilizá-la para dar suporte a outros diálogos.

6.1.3 Geração de Linguagem Natural

O agente de geração de linguagem natural (NLG, de Natural Language Generation) foi construído com base no formalismo MUG, que utiliza a estrutura de dados *functional description* (FD) e unificações como pilares de seu funcionamento.

A principal mudança, em relação ao módulo utilizado pelo robô Carl anteriormente, é a geração baseada em semântica e não mais em padrões (*templates*). Com isso, tornou-se possível utilizar a mesma linguagem de representação de conhecimento desenvolvida para o módulo KAM.

Após a criação da primeira versão desse agente, houve uma reestruturação para torná-lo mais flexível e foi acrescentado o suporte a vários tipos de atos de fala que estavam fora do escopo inicial.

6.1.4 Gestor de Diálogo

Foi concebido um novo gestor de diálogo que abordou todos os requisitos definidos para o robô Carl, nomeadamente: contexto do diálogo, resolução de referências, resolução de elipses, diálogos para confirmação de frases e resolução de pronomes, estabelecimento de conhecimento comum e iniciativa mista.

Este sistema pode ser considerado o principal, pois todos os outros dão suporte a ele. Ou, visto de uma outra forma, ele integra os serviços de todos os outros agentes, gerindo o comportamento global do Carl. Por essas características, seu desenvolvimento consumiu a parcela mais significativa do tempo dedicado a este trabalho.

Após análise de várias abordagens de gestão de diálogo, foi decidido adotar a de *Estado da Informação* (IS, do inglês *Information State*), por atender a todos os requisitos levantados e ser de implementação simples e objetiva. Essa abordagem baseia-se em uma estrutura de

dados que concentra as informações necessárias para a gestão do diálogo, o chamado *Estado da Informação*. Ao seu redor existem regras de atualização, que levam em consideração a recepção de eventos e o próprio *IS*; despacho de ações a executar, tendo em conta a lista de tarefas, um campo do *IS*; e algoritmos de controle, que efetivamente aplicam as regras de atualização e realizam o despacho de ação.

6.1.5 Avaliação de Funcionalidades

Após o desenvolvimento dos componentes supracitados, foram realizadas avaliações das funcionalidades essenciais do robô. No primeiro conjunto de experiências, o sistema de compreensão de linguagem falada (SLU, de Spoken Language Understanding) e a capacidade de clarificação semântica do gestor de diálogo foram avaliados na compreensão de frases declarativas [Quinderé *et al.*, 2007b]. Essa avaliação mostrou que o uso de diálogos de confirmação, utilizando o gestor de diálogo descrito no capítulo 4, permitiu uma melhora de 6 pontos percentuais (p.p.) na *Cobertura* e 7 p.p. na *Precisão* das relações semânticas extraídas, quando comparado ao desempenho sem diálogos de confirmação, obtido com o gestor de diálogo utilizado anteriormente [Seabra Lopes, 2002].

A segunda experiência foi realizada para avaliar o ciclo completo formado pela compreensão de linguagem falada, a gestão de diálogo, a aquisição de conhecimento e resposta a perguntas. Aqui, o cenário com diálogos de confirmação, utilizando o atual gestor de diálogo, obteve desempenho e eficiência superiores ao anterior, com melhorias de 7 p.p e 14 p.p, respectivamente. A terceira experiência foi realizada com o objetivo principal de testar a capacidade de resolução de pronomes do atual gestor de diálogo. Em 91% dos casos, a resolução de pronomes foi realizada automaticamente com base no contexto do diálogo. Nos restantes casos, foi preciso clarificar o pronome com o utilizador. A análise mostrou que o sistema foi capaz de responder a todas as questões, mostrando que a resolução de pronomes foi realizada com sucesso.

6.1.6 Avaliação Global do Sistema

Por último, foi realizada uma avaliação do sistema completo, que envolveu uma série de preparativos para sua execução, como por exemplo: a definição do modelo de linguagem a ser utilizado pelo ASR, a determinação das tarefas a serem executadas e o recrutamento de voluntários para participar nos testes.

As tarefas, que foram distribuídas aos participantes através de sorteio, envolviam frases declarativas, perguntas e comandos de navegação. Nessa avaliação, todos os agentes da arquitetura do Carl geraram relatórios detalhados (*logs*) de suas atividades, tornando árduo o processamento dos dados gerados.

A média de *tomadas de palavra* por participante na tentativa de executar uma tarefa foi de 6,4, o que corresponde a um diálogo com 12,8 *tomadas de palavra* no total. Uma

explicação para esse número talvez seja a alta taxa de rejeição das confirmações pedidas pelo robô, 45,3%, que por sua vez tem origem no mau reconhecimento de fala pelo agente ASR. Também associado a isso, destacamos a dificuldade em detectar corretamente o ato de fala. A precisão do agente NLU foi de apenas 66%.

Apesar dos problemas, a análise dos resultados mostrou que 81% das tarefas foram realizadas com sucesso, com o grupo das tarefas de navegação apresentando um percentual de realização mais elevado, 90%.

Um dado interessante dessa avaliação é o de que 80% dos participantes tentaram realizar alguma tarefa extra. Um participante chegou a criar 5 novas tarefas! Isto demonstra o interesse despertado pela interação com o robô.

Ao final, os participantes foram submetidos a um questionário para avaliar a interação com o robô, com respostas dadas numa escala de 1 (pior desempenho) até 5 (melhor desempenho). De uma maneira geral, eles consideraram fácil compreender o robô – com 100% de avaliações positivas (pontuações de 4 e 5), mas tiveram alguma dificuldade em se fazerem entender – 30% de avaliações negativas (pontuações de 1 e 2). A troca de informações com o robô foi considerada difícil por uns e fácil por outros participantes. O ritmo da interação com o robô foi apropriado, eles sabiam o que poderiam dizer a cada momento e acharam que o robô funcionou como esperado, com 70% de avaliações positivas.

6.2 Principais Resultados

Dentre os resultados atingidos, destacamos a construção de um sistema completo de interação humano-robô através de linguagem falada. Para tanto, uma nova arquitetura computacional teve de ser adotada. Essa arquitetura flexível facilitou o desenvolvimento dos demais componentes necessários ao sistema, como um gestor de diálogo de iniciativa mista, capaz de clarificar frases e pronomes, manter um contexto, resolver elipses e referências.

Além disso, ressaltamos o módulo de gestão de conhecimento e raciocínio, que permite ao robô acumular conhecimento, possivelmente contraditório, e utilizar esse conhecimento como base para diálogos futuros. Durante a avaliação, a utilidade deste módulo pôde ser comprovada. Uma mesma pergunta foi respondida de duas formas pelo Carl. Primeiramente, o robô lamentou não poder ajudar o participante. Após um diálogo intermediário, o Carl foi capaz de responder à questão colocada anteriormente, evidenciando a aquisição de conhecimento.

O resultado da avaliação de desempenho do sistema mostra que pessoas sem treinamento prévio de operação do robô foram capazes de interagir com ele e realizar tarefas através de linguagem natural falada, no caso, a língua inglesa. Dessa forma, entendemos que o trabalho desenvolvido conseguiu atingir seus objetivos iniciais, que eram definir, implementar e avaliar um sistema de diálogo utilizável na interação baseada em linguagem natural falada entre humanos e agentes inteligentes.

6.3 Sugestões para Trabalho Futuro

Existem vários pontos que podem ser melhorados no sistema do robô Carl. A compreensão de linguagem falada precisa de modelos de linguagem mais apropriados, que poderão ser construídos com a criação de base de dados de frases utilizadas na interação com o robô.

A geração de linguagem natural poder ser evoluída em vários aspectos, como na utilização do grau de confiança do módulo de gestão de conhecimento e na maior variabilidade de saída.

Os resultados da avaliação realizada indicam vários pontos que podem ser melhorados no sistema. As regras do gestor de diálogo podem ser estendidas para cobrir situações inesperadas, como por exemplo, o caso do participante responder a algo que não foi perguntado.

Com o objetivo de garantir a atenção do utilizador, o gestor de diálogo pode mudar a abordagem de perguntar diretamente para uma abordagem de pedir permissão antes de perguntar, algo como *“May I ask you something?”*.

A melhora no desempenho do agente de compreensão de linguagem natural está diretamente relacionada a uma melhora no desempenho do agente de reconhecimento automático de fala, visto que grande parte dos erros na identificação do ato de fala foram causados por problemas no reconhecimento. Este é um ponto crucial pois um erro no reconhecimento de fala causa problemas nos demais agentes e impede uma interação fluida.

Novas habilidades podem ser integradas ao sistema, como a capacidade de reconhecer gestos e referências a lugares, por exemplo *“here”* ou *“there”*.

Uma outra melhoria que poderá ser acrescentada ao gestor de diálogo é a capacidade de resolução de correferências. Isto é, a tarefa de descobrir no diálogo expressões que fazem referência a uma mesma entidade. Por exemplo, em “Eu comprei uma maçã e uma laranja. A laranja fica para ti.” A segunda “laranja” faz referência à mesma entidade da primeira frase.

Por último, mas também importante, seria a possibilidade do Carl falar e entender português.

Bibliografia

- Abreu (2006)** Filipe Abreu. Agente de Reconhecimento de Fala para um Robô Móvel Inteligente. Relatório técnico, Universidade de Aveiro. Citado na pág. 58
- ActivMedia (2002)** ActivMedia. *ARIA Reference Manual 1.1.7*. ActivMedia Robotics, September 2002. Citado na pág. 52
- Akker et al. (2005)** Rieks op den Akker, Harry Bunt, Simon Keizer e Boris van Schooten. From Question Answering to Spoken Dialogue: Towards an Information Search Assistant for Interactive Multimodal Information Extraction. Em *Interspeech 2005*, páginas 2793–2796. implementação de um SDS a partir de um sistema de question answering. Citado na pág. 34
- Allen et al. (2000)** James Allen, Donna Byron, Myroslava Dzikovska, George Ferguson, Lucian Galescu e Amanda Stent. An architecture for a generic dialogue shell. *Natural Language Engineering*, 6(3-4):213–228. Citado na pág. vii, 26, 27
- Allen (1994)** James F. Allen. The TRAINS project: A case study in building a conversational planning agent. *Journal of Experimental and Theoretical AI (JETAI)*, 7:7–48. Citado na pág. vii, 28, 29
- Asoh et al. (2001)** Hideki Asoh, Nikos A. Vlassis, Yoichi Motomura, Futoshi Asano, Isao Hara, Satoru Hayamizu, Katsunobu Itou, Takio Kurita, Toshihiro Matsui, Roland Bunschoten e Ben J. A. Kröse. Jijo-2: an office robot that communicates and learns. *IEEE Intelligent Systems*, 16(5):46–55. Citado na pág. vii, 36, 37, 38
- Aust et al. (1995)** Harald Aust, Martin Oerder, Frank Seide e Volker Steinbiss. The philips automatic train timetable information system. *Speech Commun.*, 17(3-4):249–262. ISSN 0167-6393. doi: [http://dx.doi.org/10.1016/0167-6393\(95\)00028-M](http://dx.doi.org/10.1016/0167-6393(95)00028-M). Citado na pág. ix, 22, 23
- Austin (1962)** J. L. Austin. *How to do things with words*. Oxford University Press, London, 1st ed. ed. ISBN 019824553. Citado na pág. 12, 13
- Benferhat et al. (2004)** Salem Benferhat, Souhila Kaci, Daniel Le Berre e Mary-Anne Williams. Weakening conflicting information for iterated revision and knowledge integration.

- Artif. Intell.*, 153(1-2):339–371. ISSN 0004-3702. doi: <http://dx.doi.org/10.1016/j.artint.2003.08.003>. Citado na pág. 68
- Bischoff e Graefe (2002)** R. Bischoff e V. Graefe. Dependable multimodal communication and interaction with robotic assistants. Em *International Workshop on Robot and Human Interactive Communication*, páginas 300–305. IEEE, IEEE. ISBN 0-7803-7545-9. doi: 10.1109/ROMAN.2002.1045639. URL <http://dx.doi.org/10.1109/ROMAN.2002.1045639>. Citado na pág. vii, 40
- Bischoff e Graefe (2003)** Rainer Bischoff e Volker Graefe. HERMES - an Intelligent Humanoid Robot Designed and Tested for Dependability. *Springer Tracts in Advanced Robotics*, 5:64 – 74. Citado na pág. 40
- Black et al. (1999)** A. W. Black, P. Taylor e R. Caley. The Festival Speech Synthesis System: system documentation. Relatório Técnico Edition 1.4, for Festival Version 1.4.0, University of Edinburgh, Scotland, UK,. Citado na pág. 59
- Bohus e Horvitz (2009)** Dan Bohus e Eric Horvitz. Dialog in the open world: platform and applications. Em *Proceedings of the 2009 international conference on Multimodal interfaces, ICMI-MLMI '09*, páginas 31–38, New York, NY, USA. ACM. ISBN 978-1-60558-772-1. doi: 10.1145/1647314.1647323. URL <http://doi.acm.org/10.1145/1647314.1647323>. Citado na pág. 48
- Bohus e Rudnicky (2009)** Dan Bohus e Alexander I. Rudnicky. The ravenclaw dialog management framework: Architecture and systems. *Computer Speech and Language*, 23(3):332 – 361. ISSN 0885-2308. doi: 10.1016/j.csl.2008.10.001. URL <http://www.sciencedirect.com/science/article/pii/S0885230808000545>. Citado na pág. 45, 48
- Bohus et al. (2007)** Dan Bohus, Antoine Raux, Thomas K. Harris, Maxine Eskenazi e Alexander I. Rudnicky. Olympus: an open-source framework for conversational spoken language interface research. Em *NAACL-HLT '07: Proceedings of the Workshop on Bridging the Gap*, páginas 32–39, Morristown, NJ, USA. Association for Computational Linguistics. Citado na pág. 46
- Boros et al. (1996)** M. Boros, W. Eckert, F. Gallwitz, G. Görz, G. Hanrieder e H. Niemann. Towards Understanding Spontaneous Speech: Word Accuracy vs. Concept Accuracy. Em *4th International Conference on Spoken Language Processing*, páginas 1009–1012, Philadelphia, PA, USA. Citado na pág. 47
- Bos et al. (2003a)** Johan Bos, Ewan Klein, Oliver Lemon e Tetsushi Oka. DIPPER: Description and Formalisation of an Information-State Update Dialogue System Architecture. Em *4th SIGdial Workshop on Discourse and Dialogue*, Sapporo, Japan. Citado na pág. 44

- Bos et al. (2003b)** Johan Bos, Ewan Klein e Tetsushi Oka. Meaningful conversation with a mobile robot. Em *EACL '03: Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics*, páginas 71–74, Morristown, NJ, USA. Association for Computational Linguistics. ISBN 1-111-56789-0. Citado na pág. 54
- Bratman et al. (1991)** Michael E. Bratman, David Israel e Martha Pollack. Plans and resource-bounded practical reasoning. Em Robert Cummins e John L. Pollock, editors, *Philosophy and AI: Essays at the Interface*, páginas 1–22. The MIT Press, Cambridge, Massachusetts. Citado na pág. 22
- Brazdil e Torgo (1990)** Pavel B. Brazdil e Luis Torgo. Knowledge acquisition via knowledge integration. Em B. Wielenga et al., editor, *Current Trends in AI*. IOS Press. Citado na pág. 68
- Brill (1995)** Eric Brill. Transformation-based error-driven learning and natural language processing: A case study in part of speech tagging. *Computational Linguistics*, 21(4):543–565. ISSN 0891-2017. Citado na pág. 65
- Brondsted (1999)** T Brondsted. The CPK NLP Suite for Spoken Language Understanding. Em *Eurospeech'99*, páginas 2655–2658, Hungary. Citado na pág. 4
- Chu et al. (2005)** Shiu-Wah Chu, Ian O'Neill, Philip Hanna e Michael McTear. An Approach to Multi-Strategy Dialogue Management. Em *Interspeech 2005*, páginas 865–868. implementação de um SDS a partir de um sistema de question answering. Citado na pág. 26
- Clark (1996)** Herbert H. Clark. *Using Language*. Cambridge University Press, Cambridge, UK. ISBN 0521567459. Citado na pág. 14
- Clark e Schaefer (1989)** Herbert H. Clark e Edward F. Schaefer. Contributing to discourse. *Cognitive Science*, 13(2):259 – 294. ISSN 0364-0213. doi: DOI:10.1016/0364-0213(89)90008-6. URL <http://www.sciencedirect.com/science/article/B6W48-4B4RPJJ-12/2/67182c1ff4c2e8271f45c53e6af1d0e6>. Citado na pág. 14
- Clark e Brennan (1991)** HH Clark e SE Brennan. Grounding IN Communication. Em Resnick, LB and Levine, JM and Teasley, SD, editor, *Perspectives on Socially Shared Cognition*, páginas 127–149. Amer Psychol Assoc, Amer Psychological Assoc. ISBN 1-55798-121-3. Conf on Socially Shared Cognition, Pittsburgh, Pa, Feb, 1989. Citado na pág. 14
- Daelemans et al. (2002)** W. Daelemans, J. Zavrel, A. van den Bosch e K. van der Sloot. TiMBL: Tilburg Memory-Based Learner Reference Guide. Relatório Técnico version 4.2, University of Tilburg, P.O. Box 90153, NL-5000 LE. Citado na pág. 58, 64

- Dalianis (1996)** H. Dalianis. *Concise Natural Language Generation from Formal Specifications*. Issn 1101-8526, Royal Institute of Technology, Stockholm University. Citado na pág. 76
- Dautenhahn (2004)** K. Dautenhahn. Robots we like to live with?! – a developmental perspective on a personalized, life-long robot companion. Em *13th IEEE International Workshop on Robot and Human Interactive Communication, 2004 (ROMAN 2004)*, páginas 17–22. doi: 10.1109/ROMAN.2004.1374720. Citado na pág. 2
- Dautenhahn et al. (2005)** K. Dautenhahn, S. Woods, C. Kaouri, M.L. Walters, Kheng Lee Koay e I. Werry. What is a robot companion – friend, assistant or butler? Em *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2005 (IROS 2005)*, páginas 1192–1197. doi: 10.1109/IROS.2005.1545189. Citado na pág. 2
- Denecke (2002)** Matthias Denecke. Rapid prototyping for spoken dialogue systems. Em *COLING*. arquitetura para desenvolvimento do SDS. Citado na pág. 42, 43
- Dignum e Greaves (2000)** Frank Dignum e Mark Greaves. *Issues in Agent Communication*, chapter Issues in Agent Communication: An Introduction, páginas 1–16. Springer. Citado na pág. 28
- do Mar (2006)** Pedro do Mar. Agente de Geração de Linguagem Natural para um Robô Móvel Inteligente. Relatório técnico, Universidade de Aveiro. Citado na pág. 76
- Dutoit (1997)** Thierry Dutoit. *An introduction to text-to-speech synthesis*. Kluwer Academic Publishers, Norwell, MA, USA. ISBN 0-7923-4498-7. Citado na pág. 19
- Dybkjear et al. (2004)** Laila Dybkjear, Niels Ole Bernsen e Wolfgang Minker. Evaluation and Usability of Multimodal Spoken Language Dialogue Systems. *Speech Communication*, 43:33–54. Citado na pág. 46, 47
- Elhadad (1993)** M Elhadad. Fuf: the universal unifier user manual, version 5.2. Relatório técnico, Columbia University, New York. Citado na pág. 77, 78
- Elhadad e Robin (1998)** Michael Elhadad e Jacques Robin. Surge: a comprehensive plug-in syntactic realization component for text generation. Relatório técnico. Citado na pág. 78
- Eliasson (2005)** Karolina Eliasson. Towards a robotic dialogue system with learning and planning capabilities. Em *IJCAI'2005 4th Workshop on Knowledge and Reasoning in Practical Dialogue Systems*. Citado na pág. 7
- Fabrizio e Lewis (2004)** Giuseppe Di Fabrizio e Charles Lewis. Florence: a dialogue manager framework for spoken dialogue systems. Em *ICSLP 2004, 8th International Conference on Spoken Language Processing*, Jeju, Jeju Island, Korea. Citado na pág. vii, 31, 32

- García-Serrano et al. (2004)** Ana García-Serrano, Paloma Martínez e Josefa Z. Hernández. Using ai techniques to support advanced interaction capabilities in a virtual assistant for e-commerce. *Expert Syst. Appl.*, 26(3):413–426. Citado na pág. 33
- Goodwin e Heritage (1990)** Charles Goodwin e John Heritage. Conversation analysis. *Annual Review of Anthropology*, 19(1):283–307. doi: 10.1146/annurev.an.19.100190.001435. URL <http://www.annualreviews.org/doi/abs/10.1146/annurev.an.19.100190.001435>. Citado na pág. 15
- Grice (1975)** H. P. Grice. Logic and conversation. Em Peter Cole e Jerry L. Morgan, editors, *Syntax and semantics*, volume 3. New York: Academic Press. Citado na pág. 15
- Haasch et al. (2004)** A. Haasch, S. Hohenner, S. Hüwel, M. Kleinhagenbrock, S. Lang, I. Topsis, G. A. Fink, J. Fritsch, B. Wrede e G. Sagerer. Biron – the bielefeld robot companion. Em E. Prassler, G. Lawitzky, P. Fiorini e M. Hägele, editors, *Proc. Int. Workshop on Advances in Service Robotics*, páginas 27–32, Stuttgart, Germany. Fraunhofer IRB Verlag, Fraunhofer IRB Verlag. URL <files/papers/Haasch2004-BTB.ps.gz>. Citado na pág. 38
- Hanafiah et al. (2004)** Zaliyana Mohd Hanafiah, Chizu Yamazaki, Akio Nakamura e Yoshinori Kuno. Understanding inexplicit utterances using vision for helper robots. Em *ICPR '04: Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 4*, páginas 925–928, Washington, DC, USA. IEEE Computer Society. ISBN 0-7695-2128-2. doi: <http://dx.doi.org/10.1109/ICPR.2004.959>. Citado na pág. 7
- Hanna et al. (2005)** Philip Hanna, Ian O'Neill, Xingkun Liu e Michael McTear. Developing Extensible and Reusable Spoken Dialogue Components: An Examination of the Queen's Communicator. Em *Interspeech 2005*, páginas 1865–1868. DARPA communicator, java, orientado a objecto, parecido com OAA. Citado na pág. 28
- Hutchby e Wooffitt (1998)** Ian Hutchby e Robin Wooffitt. *Conversation analysis: principles, practices and applications*. Polity Press, Cambridge, UK. Citado na pág. 15
- Johnson e Agah (2009)** David Johnson e Arvin Agah. Human robot interaction through semantic integration of multiple modalities, dialog management, and contexts. *International Journal of Social Robotics*, 1:283–305. ISSN 1875-4791. URL <http://dx.doi.org/10.1007/s12369-009-0028-0>. 10.1007/s12369-009-0028-0. Citado na pág. 7, 41
- Jurafsky e Martin (2008)** Daniel Jurafsky e James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition (2nd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2nd ed. ISBN 0131873210. Citado na pág. 11, 19, 24, 25

- Kiatissevi et al. (2004)** Pattara Kiatissevi, Vutichai Ampornaramveth e Haruki Ueno. Knowledge-based interactive robot: System architecture and dialogue manager. Em Chengqi Zhang, Hans W. Guesgen e Wai-Kiang Yeap, editors, *PRICAI*, volume 3157 of *Lecture Notes in Computer Science*, páginas 673–682. Springer. ISBN 3-540-22817-9. Citado na pág. 7
- Kiatissevi et al. (2006)** Pattara Kiatissevi, Vutichai Ampornaramveth e Haruki Ueno. A frame-based knowledge software tool for developing interactive robots. *International Journal of Artificial Life and Robotics*, 10(1):18–28. Citado na pág. 7
- Koch et al. (2008)** Jan Koch, Holger Jung, Jens Wettach, Geza Nemeth e Karsten Berns. Dynamic speech interaction for robotic agents. Em Sukhan Lee, Il Suh e Mun Kim, editors, *Recent Progress in Robotics: Viable Robotic Service to Human*, volume 370 of *Lecture Notes in Control and Information Sciences*, páginas 303–315. Springer Berlin, Heidelberg. ISBN 978-3-540-76728-2. Citado na pág. 7, 41
- Konolige et al. (1997)** K. Konolige, K.L. Myers, E.H. Ruspini e A. Saffiotti. The saphira architecture: a design for autonomy. *J. of Experimental and Theoretical Artificial Intelligence*, 9(2-3):215–235. Citado na pág. 52, 60
- Konolige (2001)** K. G. Konolige. *Saphira Software Manual. Saphira Version 8. Saphira/Aria Integration*. SRI International, Menlo Park, Califórnia, 2001. Citado na pág. 52
- Labrou e Finin (1997)** Y. Labrou e T Finin. A Proposal for a New KQML Specification. Relatório Técnico CS-97-03, University of Maryland, Baltimore County. Citado na pág. 61
- Labrou e Finin (1998)** Yannis Labrou e Tim Finin. *Intelligent Agents IV Agent Theories, Architectures, and Languages*, volume 1365, chapter Semantics for an agent communication language, páginas 209 –214. Springer. Citado na pág. 28
- Lappin e Leass (1994)** Shalom Lappin e Herbert J. Leass. An algorithm for pronominal anaphora resolution. *Computational Linguistics*, 20(4):535–561. ISSN 0891-2017. Citado na pág. 101
- Larsson e Traum (2000)** Staffan Larsson e David R. Traum. Information state and dialogue management in the TRINDI Dialogue Move Engine Toolkit. *Natural Language Engineering*, 6(3-4):323–340. ISSN 1351-3249. Citado na pág. vii, 25, 26, 44
- Lauria et al. (2001)** Stanislaw Lauria, Guido Bugmann, Theocharis Kyriacou, Johan Bos e Ewan Klein. Training personal robots using natural language instruction. *IEEE Intelligent Systems*, 16(5):38–45. ISSN 1541-1672. doi: <http://dx.doi.org/10.1109/5254.956080>. Citado na pág. 7

- LeBlanc (2010)** Kevin LeBlanc. *Cooperative anchoring [Elektronisk resurs] : sharing information about objects in multi-robot systems*. Örebro universitet, Örebro. ISBN 978-91-7668-754-3. Citado na pág. 7
- Levin et al. (2000)** Esther Levin, Roberto Pieraccini e Wieland Eckert. A stochastic model of human-machine interaction for learning dialog strategies. *IEEE Transactions on Speech and Audio Processing*, 8(1):11–23. Citado na pág. 25
- Levinson (1983)** Stephen C. Levinson. *Pragmatics*. Cambridge University Press, Cambridge. Citado na pág. 11
- Li et al. (2006)** Shuyin Li, Britta Wrede e Gerhard Sagerer. A computational model of multi-modal grounding for human robot interaction. Em *Proceedings of the 7th SIGdial Workshop on Discourse and Dialogue*, SigDIAL '06, páginas 153–160, Stroudsburg, PA, USA. Association for Computational Linguistics. ISBN 1-932432-71-X. URL <http://dl.acm.org/citation.cfm?id=1654595.1654626>. Citado na pág. 40
- Loutfi et al. (2008)** Amy Loutfi, Silvia Coradeschi, Marios Daoutis e Jonas Melchert. Using knowledge representation for perceptual anchoring in a robotic system. *International Journal on Artificial Intelligence Tools*, 17(5):925–944. Citado na pág. 7, 41
- Marcus et al. (1993)** Mitchell P. Marcus, Mary Ann Marcinkiewicz e Beatrice Santorini. Building a large annotated corpus of english: the penn treebank. *Computational Linguistics*, 19(2):313–330. ISSN 0891-2017. Citado na pág. 65
- Martin et al. (1999)** D. Martin, A. Cheyer e D. Moran. The Open Agent Architecture: a Framework for Building Distributed Software Systems. *Applied Artificial Intelligence*, 13(1/2):91–128. Citado na pág. 28, 54, 131
- McCune (1998)** William McCune. Automatic proofs and counterexamples for some ortho-lattice identities. *Information Processing Letters*, 65(6):285 – 291. ISSN 0020-0190. doi: DOI:10.1016/S0020-0190(98)00015-5. URL <http://www.sciencedirect.com/science/article/B6V0F-3T5GNSW-1/2/1be2d8e0c519fa40c0d9af0c7286000f>. Citado na pág. 45
- McKay et al. (1994)** Don McKay, Tim Finin, Rich Fritzson e Robin McEntire. Kqml – a language and protocol for knowledge and information exchange. Relatório Técnico WS-94-02, Computer Science Department, University of Maryland e Valley Forge Engineering Center, Unisys Corporation. Citado na pág. 28
- McTear (2002)** Michael F. McTear. Spoken dialogue technology: enabling the conversational user interface. *ACM Comput. Surv.*, 34(1):90–169. ISSN 0360-0300. doi: <http://doi.acm.org/10.1145/505282.505285>. Citado na pág. vii, ix, 19, 20, 21, 22, 24

- Michalski (1994)** Ryszard S. Michalski. Inferential Theory of Learning: Developing Foundations for Multistrategy Learning. Em *Machine Learning: A Multistrategy Approach*, chapter 1, páginas 3–61. Morgan Kaufmann. Citado na pág. 68, 69, 70
- Mikheev (1997)** Andrei Mikheev. Automatic rule induction for unknown word guessing. *Computational Linguistics*, 23(3):405–423. Citado na pág. 65
- Mohan e Laird (2011)** Shiwali Mohan e John Laird. Towards situated, interactive, instructable agents in a cognitive architecture, 2011. URL <http://www.aaai.org/ocs/index.php/FSS/FSS11/paper/view/4165>. Citado na pág. 7
- Nakadai et al. (2008)** Kazuhiro Nakadai, Shun'ichi Yamamoto, Hiroshi G. Okuno, Hirofumi Nakajima, Yuji Hasegawa e Hiroshi Tsujino. A robot referee for rock-paper-scissors sound games. Em *ICRA*, páginas 3469–3474. IEEE. Citado na pág. 48, 130
- Nakano et al. (2005)** M. Nakano, Y. Hasegawa e Nakadai. A Two Layer Model for Behavior and Dialogue Planning in Conversational Service Robots. Em *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, páginas 1542–1548. Citado na pág. vii, 34, 35, 36
- Nickerson (1977)** R. S. Nickerson. On conversational interaction with computers. Em *UODIGS '76: Proceedings of the ACM/SIGGRAPH workshop on User-oriented design of interactive graphics systems*, páginas 101–113, New York, NY, USA. ACM. doi: <http://doi.acm.org/10.1145/1024273.1024286>. Citado na pág. 9
- Nuance (2001)** Nuance. *Application Developer's Guide: Nuance Speech Recognition System Version 8.0*. Nuance, 2001. Citado na pág. 114
- Ogura et al. (2003)** Yu. Ogura, S. Ando, N. Hieda, Hun-ok Lim e A. Takanishi. Macro-walking instruction for biped humanoid robot. volume 1, páginas 417 – 422 vol.1. Citado na pág. 7
- Oh et al. (2009)** SeungSub Oh, Young-Ho Choi, SungYong Yun, Bong-Jin Jun, Cheongjae Lee, Hyukjoon Jang, Jaiyoun Song, Jeong-Gwan Kang, Won-Seok Choi e Su-Yong An. A system architecture for intelligent building guide robot phope. Em *Autonomous Robots and Agents, 2009. ICARA 2009. 4th International Conference on*, páginas 640–645. Citado na pág. 48, 130
- OMG (1995)** OMG. The Common Object Request Broker: Architecture and Specification. Relatório Técnico revision 2, Object Management Group. Citado na pág. 28
- Peltason e Wrede (2010)** Julia Peltason e Britta Wrede. Pamini: a framework for assembling mixed-initiative human-robot interaction from generic interaction patterns. Em

- Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, SIGDIAL '10, páginas 229–232, Stroudsburg, PA, USA. Association for Computational Linguistics. ISBN 978-1-932432-85-5. URL <http://dl.acm.org/citation.cfm?id=1944506.1944546>. Citado na pág. 48
- Peltason e Wrede (2011)** Julia Peltason e Britta Wrede. The curious robot as a case-study for comparing dialog systems. *AI Magazine*, 32(4):85–99. Citado na pág. 49
- Pomerantz (1984)** A. M. Pomerantz. Agreeing and disagreeing with assessment: Some features of preferred/dispreferred turn shapes. Em J. M. Atkinson e J. Heritage, editors, *Structure of Social Action: Studies in Conversation Analysis*, páginas 57 – 101. Cambridge University Press. Citado na pág. 15
- Qi et al. (2007)** Guilin Qi, Weiru Liu e David Bell. Combining multiple prioritized knowledge bases by negotiation. *Fuzzy Sets Syst.*, 158(23):2535–2551. ISSN 0165-0114. doi: <http://dx.doi.org/10.1016/j.fss.2007.02.013>. Citado na pág. 68
- Quinderé et al. (2007a)** Marcelo Quinderé, Luís Seabra Lopes e António Teixeira. A Dialogue Manager for an Intelligent Mobile Robot. Em Bernadette Sharp, editor, *Natural Language Processing and Cognitive Science (NLPCS)*, Madeira, Portugal. Citado na pág. 7
- Quinderé et al. (2007b)** Marcelo Quinderé, Luís Seabra Lopes, António Teixeira e Mário Rodrigues. An Information State Based Dialogue Manager for a Mobile Robot. Em *Interspeech'2007 - Eurospeech - 10th European Conference on Speech Communication and Technology*, Antwerp, Belgium. Citado na pág. 7, 107, 109, 133
- Quinderé et al. (2013)** Marcelo Quinderé, Luís Seabra Lopes e António Teixeira. Evaluation of a dialogue manager for a mobile robot. Em *22th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN 2013)*. Citado na pág. 7
- Reitter (2004)** David Reitter. A Development Environment for Multimodal Functional Unification Generation Grammars. Em *3rd International Natural Language Generation Conference*, Brockenhurst, UK. Citado na pág. 59, 79
- Rodrigues (2004)** Mário Rodrigues. Interação humano-robô ao nível da linguagem. Dissertação de Mestrado, Universidade de Aveiro. Citado na pág. 61, 65
- Rodrigues et al. (2004)** Mário Rodrigues, António Teixeira e Luís Seabra Lopes. An Hybrid Approach for Spoken Natural Language Understanding Applied to a Mobile Intelligent Robot. Em Bernadette Sharp, editor, *Natural Language Understanding and Cognitive Science (NLUCS)*, páginas 145–150, Porto, Portugal. INSTICC Press. ISBN 972-8865-05-8. Citado na pág. viii, 63, 65

- Rosé e Lavie (1998)** C. P. Rosé e A. Lavie. LCFlex: An Efficient Robust Left-Corner Parser. Relatório técnico, University of Pittsburgh. Citado na pág. 58, 63
- Rudnicky et al. (2010)** Alexander Rudnicky, Aasish Pappu, Peng Li, Matthew Marge e Benjamin Frisch. Instruction taking in the teamtalk system, 2010. URL <http://aaai.org/ocs/index.php/FSS/FSS10/paper/view/2217/2724>. Citado na pág. 48
- Rudnicky et al. (1999)** Alexander I. Rudnicky, E. Thayer, Paul Constantinides, C. Tchou, R. Shern, Kevin Lenzo, W. Xu e A. Oh. Creating natural dialogs in the carnegie mellon communicator system. Em *EUROSPEECH: European Conference on Speech Processing*, páginas 1531–1534, Budapest, Hungary. Citado na pág. 45
- Rumbaugh et al. (2004)** James Rumbaugh, Ivar Jacobson e Grady Booch. *Unified Modeling Language Reference Manual, The (2nd Edition)*. Pearson Higher Education. ISBN 0321245628. Citado na pág. 62
- Russell e Norvig (2010)** Stuart Russell e Peter Norvig. *Artificial Intelligence: A Modern Approach (3rd Edition)*. Prentice Hall, 3 ed. ISBN 0136042597. Citado na pág. 62
- Sacks et al. (1974)** H. Sacks, E. A. Schegloff e G. Jefferson. A simplest systematics for the organization of turn-taking for conversation. *Language*, 50(4):696–735. Citado na pág. 12
- Samani et al. (2011)** Hooman Aghaebrahimi Samani, Adrian David Cheok e Owen Noel Newton Fernando. An affective interactive audio interface for lovotics. *Comput. Entertain.*, 9(2):6:1–6:14. ISSN 1544-3574. doi: 10.1145/1998376.1998377. URL <http://doi.acm.org/10.1145/1998376.1998377>. Citado na pág. 7
- Schegloff e Sacks (1973)** E. A. Schegloff e H. Sacks. Opening up closings. *Semiotica*, 8(4): 289–327. Citado na pág. 15
- Seabra Lopes (2001)** Luís Seabra Lopes. Carl, a Learning Robot, serving Food at the AAI Reception. Em *AAAI Mobile Robot Competition and Exhibition Workshop*, páginas 1–7, Seattle, WA. Citado na pág. 3
- Seabra Lopes (2002)** Luís Seabra Lopes. Carl: from situated activity to language level interaction and learning. Em *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2002)*, volume 1, páginas 890 – 896, Lausanne, Switzerland. Citado na pág. vii, 52, 109, 133
- Seabra Lopes e Connell (2001)** Luís Seabra Lopes e Jonathan H. Connell. Guest editors' introduction: Semisentient robots– routes to integrated intelligence. *IEEE Intelligent Systems*, 16(5):10–14. ISSN 1094-7167. doi: <http://dx.doi.org/10.1109/5254.956076>. Citado na pág. 2

- Seabra Lopes e Teixeira (2000)** Luís Seabra Lopes e António Teixeira. Human-Robot Interaction through Spoken Language Dialogue. Em *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000)*, volume 1, páginas 528–534, Japan. Citado na pág. 2, 41, 61
- Seabra Lopes et al. (2003a)** Luís Seabra Lopes, António Teixeira, Mário Rodrigues, D Gomes, C Teixeira, L Ferreira, P Soares, J Girão e N Sénica. Towards a Personal Robot with Language Interface. Em *Eurospeech*, páginas 2205–2208. Citado na pág. 3, 65
- Seabra Lopes et al. (2003b)** Luís Seabra Lopes, António J. S. Teixeira, Mário Rodrigues, D. Gomes, C. Teixeira, L. Ferreira, P. Soares, J. Girao e N. Senica. Towards a personal robot with language interface. Em *INTERSPEECH*. ISCA. Citado na pág. 107
- Seabra Lopes et al. (2005a)** Luís Seabra Lopes, António Teixeira e Marcelo Quinderé. A Knowledge Representation and Reasoning Module for a Dialog System in a Mobile Robot. Em Bernadette Sharp, editor, *Natural Language Understanding and Cognitive Science (NLUCS)*, páginas 172 – 177, Miami, USA. Citado na pág. 6, 63, 66, 132
- Seabra Lopes et al. (2005b)** Luís Seabra Lopes, António Teixeira, Marcelo Quinderé e Mário Rodrigues. From Robust Spoken Language Understanding to Knowledge Acquisition and Management. Em *Interspeech'2005 - Eurospeech - 9th European Conference on Speech Communication and Technology*, páginas 3469 – 3472, Lisbon, Portugal. Citado na pág. 7, 109
- Seabra Lopes et al. (2008)** Luís Seabra Lopes, António Teixeira, Marcelo Quinderé e Mário Rodrigues. Integrated capabilities for knowledge acquisition through spoken language interaction in a mobile robot. Em *8th Conference on Autonomous Robot Systems and Competitions - Robotica 2008*, páginas 79–86, Aveiro, Portugal. Citado na pág. 7
- Seabra Lopes et al. (2009)** Luís Seabra Lopes, António Teixeira, Marcelo Quinderé e Mário Rodrigues. Integrated capabilities for knowledge acquisition through spoken language interaction in a mobile robot. *Robótica – Automação, Controlo e Instrumentação*, 01(74): 16–22. Citado na pág. 7
- Searle (1975)** John Searle. *A taxonomy of illocutionary acts*, páginas 334–369. University of Minnesota Press, Minneapolis. Citado na pág. 13
- Seneff et al. (1998)** S. Seneff, E. Hurley, R. Lau, C. Pao, P. Schmid e V. Zue. Galaxy-ii: A reference architecture for conversational system development. Em *Proc. ICSLP-98*, volume 3, páginas 931–934. Citado na pág. 42
- Singh et al. (2002)** Satinder P. Singh, Diane J. Litman, Michael J. Kearns e Marilyn A. Walker. Optimizing dialogue management with reinforcement learning: Experiments with the njfun system. *J. Artif. Intell. Res. (JAIR)*, 16:105–133. Citado na pág. 25

- Sowa (2005)** J.F. Sowa. Semantic Networks. Em Stuart C. Shapiro, editor, *Encyclopedia of Artificial Intelligence (an revised and extended version of an article originally from 1987)*, <http://www.jfsowa.com/pubs/semnet.htm>, New York, NY, USA. John Wiley and Sons. Citado na pág. 62
- Stalnaker (2002)** Robert Stalnaker. Common ground. *Linguistics and Philosophy*, 25(5): 701 – 721. Citado na pág. 14
- Sung et al. (2009)** JaYoung Sung, Henrik I. Christensen e Rebecca E. Grinter. Sketching the future: Assessing user needs for domestic robots. Em *18th IEEE International Workshop on Robot and Human Interactive Communication (ROMAN 2009)*, Toyama, Japan. Citado na pág. 1
- Tan et al. (2010)** YongChai Tan, BentFei Lew, KhimLeng Tan, KaeVin Goh, KienLoong Lee e ZhiChao Khor. A new automated food delivery system using autonomous track guided centre-wheel drive robot. Em *Sustainable Utilization and Development in Engineering and Technology (STUDENT), 2010 IEEE Conference on*, páginas 32 –36. doi: 10.1109/STUDENT.2010.5686986. Citado na pág. 7
- Ten Bosch et al. (2004)** L. F. M. Ten Bosch, N. H. J. Oostdijk e J. P. de Ruiter. Turn-taking in social talk dialogues: Temporal, formal and functional aspects. Em *Proceedings SPECOM*. URL <http://edoc.mpg.de/226529>. Citado na pág. 12
- Tomko et al. (2005)** Stefanie Tomko, Thomas K. Harris, Arthur Toth, James Sanders, Alexander Rudnicky e Roni Rosenfeld. Towards efficient human machine speech communication: The speech graffiti project. *ACM Trans. Speech Lang. Process.*, 2(1):1–27. ISSN 1550-4875. doi: <http://doi.acm.org/10.1145/1075389.1075391>. Citado na pág. 45
- Toptsis et al. (2005)** I. Toptsis, A. Haasch, S. Hä \tilde{A} $\frac{1}{4}$ wel, J. Fritsch e G.A. Fink. Modality integration and dialog management for a robotic assistant. Em *Proc. European Conf. on Speech Communication and Technology*, Lisboa, Portugal. URL <files/papers/Toptsis2005-MIA.pdf>. Citado na pág. 7
- Traum e Allen (1994)** David R. Traum e James F. Allen. Discourse obligations in dialogue processing. Em James Pustejovsky, editor, *Proceedings of the Thirty-Second Meeting of the Association for Computational Linguistics*, páginas 1–8, San Francisco. Morgan Kaufmann. Citado na pág. ix, 22, 24
- Traum e Larsson (2003)** David R. Traum e Staffan Larsson. The Information State Approach to Dialogue Management. Em Jan van Kuppevelt and Ronnie Smith, editor, *Current and New Directions in Discourse and Dialogue*. Kluwer. Citado na pág. 25, 45

- Vasudevan et al. (2007)** Shrihari Vasudevan, Stefan Gächter, Viet Nguyen e Roland Siegart. Cognitive maps for mobile robots—an object based approach. *Robot. Auton. Syst.*, 55(5):359–371. ISSN 0921-8890. doi: <http://dx.doi.org/10.1016/j.robot.2006.12.008>. Citado na pág. 7
- Wachsmuth et al. (2009)** Sven Wachsmuth, Marc Hanheide, Frederic Siepman e Thorsten P. Spexard. Tobi - team of bielefeld: The human-robot interaction system for robocup@home 2009. Relatório técnico. Citado na pág. vii, 38, 39
- Wachsmuth et al. (2010)** Sven Wachsmuth, Marc Hanheide, Frederic Siepman e Thorsten Spexard. Tobi team of bielefeld: The human-robot interaction system for robocup home 2010. Relatório técnico. Citado na pág. 48
- Walker et al. (2002)** M. Walker, R. Passonneau, J. Aberdeen, J. Boland, E. Bratt, J. Garofolo, L. Hirschman, A. Le, S. Lee, S. Narayanan, K. Papineni, B. Pellom, J. Polifroni, A. Potamianos, P. Prabhu, A. Rudnick, G. Sanders, S. Seneff, D. Stallard e S. Whittaker. Cross-site evaluation in darpa communicator: The june 2000 data collection. Em *Computer Speech and Language*. Citado na pág. 30
- Walker et al. (1997)** Marilyn A. Walker, Diane Litman J., Candace A. Kamm e Alicia Abella. PARADISE: A framework for evaluating spoken dialogue agents. Em Philip R. Cohen e Wolfgang Wahlster, editors, *Proceedings of the Thirty-Fifth Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, páginas 271–280, Somerset, New Jersey. Association for Computational Linguistics. Citado na pág. 47
- Walker et al. (2004)** Willie Walker, Paul Lamere, Philip Kwok, Bhiksha Raj, Rita Singh, Evandro Gouvea, Peter Wolf e Joe Woelfel. Sphinx-4: A Flexible Open Source Framework for Speech Recognition. Relatório Técnico SMLI TR-2004-139, Sun Microsystems. Citado na pág. 58
- Weidenbach (1999)** Christoph Weidenbach. System description: Spass version 1.0.0. Em *CADE-16: Proceedings of the 16th International Conference on Automated Deduction*, páginas 378–382, London, UK. Springer-Verlag. ISBN 3-540-66222-7. Citado na pág. 45
- Weizenbaum (1966)** Joseph Weizenbaum. Eliza—a computer program for the study of natural language communication between man and machine. *Commun. ACM*, 9(1):36–45. ISSN 0001-0782. doi: <http://doi.acm.org/10.1145/365153.365168>. Citado na pág. 18
- Woods (1969)** W. A. Woods. Augmented transition networks for natural language analysis. Relatório Técnico CS-I, Harvard Computation Laboratory, Harvard University, Cambridge, MA. Citado na pág. 32

Apêndice A

Interface de Acesso ao Estado da Informação

Os métodos de acesso ao Estado da Informação são aqui brevemente descritos (observe que a nomenclatura utiliza “IS”, em referência a *Information State*):

- Acesso ao Estado da Informação em geral:
 - `empty_IS_field(+IS_Field)` – retorna *verdadeiro* se o campo *IS_Field* está vazio (não tem valor atribuído), e *falso* caso contrário.
 - `set_IS_value(+IS_Field, +Value)` – atribui *Value* ao campo *IS_Field*.
 - `get_IS_value(+IS_Field, -Value)` – retorna o valor do campo *IS_Field* em *Value*.
 - `valid_IS_value(+IS_Field)` – retorna *verdadeiro* se o campo *IS_Field* tem um valor atribuído, e *falso* caso contrário. Equivale à negação de `empty_IS_field(+IS_Field)`.
 - `clean_IS_field(+IS_Field)` – torna vazio o campo *IS_Field*.
- Acesso a listas no Estado da Informação:
 - `empty_IS_list(+IS_Field)` – retorna *verdadeiro* se a lista *IS_Field* estiver vazia, e *falso* caso contrário.
 - `get_IS_list_value(+IS_Field, -Value)` – retorna a lista definida por *IS_Field* em *Value*.
 - `get_IS_list_length(+IS_Field, -Length)` – retorna o tamanho da lista identificada por *IS_Field* em *Length*.
 - `add_IS_list_value(+IS_Field, +Value)` – adiciona *Value* no início da lista identificada por *IS_Field*.

- `add_IS_sorted_list_value(+IS_Field, +Value)` – adiciona *Value* à lista identificada por *IS_Field* mantendo a lista ordenada de maneira decrescente.
- `get_IS_list_head(+IS_Field, -Value)` – retorna o valor do primeiro elemento da lista identificada por *IS_Field* em *Value*.
- `get_IS_list_tail(+IS_Field, -Value)` – retorna a cauda da lista identificada por *IS_Field* em *Value*.
- `get_IS_list_member(+IS_Field, +Pos, -Value)` – retorna em *Value* o valor do elemento posicionado em *Pos* na lista identificada por *IS_Field*. Falha se a posição *Pos* for maior do que o comprimento da lista.
- `isfirst_IS_list(+IS_Field, +Value)` – retorna *verdadeiro* se o primeiro elemento da lista identificada por *IS_Field* for *Value*, caso contrário retorna *falso*.
- `ismember_IS_list(+IS_Field, +Value)` – retorna *verdadeiro* se *Value* faz parte da lista identificada por *IS_Field*, caso contrário, retorna *falso*.
- `get_first_n_positions_IS_list(+IS_Field, +N, -SubList)` – retorna a sublista *SubList* composta das primeiras *N* posições da lista identificada por *IS_Field*.
- `delete_IS_list_value(+IS_Field, +Value)` – remove todas as ocorrências de *Value* da lista identificada por *IS_Field*.
- `clean_IS_list(+IS_Field)` – torna vazia a lista *IS_Field*.

Apêndice B

Acesso às Tarefas no Estado da Informação

Abaixo listamos os procedimentos específicos para tarefas, definidos por conveniência, que manipulam diretamente o campo *task_list* do Estado da Informação:

- `task_list(-TaskList)` – equivale a `get_IS_value(task_list, -TaskList)`.
- `add_task(+Task)` – procedimento que adiciona uma tarefa à lista de tarefas, utiliza procedimentos auxiliares para associar uma prioridade à tarefa e depois invoca `add_IS_sorted_list_value(task_list, +task(Priority, Task, Plan))`.
- `in_task_list(+Task)` – equivale a `ismember_IS_list(task_list, +Task)`.
- `delete_task(+Task)` – equivale a `delete_IS_list_value(task_list, +Task)`.
- `empty_task_list()` – equivale a `empty_IS_list(task_list)`.
- `get_task_priority(+Task, -Priority)` – retorna a prioridade (*Priority*) da tarefa *Task*.
- `set_task_priority(+Task, +Priority)` – altera dinamicamente o valor da prioridade da tarefa *Task* para *Priority*.

B.1 Exemplos de Tarefas

B.1.1 Abertura

A tarefa que vemos em B.1 mostra as ações que devem ser executadas na fase de abertura do diálogo. Podemos ver três mensagens para o agente NLG: uma de saudação, uma de apresentação e última de oferta de ajuda. Além disso, o plano também inclui a tarefa de obter o nome do utilizador.

Tarefa : opening

Prioridade : 5

$$\text{Plano : } \left\{ \begin{array}{l} \text{oaa_action(nlg(greet, keep_turn, _, _, _, _))} \\ \text{oaa_action(nlg(introduce, keep_turn, _, _, _, _))} \\ \text{task(get_user_name(UserName))} \\ \text{oaa_action(nlg(offer_help, release_turn, _, _, _, _))} \end{array} \right. \quad (\text{B.1})$$

B.1.2 Aquisição do Nome do Utilizador

O plano da tarefa para aquisição do nome do utilizador consiste em, primeiramente, verificar o valor do campo *user_name* do Estado da Informação. Se este campo estiver vazio, o plano pergunta o nome ao utilizador. Se não houver uma resposta em 30 segundos, o nome fictício 'John Doe' é utilizado. Ao final, é retornado o valor do campo *user_name*. Isto pode ser visto em B.2.

Tarefa : get_user_name(-UserName)

Prioridade : 7

$$\text{Plano : } \left\{ \begin{array}{l} \text{if_then(} \\ \quad \text{empty_IS_value(user_name),} \\ \quad \text{[oaa_action(nlg(ask_name, release_turn, _, _, _, _))}, \\ \quad \text{wait_condition(valid_IS_value(user_name), 30,} \\ \quad \quad \text{set_IS_value(user_name, 'John Doe'))]} \\ \quad \text{)} \\ \text{action(get_IS_value(user_name, UserName))} \end{array} \right. \quad (\text{B.2})$$

B.1.3 Rejeição – Confiança Muito Baixa do Agente de Reconhecimento de Fala Automático

A tarefa B.3 lida com a situação na qual a confiança do agente de reconhecimento de fala automático é considerada extremamente baixa, tão baixa que não valeria a pena utilizar a semântica extraída. O plano aqui consiste em simplesmente enviar ao agente NLG uma mensagem com o ato de fala *rejection*, indicando a rejeição do que foi reconhecido pelo ASR.

Tarefa : reject

Prioridade : 5

$$\text{Plano : } \left\{ \text{oaa_action(nlg(rejection, release_turn, none, none, [], _))} \right. \quad (\text{B.3})$$

B.1.4 Clarificação – Análise Semântica Parcial

A tarefa B.4 lida com os casos de análise semântica realizada de forma superficial (ver capítulo 3, seção 3.2). Seu plano consiste em gerar uma pergunta de clarificação baseada na semântica que foi extraída.

$$\begin{aligned} \text{Tarefa} &: \text{clarify_shallow}(+\text{Semantics}) \\ \text{Prioridade} &: 5 \\ \text{Plano} &: \left\{ \begin{array}{l} \text{action}(\text{generate_shallow_ask}(\text{Semantics}, \text{Ask})) \\ \text{oaa_action}(\text{nlg}(\text{clarify}, \text{release_turn}, \text{shallow}, \text{none}, \text{Ask}, _)) \end{array} \right. \end{aligned} \quad (\text{B.4})$$

B.1.5 Bateria

A tarefa B.5 trata da situação de nível baixo da bateria, enquanto a tarefa B.6 lida com o nível da bateria crítico. Observe que ambas possuem uma prioridade bastante elevada, 12 e 15, respectivamente, e que, além de parar o robô e informar o nível da bateria, a tarefa que lida com o nível crítico também se despede do utilizador e encerra a interação.

$$\begin{aligned} \text{Tarefa} &: \text{handle_low_battery} \\ \text{Prioridade} &: 12 \\ \text{Plano} &: \left\{ \begin{array}{l} \text{oaa_action}(\text{navigation}(\text{stop})) \\ \text{oaa_action}(\text{nlg}(\text{inform}, \text{release_turn}, \text{low_battery}, _, _, _)) \end{array} \right. \end{aligned} \quad (\text{B.5})$$

$$\begin{aligned} \text{Tarefa} &: \text{handle_critical_battery} \\ \text{Prioridade} &: 15 \\ \text{Plano} &: \left\{ \begin{array}{l} \text{oaa_action}(\text{navigation}(\text{stop})) \\ \text{oaa_action}(\text{nlg}(\text{inform}, \text{keep_turn}, \text{critical_battery}, _, _, _)) \\ \text{oaa_action}(\text{nlg}(\text{bye}, \text{release_turn}, _, _, _, _)) \\ \text{action}(\text{set_IS_value}(\text{program_state}, \text{stop})) \end{array} \right. \end{aligned} \quad (\text{B.6})$$

B.1.6 Ambiente Ruidoso

O plano da tarefa B.7 simplesmente sugere uma mudança de ambiente ao utilizador. Como se verá, esta tarefa é executada tipicamente em situações de ambiente ruidoso.

$$\begin{aligned}
& \text{Tarefa : change_place} \\
& \text{Prioridade : 6} \\
& \text{Plano : } \left\{ \text{oaa_action(nlg(suggest, release_turn, change_place, _, _, _))} \right.
\end{aligned} \tag{B.7}$$

B.2 Exemplos de Regras Gerais

B.2.1 Saudação

A regra B.8 é aplicada quando o utilizador faz uma saudação ao robô. Sobre o significado da estrutura semântica $\text{sem}(\text{SA}, \text{SA_data}, _, \text{Rels})$, consulte-se a seção 3.2.4. Podemos observar que esta regra possui, como condição para sua aplicação, o reconhecimento de um ato de fala do tipo *greet* (saudação), e como efeito a adição da tarefa *opening*.

$$\begin{aligned}
& \text{Regra : greetings} \\
& \text{LC : } \left\{ \begin{array}{l} \text{valid_IS_value(input_resolved)} \\ \text{get_IS_value(input_resolved, input(RecConf, sem(SA, SA_data, _, Rels)))} \\ \text{SA = greet} \end{array} \right. \\
& \text{LE : } \left\{ \text{add_task(opening)} \right.
\end{aligned} \tag{B.8}$$

B.2.2 Despedida

A regra B.9 é aplicada quando o utilizador se despede do robô.

$$\begin{aligned}
& \text{Regra : bye} \\
& \text{LC : } \left\{ \begin{array}{l} \text{valid_IS_value(input_resolved)} \\ \text{get_IS_value(input_resolved, input(RecConf, sem(SA, SA_data, _, Rels)))} \\ \text{SA = bye} \end{array} \right. \\
& \text{LE : } \left\{ \text{add_task(goodbye)} \right.
\end{aligned} \tag{B.9}$$

B.2.3 Bateria

Dois regras lidam com a carga da bateria. A regra B.10 adiciona a tarefa para lidar com a situação de baixa bateria, enquanto a regra B.11 é aplicada quando o nível da bateria já se encontra crítico.

Regra : lowBattery

$$\begin{aligned} \text{LC} : & \left\{ \begin{array}{l} \text{navigation_event(battery, BatteryValue)} \\ \text{BatteryValue} < 10\% \\ \text{BatteryValue} > 5\% \end{array} \right. & \text{(B.10)} \\ \text{LE} : & \left\{ \text{add_task(handle_low_battery)} \right. \end{aligned}$$

Regra : criticalBattery

$$\begin{aligned} \text{LC} : & \left\{ \begin{array}{l} \text{navigation_event(battery, BatteryValue)} \\ \text{BatteryValue} < 5\% \end{array} \right. & \text{(B.11)} \\ \text{LE} : & \left\{ \text{add_task(handle_critical_battery)} \right. \end{aligned}$$

Apêndice C

Frases Utilizadas nas Experiências para Avaliação de Funcionalidades

A tabela C.1 mostra a lista completa de frases que foram utilizadas nas experiências para avaliação de funcionalidades, descritas no capítulo 5, seção 5.1.

Tabela C.1: Frases utilizadas – avaliação de funcionalidades

Frases
Peter likes the lab
A robot likes music
A man likes car
A man likes robot
A car has wheel
A robot has battery
A country has city
A man has a head
A robot has a computer
A computer has disk
A lab has a computer
Paul is in the country
Peter is at the lab
Peter is in the area
Tony is in Canada
Tony is a director of Peter
Peter is a coordinator of Paul
Paul is a professor of Jim
Jim is a coordinator of Tony
Peter is a doctor
Tony is a lawyer
Paul is a scientist
Canada is a country
Paul is a robot
France is a country
Peter is a man
A country is a place
A robot is a machine
A man is a mammal
A car is a machine
A computer is a machine
A disk is a machine
A lab is a place
The professor of Peter likes France
The director of Tony likes Canada
The coordinator of Paul likes the lab
The professor of Jim likes the area
The director of Peter likes the professor
The professor of peter is in france
The lab coordinator is in the country
the professor of mary likes the company president

Apêndice D

Frases Utilizadas nas Experiências para Resolução de Pronomes

A tabela D.1 mostra a lista com os pares de frases que foram utilizados na experiência para avaliação da resolução de pronomes, descrita no capítulo 5, seção 5.1.3.

Tabela D.1: Frases utilizadas – resolução de pronomes

Primeira Frase	Segunda Frase
Tony is a doctor	He is in the country
Paul is a professor of Jim	He is a scientist
Tony is in Canada	It is a country
Tony is in the area	He is a man
Paul is a lawyer	He is in Canada
Laura is a professor of Mary	She is in the lab
Tony is a director of Paul	He is a doctor
Mary is a lawyer	She likes Tony
Mary is in Portugal	It is a place
Paul is a man	He is in Canada

Apêndice E

Tabelas Extras

A tabela E.1 mostra um resumo das tomadas de palavra por participante. Podemos observar que o primeiro participante realizou com relativa facilidade 4 tarefas, que precisaram de apenas 26 tomadas de palavra (média de 6,5 tomadas por tentativa), entretanto a quinta tarefa consumiu 77% do total de tomadas de palavra (88/114) e não foi realizada com sucesso. Comportamento semelhante pode ser observado no segundo participante, que precisou de 29 tomadas de palavra para realizar 4 tarefas e gastou 55 tomadas (65%) na tentativa de cumprir a quinta tarefa. Apesar de ter realizado todas as tarefas, o participante 5 foi o que precisou de mais tomadas por tentativa, uma média de 25,7 num total de 154.

Tabela E.1: Tomadas de palavra por participante

	Participante	Tarefas	Tentativas	Tomadas	Tomadas/Tentativa
1	Total	5	5	114	22,8
	OK	4	4	26	6,5
	OK(%)	80%	80%	23%	-
2	Total	5	5	84	16,8
	OK	4	4	29	7,3
	OK(%)	80%	80%	35%	-
3	Total	5	5	35	7,0
	OK	1	1	6	6,0
	OK(%)	20%	20%	17% -	-
4	Total	5	11	166	15,1
	OK	5	8	144	18,0
	OK(%)	100%	73%	87%	-
5	Total	6	6	154	25,7

	Participante	Tarefas	Tentativas	Tomadas	Tomadas/Tentativa
	OK	6	6	154	25,7
	OK(%)	100%	100%	100%	-
6	Total	8	8	42	5,3
	OK	7	7	42	6,0
	OK(%)	88%	88%	100%	-
7	Total	8	10	64	6,4
	OK	6	8	36	4,5
	OK(%)	75%	80%	56%	-
8	Total	6	6	87	14,5
	OK	5	5	52	10,4
	OK(%)	83%	83%	60%	-
9	Total	5	5	47	9,4
	OK	4	4	41	10,3
	OK(%)	80%	80%	87%	-
10	Total	5	5	52	10,4
	OK	5	5	52	10,4
	OK(%)	100%	100%	100%	-
Total	média	5,80	6,60	84,50	13,33
	desvio padrão	1,17	2,15	44,15	6,60
OK	média	4,70	5,20	58,20	10,50
	desvio padrão	1,55	2,04	47,20	6,24
OK(%)	média	81%	78%	66%	-
	desvio padrão	22%	21%	31%	-

A tabela E.2 mostra a quantidade de tomadas de palavra por tarefa. Podemos ver que enquanto as tarefas 1 e 3 registraram um elevado número de tomadas por tentativas, 35 e 35,7, respectivamente, as tarefas 5 e 11 marcaram apenas 2 e 4,3.

Tabela E.2: Tomadas de palavra por tarefa

	Tarefa	Tentativas	Tomadas	Tomadas/Tentativa
1	Total	1	35	35,0

	Tarefa	Tentativas	Tomadas	Tomadas/Tentativa
	OK	0 (0%)	0 (0%)	-
2	Total	2	13	6,5
	OK	2 (100%)	13 (100%)	6,5
3	Total	3	107	35,7
	OK	2 (67%)	52 (49%)	26,0
4	Total	2	28	14,0
	OK	1 (50%)	8 (29%)	8,0
5	Total	3	6	2,0
	OK	3 (100%)	6 (100%)	2,0
6	Total	4	103	25,8
	OK	3 (75%)	15 (15%)	5,0
7	Total	8	147	18,4
	OK	5 (63%)	121 (82%)	24,2
8	Total	5	63	12,6
	OK	4 (80%)	53 (84%)	13,3
9	Total	4	31	7,8
	OK	2 (50%)	18 (58%)	9,0
10	Total	3	16	5,3
	OK	2 (67%)	14 (88%)	7,0
11	Total	6	26	4,3
	OK	5 (83%)	26 (100%)	5,2
12	Total	5	43	8,6
	OK	5 (100%)	43 (100%)	8,6
13	Total	7	45	6,4
	OK	7 (100%)	45 (100%)	6,4
14	Total	6	88	14,7
	OK	4 (67%)	74 (84%)	18,5
15	Total	7	94	13,4
	OK	7 (100%)	94 (100%)	13,4
Total	média	4,40	56,33	-
	desvio padrão	2,03	40,49	-

	Tarefa	Tentativas	Tomadas	Tomadas/Tentativa
OK	média	3,47	38,80	-
	desvio padrão	2,00	34,11	-
OK(%)	média	73%	73%	-
	desvio padrão	26%	33%	-

A tabela E.3 mostra os tipos de elocuições por participante. Podemos ver que o participante 3 não precisou de clarificação para as tarefas que conseguiu realizar e que as elocuições de manutenção de diálogo se concentraram no participante 8.

Tabela E.3: Tipo de elocuições utilizadas por participante

Participante	Tarefa	Clarificação	Manutenção de diálogo	Resolução de pronomes	Todas as elocuições
1	Total	39	26	0	66
	OK	8	4	0	13
	OK(%)	21%	15%	-	20%
2	Total	37	13	0	50
	OK	14	4	0	18
	OK(%)	38%	31%	-	36%
3	Total	18	9	0	27
	OK	3	0	0	3
	OK(%)	17%	0%	-	11%
4	Total	69	23	0	92
	OK	60	20	0	80
	OK(%)	87%	87%	-	87%
5	Total	66	30	0	96
	OK	66	30	0	96
	OK(%)	100%	100%	-	100%
6	Total	17	7	1	25
	OK	15	7	1	23
	OK(%)	88%	100%	100%	92%
7	Total	18	14	0	32

Participante	Tarefa	Clarificação	Manutenção de diálogo	Resolução de pronomes	Todas as elocuições
	OK	9	9	0	18
	OK(%)	50%	64%	-	56%
	Total	27	23	4	54
8	OK	15	12	3	30
	OK(%)	56%	52%	75%	56%
	Total	19	10	0	29
9	OK	16	10	0	26
	OK(%)	84%	100%	-	90%
	Total	21	6	0	27
10	OK	21	6	0	27
	OK(%)	100%	100%	-	100%
Total	Média	33,10	16,10	0,50	49,80
	desvio padrão	18,76	8,20	1,20	25,69
OK	Média	22,70	10,20	0,40	33,40
	desvio padrão	20,74	8,38	0,92	28,50
OK(%)	Média	64%	65%	88%	65%
	desvio padrão	30%	37%	13%	32%

A tabela E.4 mostra os tipos de elocuições por tarefa.

Tabela E.4: Tipo de elocuições utilizadas por tarefa

Tarefa	Tarefa	Clarificação	Manutenção de diálogo	Resolução de pronomes	Todas as elocuições
	Total	12	10	1	23
1	OK	0	0	0	0
	OK(%)	0%	0%	0%	0%
	Total	7	2	0	9
2	OK	7	2	0	9
	OK(%)	100%	100%	-	100%
	Total	49	14	0	63

3

	Tarefa	Tarefa	Clarificação	Manutenção de diálogo	Resolução de pronomes	Todas as elocuições
	OK	26	5	0	0	31
	OK(%)	53%	36%	-	-	49%
	Total	8	9	0	0	17
4	OK	3	1	0	0	4
	OK(%)	38%	11%	-	-	24%
	Total	3	0	0	0	3
5	OK	3	0	0	0	3
	OK(%)	100%	-	-	-	100%
	Total	40	23	0	0	63
6	OK	9	1	0	0	10
	OK(%)	23%	4%	-	-	16%
	Total	60	19	0	0	79
7	OK	52	14	0	0	66
	OK(%)	87%	74%	-	-	84%
	Total	27	10	0	0	37
8	OK	23	9	0	0	32
	OK(%)	85%	90%	-	-	86%
	Total	16	3	0	0	19
9	OK	7	2	0	0	9
	OK(%)	44%	67%	-	-	47%
	Total	5	2	0	1	8
10	OK	4	2	0	1	7
	OK(%)	80%	100%	-	100%	88%
	Total	10	6	2	0	18
11	OK	6	6	2	0	14
	OK(%)	60%	100%	100%	-	78%
	Total	12	14	1	0	27
12	OK	12	14	1	0	27
	OK(%)	100%	100%	100%	-	100%
	Total	16	9	0	0	25
13	OK	16	9	0	0	25
	OK(%)	100%	100%	-	-	100%

	Tarefa	Tarefa	Clarificação	Manutenção de diálogo	Resolução de pronomes	Todas as elocuições
14	Total	35	15	0	0	50
	OK	28	13	0	0	41
	OK(%)	80%	87%	-	-	82%
15	Total	31	25	1	0	57
	OK	31	24	1	0	56
	OK(%)	100%	96%	100%	-	98%
Total	Média	22,07	10,73	0,33	0,07	33,20
	desvio padrão	16,85	7,39	0,60	0,25	22,77
OK	Média	15,13	6,80	0,27	0,07	22,27
	desvio padrão	13,83	6,73	0,57	0,25	19,33
OK(%)	Média	70%	64%	20%	7%	70%
	desvio padrão	31%	40%	40%	25%	33%