DANIEL ASSIS ALFENAS

Aplicações da tecnologia adaptativa no gerenciamento de diálogo falado em sistemas computacionais

São Paulo
2014

DANIEL ASSIS ALFENAS

Aplicações da tecnologia adaptativa no gerenciamento de diálogo falado em sistemas computacionais

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Mestre em Engenharia

São Paulo
2014

DANIEL ASSIS ALFENAS

Aplicações da tecnologia adaptativa no gerenciamento de diálogo falado em sistemas computacionais

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Mestre em Engenharia

Área de Concentração: Engenharia de Computação

Orientador: Prof. Dr. João José Neto

São Paulo

2014

Este exemplar foi revisado e corrigido em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, de dezembro de 2014.

Assinatura do autor _____

Assinatura do orientador _____

CATALOGAÇÃO-NA-PUBLICAÇÃO

Alfenas, Daniel Assis
Aplicações da tecnologia adaptativa no gerenciamento de diálogo falado em sistemas computacionais / D.A. Alfenas. -- versão corr. -- São Paulo, 2014.
146 p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1.Tomada de decisão 2.Inteligência artificial 3.Sistemas de diálogo falado 4.Adaptatividade I. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II.t.

Dedico este trabalho à minha família.

AGRADECIMENTOS

Agradeço ao meu orientador, Prof. Dr. João José Neto, pela orientação e pela grande paciência. Agradeço ainda mais pelas excelentes conversas, tanto sobre o mestrado quanto sobre as outras coisas da vida.

Agradeço ao Prof. Dr. Ricardo Luis de Azevedo da Rocha, também do PCS e do Laboratório de Linguagens e Técnicas Adaptativas, por também ter me ajudado com conselhos preciosos tanto agora quanto em outros momentos da minha carreira, pois com eles consegui motivação para começar um novo mestrado após ter falhado em uma primeira tentativa.

Agradeço ao Prof. Dr. Marcos Ribeiro Pereira-Barretto, do PMR, pelos diversos conselhos sobre o meu trabalho e sobre a vida. Foi por causa dele e de seu trabalho que os sistemas de diálogo falado capturaram minha atenção e se tornaram parte do foco da minha pesquisa pelos últimos três anos.

Agradeço aos professores Dr. Leland McCleary, da FFLCH-USP, Dr. Fabio Cozman, do PMR e Dr. Paulo Muniz, do PCS, pelas contribuições ao meu trabalho de pesquisa.

Agradeço à minha esposa, Carine, e à minha filha, Catarine, por entenderem o quanto este mestrado foi importante para mim e por retribuírem minha ausência no lar com amor.

Agradeço também aos meus companheiros de profissão, que entenderam quando precisei me ausentar do trabalho para cuidar da pesquisa e por discutirem comigo muitas das ideias aqui relatadas. Dentre eles, agradeço especialmente ao Danilo Picagli Shibata e ao Daniel Sguillaro.

Agradeço, finalmente, aos meus pais, que me permitiram iniciar a caminhada que me trouxe até aqui, me ensinando a ser persistente e a tratar com honestidade e respeito todo ser humano.

RESUMO

Este trabalho apresenta um estudo sobre como a tecnologia adaptativa pode ser utilizada para aprimorar métodos existentes de gerenciamento de diálogo. O gerenciamento de diálogo é a atividade central em um sistema computacional de diálogo falado, sendo a responsável por decidir as ações comunicativas que devem ser enviadas ao usuário. Para evidenciar pontos que pudessem ser melhorados através do uso da tecnologia adaptativa, faz-se uma revisão literária ampla do gerenciamento do diálogo. Esta revisão também permite elencar critérios existentes e criar outros novos para avaliar gerenciadores de diálogos. Um modelo de gerenciamento adaptativo baseado em máquinas de estados, denominado Adaptalker, é então proposto e utilizado para criar um framework de desenvolvimento de gerenciadores de diálogo, o qual foi exercitado pelo desenvolvimento ilustrativo de uma aplicação simples de venda de pizzas. A análise desse exemplo permite observar como a adaptatividade é utilizada para aperfeiçoar o modelo, tornando-o capaz, por exemplo, de lidar de forma mais eficiente tanto com o reparo do diálogo quanto com a iniciativa do usuário. As regras de gerenciamento do Adaptalker são organizadas em submáquinas, que trabalham de forma concorrente para decidir qual a próxima ação comunicativa.

Palavras-Chave: Adaptatividade; Sistemas de Diálogo Falado; Inteligência Artificial; Tomada de Decisão.

ABSTRACT

This work presents a study on how to apply adaptive technologies to improve existing dialog management methodologies. Dialog management is the central activity of a spoken dialog system, being responsible for choosing the communicative actions sent to the system user. In order to evidence parts that can be improved with adaptive technology, a large review on dialog management is presented. This review allows us to point existing criteria and create new ones to evaluate dialog managers. An adaptive management model based on finite state-based spoken dialog systems, Adaptalker, is proposed and used to build a development framework of dialog managers, which is illustrated by creating a pizza selling application. Analysis of this example allows us to observe how to use adaptivity to improve the model, allowing it to handle both dialog repair and user initiative more efficiently. Adaptalker groups its management rules in submachines that work concurrently to choose the next communication action.

Keywords: Adaptivity; Spoken Dialog Systems; Artificial Intelligence; Decision-making.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURA E SIGLAS

ASR	*Automatic speech recognizer* ou reconhecimento automático de voz
DATD	Dispositivo adaptativo de tomada de decisão
ECA	*Embedded conversational agentes* ou agente conversacional embutido
Ex.	Exemplo
GD	Gerenciador de diálogo
HIS	*Hidden Information State* ou modelo do estado da informação oculta
LN	Linguagem natural
MDP	*Markov decision process* ou processo de decisão de Markov
POMDP	*Partially observable MDP* ou MDP parcialmente observáel
SDS-POMDP	POMDP para sistemas de diálogo
SLU	Spoken language understanding ou entendimento de linguagem falada
SVM	*Support vector machines* ou máquinas de vetores de suporte
TRP	*Transitional-relevance place* ou ponto de relevância para transição
XML	*Extensible Markup Language*

LISTA DE SÍMBOLOS

g	Objetivo corrente do usuário
a_u	Última ação do usuário
H	Conjunto de hipóteses sobre a_u
o	Observação sobre a ação do usuário, $o \in O$
c	Grau de confiança
s_d	Histórico do diálogo
b_s	Estado de crença
a_s	Próxima ação do sistema. $a_s \in A_s$
Q	Conjunto de estados
A_s	Conjunto de todas as ações que o sistema pode realizar
T	Conjunto de probabilidades de transição entre estados
P	Probabilidade
s	Estado de um MDP ou POMDP, $s \in Q$
R	Conjunto de expectativas de recompensa imediata
r	Cada função de expectativa, $r \in R$, definida em termos de s_d, a_u, g,,a_s
O	Conjunto de observações possíveis sobre a ação do usuário
Z	Distribuição de probabilidades de observação $P(o\|s',a_s)$
λ	Fator de desconto geométrico para cálculo da expectativa de recompensa em longo prazo para MDP ou POMDP. Ou, em dispositivos adaptativos, qualquer um dos parâmetros formais de uma função adaptativa
b_0	Estado de crença inicial de um POMDP
k	Constante de normalização
$\pi(b)$	Política de escolhas de um MDP ou POMDP
$\pi^*(b)$	Política ótima de escolhas
K	Cadeia de markov; ou transição pendente de resposta do usuário em μ
q_0	Estado inicial
n	Tamanho de um conjunto, por exemplo, para uma cadeia de Markov é o tamanho do conjunto de estados
γ_{ij}	Transição ligando um estado q_i a um estado q_j

ρ_{ij}	Probabilidade do estado q_j ser atingido estando em q_i
Γ_q	Conjunto de todas as transições que se originam em um estado q
Ψ	Alfabeto de saída de um dispositivo
M	Máquina de Markov adaptativa; ou conjunto de todas as submáquinas de um DATD
ε	Cadeia vazia
p_{ij}	Símbolo inserido na cadeia de saída de M quando γ_{ij} é acionada. $p_{ij} \in \Psi \cup \{\varepsilon\}$
F	Conjunto de funções adaptativas
f	Função adaptativa, $f \in F$
Λ	Sequência de parâmetros formais de uma função adaptativa
m	Tamanho de Λ ou quantidade de segmentos em o_i^t
V	Conjunto de identificadores de variáveis de uma função adaptativa
v	Identificador de variável de uma função adaptativa
G	Conjunto de identificadores de geradores de uma função adaptativa
g^*	Identificador de gerador de uma função adaptativa
C	Sequência de ações adaptativas elementares executadas em uma função adaptativa
a_{ij}	Ação adaptativa executada em γ_{ij}
ω	Argumento de uma ação adaptativa
Σ	Alfabeto de saída
κ	Identificador de uma máquina de Markov
a_s^t	t-ésima ação do sistema
Δ	Quantidade total de ações do sistema geradas desde o início do diálogo
a_u^t	t-ésima ação do usuário
U	Quantidade total de ações do usuário recebidas pelo gerenciador desde o início de diálogo
o_i^t	Cada uma das n observações (hipóteses) paralelas sobre a_u^t
c_i^t	Grau de confiança em uma hipótese o_i^t
$\theta_{i,j}^t$	Cada um dos segmentos funcionais de uma observação o_i^t
$\varphi_{i,j,k}^t$	Cada um dos atos dialogais de $\theta_{i,j}^t$

p	Quantidade de atos dialogais em $\theta_{i,j}^{t}$ ou em a_s^t
Ψ_s	Conjunto de todos os atos dialogais que podem ser gerados pelo sistema
φ_{sk}^{t}	Qualquer um dos atos dialogais de a_s^t
$e_{k,l}^{t}$	Qualquer uma das expectativas sobre a próxima ação do usuário associadas a φ_{sk}^{t}
$c_{k,l}^{t}$	Grau de confiança do sistema em $e_{k,l}^{t}$
c_{min}	Grau de confiança mínimo aceitável
c_{mindif}	Diferença mínima exigida entre os graus de confiança das duas hipóteses mais prováveis sobre a ação do usuário
M_a	Lista ordenada de submáquinas ativas, $M_a \in M$
M_0	Configuração inicial de M_a
μ	Submáquina de um DATD. $\mu \in M$
B	Conjunto de estados de *binding*
X	Conjunto de variáveis que podem ser utilizadas em μ para armazenar resultados de consultas entre as transições
Q_f	Conjunto de estados de aceitação de μ
π	Condição para que uma transição de μ seja disparada
x	Identificadores de variáveis de μ, $x \in X$
c_π	Expectativa que o gerenciador de diálogos tem de que a condição π será satisfeita
α_a	Ação adaptativa executada antes da transição de estado
α_p	Ação adaptativa executada após a transição de estado
β	Tarefa que será executada ao final da transição
foco	Diz quem receberá o foco de execução das regras após a execução da tarefa β
A_f	Sequência de ações adaptativas elementares executadas em f
β_c	Ação elementar de consulta
ζ	Lista de atos dialogais de entrada de um DATD
q_a	Estado correntemente ativo de μ
Φ	Lista dos p atos dialogais φ_{sk}^{t}de saída de μ
E	Lista das expectativas $e_{k,l}^{t}$ de μ sobre a próxima ação do usuário

ζ_μ	Lista de entradas da submáquina. $\zeta_\mu \in \zeta$
floor	Variável que guarda o valor do foco requisitado pela última regra de μ
i_ζ	Distância entre as expectativas correntes da submáquina e a regra que irá processar o ato dialogal
$\gamma_{i,\pi}$	Transição de μ iniciando em um estado q_i com condição π

SUMÁRIO

1 INTRODUÇÃO

Robôs sociáveis, avatares virtuais (ECA, *embedded conversational agents*), aplicativos de controle de voz e serviços de atendimento telefônico podem ser classificados em um grupo maior de sistemas computacionais, os sistemas de diálogo falado. Esses sistemas são formados por componentes de reconhecimento de voz, entendimento de linguagem natural, gerenciamento do diálogo, geração de linguagem natural e síntese de voz. O gerenciador do diálogo é o módulo responsável pela tomada de decisões, controlando o fluxo do diálogo. Ele pode ser visto como o coração do sistema. Existem várias técnicas para modelagem do gerenciamento do diálogo, cada uma com vantagens e desvantagens. Este trabalho propõe a utilização de tecnologia adaptativa para aperfeiçoar estas técnicas.

1.1 Objetivo

O trabalho apresentado neste documento tem como objetivo principal aperfeiçoar técnicas existentes de gerenciamento do diálogo, aplicadas em produtos comerciais ou propostas em trabalhos de pesquisa, através da incorporação de mecanismos adaptativos. A adaptatividade é um termo que se refere à capacidade de um sistema de, sem a interferência de qualquer agente externo, tomar a decisão de modificar seu próprio comportamento, em resposta ao seu histórico de operação e aos dados de entrada (JOSÉ NETO, 2007). A tecnologia adaptativa já foi utilizada com sucesso em diversos tipos de aplicação, como síntese de voz (SHIBATA, 2008), tomada de decisão (PISTORI; JOSÉ NETO; PEREIRA, 2006; TCHEMRA, 2009), geração de música (ALFENAS et al., 2012; BASSETO, 2000) e navegação de robôs (HIRAKAWA; SARAIVA; CUGNASCA, 2012), entre muitos outros.

Este trabalho apresenta, portanto, ao menos uma técnica para o gerenciamento de diálogo utilizando adaptatividade. Mais especificamente, estre trabalho propõe que as técnicas de gerenciamento aqui descritas sejam aplicadas no robô sociável Minerva, desenvolvido pelo Laboratório de Robôs Sociáveis do Departamento de Engenharia Mecatrônica da Escola Politécnica da USP. As propostas têm que ser, portanto, adequadas às capacidades do robô. Por exemplo, a capacidade comunicacional dela vai além da fala, sendo capaz, por exemplo, de detectar e manifestar expressões faciais, mas não de gesticular (ALFENAS; PEREIRA-BARRETTO; PAIXÃO, 2013).

Este trabalho se propõe, também, a fazer uma revisão bibliográfica extensa do estudo teórico e prático da conversação e das técnicas utilizadas no gerenciamento de diálogo, correlacionando as características desejadas de um diálogo e as propostas existentes de gerenciamento, para ressaltar os pontos que precisam de aperfeiçoamento.

1.2 Justificativa

Existem vários motivos para a utilização de diálogo falado em linguagem natural em sistemas computacionais, ao invés de interfaces tradicionais utilizando mouse, teclado ou telas sensíveis ao toque. Edlund et al. (2008) lista uma série de situações em que usar diálogo falado é melhor do que usar outros tipos de interface:

- Situações em que as mãos e olhos precisam estar livres para realizar outras tarefas, como na direção de um automóvel;
- Quando as outras interfaces são inconvenientes, como, por exemplo, em um passeio no museu;
- Quando deficiências tornam as outras interfaces inúteis como, por exemplo, deficiência visual;
- Situações em que o hardware utilizado restringe as modalidades de comunicação que podem ser empregadas, como o telefone.

Edlund lista também os benefícios trazidos pelos sistemas de diálogo falado mesmo em situações já suportadas usualmente por outras interfaces, como facilidade de uso (mais seres humanos sabem falar do que usar um computador, sendo necessário menos ou mesmo nenhum treinamento), flexibilidade, tratamento de erros bastante robusto, sociabilidade e divertimento.

Aproveitando-se dessas características, a utilização de robôs sociáveis tem sido proposta em áreas variadas. Podem-se citar os trabalhos promissores de robôs sociáveis com crianças (DÍAZ et al., 2011), como treinamento, reabilitação, terapia para autistas e entretenimento, e os trabalhos como guias em centros comerciais e museus, dos quais podemos citar o robô homônimo Minerva (THRUN et al., 1999). Uma pesquisa aponta, inclusive, que a satisfação dos usuários com robôs-guia é maior se estes conversam com os mesmos, principalmente quando os participantes se movimentam em baixa velocidade (ZHENG et al., 2013).

Bastante motivador foi também a existência de um projeto de robô sociável na própria USP. Este projeto já suscitou diversos trabalhos de mestrado e de graduação, principalmente na detecção de emoções (CUEVA, 2012; GONÇALVES, 2013), assim como a publicação de diversos trabalhos em congressos. Através de um trabalho de graduação já foi possível incorporar capacidade de diálogo na Minerva, mesmo que bastante reduzida, utilizando-se a mesma tecnologia utilizada na ALICE (WALLACE, 2003), um agente virtual de bate-papo.

A motivação principal do uso da adaptatividade foi determinada pela sua capacidade de transformar dispositivos livres de contexto em dispositivos sensíveis ao contexto com poucas modificações estruturais. O diálogo tem características sensíveis ao contexto, mas algumas das técnicas existentes, mesmo que viáveis em aplicações específicas, são capazes apenas de modelar diálogos livres de contexto, retirando grande parte da flexibilidade do diálogo.

1.3 Estrutura do texto

Os demais capítulos deste trabalho de dissertação podem ser agrupados em três seções. A primeira, composta pelos capítulos 2, 3 e 4, contém a revisão da literatura relacionada. A segunda, composta pelos capítulos 5, 6, 7 e 8, apresenta a técnica de gerenciamento adaptativo proposta, com todas as contribuições e resultados alcançados através deste trabalho. A terceira seção discute os resultados e apresenta as conclusões.

O Capítulo 2 apresenta a parte teórica da conversação conforme os trabalhos em áreas correlacionadas, como ciências sociais e linguística, e faz a correlação com os trabalhos na área de computação, mostrando como os aspectos funcionais do diálogo são adotados nos sistemas computacionais.

O Capítulo 3 faz a revisão dos sistemas computacionais de diálogo falado, cobrindo aspectos de arquitetura tanto do sistema quando do gerenciamento de diálogo. Dois modelos de gerenciamento de diálogo são revisados com detalhe. Este capítulo também revisa aspectos importantes da metodologia de desenvolvimento, incluindo critérios de avaliação comumente utilizados em sistemas de diálogo falado.

O Capítulo 4 faz uma revisão breve sobre a adaptatividade. Nele está incluído o detalhamento completo de um dispositivo, o Sistema de Markov Adaptativo

(ALFENAS et al., 2012). Além deste, outros dois dispositivos são descritos de forma resumida.

O Capítulo 5 revisa o sistema em que o gerenciador proposto deve ser utilizado, incluindo análise das características da Minerva, arquitetura de comunicação entre os componentes e especificações de entradas e saídas do gerenciador de diálogo.

O Capítulo 6 descreve a notação e os critérios utilizados para seleção e refinamento do método de gerenciamento de diálogo proposto, pois os critérios de avaliação do gerenciamento do diálogo encontrados na literatura não foram suficientes para orientar o trabalho de pesquisa. Uma lista abrangente de cenários de diálogo é enumerada, sempre do ponto de vista do gerenciamento de diálogo, de forma que todas as situações possíveis em um diálogo possam ser classificadas conforme estes cenários.

O Capítulo 7 descreve o modelo proposto para o gerenciamento adaptativo do diálogo, o Adaptalker. O capítulo contém uma descrição das camadas do modelo, incluindo a formalização do dispositivo adaptativo utilizado no Adaptalker.

O Capítulo 8 lista os resultados alcançados, descrevendo em alto nível um gerenciador completo desenvolvido para venda de pizzas. Uma das partes do gerenciador, responsável pelo cadastro do usuário, é descrita de forma mais detalhada, para que se possa verificar como os critérios levantados são atendidos pelo modelo proposto.

O Capítulo 9 discute os resultados obtidos, comparando com outras técnicas de gerenciamento não adaptativas.

O Capítulo 10 finaliza este texto com as conclusões sobre o trabalho de pesquisa realizado, incluindo contribuições e trabalhos futuros.

2 REVISÃO DA LITERATURA: O ESTUDO DA CONVERSAÇÃO

O diálogo é uma das mais naturais e básicas formas de comunicação. Crianças aprendem a conversar antes de ler ou escrever, o que contrasta com o fato de que apenas recentemente a humanidade passou a estudar a maneira como o diálogo é entendido, produzido e mantido. Embora existam, atualmente, várias áreas que estudam direta ou indiretamente o diálogo, como a sociologia, a linguística, a comunicação, a psicologia, a biologia e mesmo a computação, o estudo do funcionamento da conversação e de outras formas de interação faladas começou apenas no final da década de 60 com os estudos sobre análise da conversação de Harvey Sacks (SCHEGLOFF; SACKS, 1973). Antes disso, a conversa era um assunto desprestigiado nas teorias discursivas (KERBRAT-ORECCHIONI, 1996).

A análise da conversação se baseia na etnometodologia, que é uma corrente das ciências sociais cuja pesquisa é focada nos métodos que as pessoas utilizam no seu dia-a-dia para construir suas relações sociais e que se tornou conhecida a partir da publicação de *Studies in ethnomethodology* (GARFINKEL, 1967). A etnometodologia passa a estudar a interação entre indivíduos ao invés de padrões sociais, passando a prestar atenção naquilo que antes era “visto, mas despercebido”, isto é, algo implícito. A análise da conversação incorpora estes princípios ao partir da análise de interações conversacionais específicas.

A análise da conversação é a responsável pela teorização de vários conceitos importantes do diálogo, como o sistema de tomada de turnos, mas não é a única corrente que contribui para o seu estudo. Nos tópicos a seguir, discute-se a literatura referente às características do diálogo, trabalhos teóricos e práticos relacionados nas áreas relevantes a este trabalho, e faz-se um paralelo na computação, com referências e definições utilizadas na pesquisa relacionada a robôs sociáveis e a sistemas de diálogo falados.

2.1 Organização interacional das conversas: o sistema de tomada de turnos

O sistema de tomada de turnos introduzido em Sacks et al. (1974) tornou-se o modelo padrão para análises de conversação. Embora o modelo original de Sacks, Schegloff e Jefferson considere apenas a voz (foi baseado em gravações de conversas telefônicas), sua validade tem se mantido mesmo quando se considera a

multimodalidade da comunicação, feitas as adaptações adequadas (LEITE, 2008). Todos os sistemas de diálogo falado verificados por este trabalho são baseados neste sistema.

Em um breve resumo, tal sistema relata que, em uma conversa:

- A pessoa que fala muda ao menos uma vez;
- As pessoas falam uma de cada vez;
- Sobreposições de fala acontecem, mas tendem a ser curtas;
- Há pouca latência entre os turnos;
- O tamanho da conversa, a ordem e tamanho dos turnos e o que as pessoas falam pode variar e não pode ser definido a priori;
- Técnicas de alocação de turno podem ser utilizadas. Por exemplo, o falante atual pode selecionar quem será o próximo a falar;
- Os turnos são compostos por unidades menores, chamadas de unidades de construção de turno. Estas unidades podem ser de vários tipos, como sílabas, palavras e frases. Eventualmente, é possível que o ouvinte perceba o complemento de uma unidade de construção antes que o interlocutor a fale. Quando isto acontece, diz-se que o ouvinte *projetou* a fala do interlocutor;
- Mecanismos de reparo existem para lidar com erros na tomada de turnos.

Quando comparado com outros sistemas falados de interação, como debates e tribunais de julgamento, notam-se algumas diferenças. Em debates televisionados, por exemplo, o costume é ter a ordem dos turnos pre-especificada e a duração dos turnos limitada.

As sobreposições, embora em geral não desejadas, são naturais e tendem a acontecer nos assim chamados *pontos de relevância para transição*, ou TRP (*transitional-relevance place*), que é o nome dado ao momento em que um ouvinte tem a oportunidade de iniciar um turno. Um possível ponto em que o ouvinte consegue projetar o que está sendo dito e tem a oportunidade de completar constitui um ponto de relevância para transição. É possível detectar estes pontos a partir da análise de diversos aspectos da conversa, como gestos, prosódia, léxico, sintaxe, etc. A esta capacidade de projetar uma intervenção até seu final a partir de certo ponto durante a própria intervenção dá-se o nome de *projetabilidade*, tanto por parte do falante quanto do receptor. Aquele que projeta já é capaz de planejar uma reação como, por exemplo, uma correção. Reflexos deste planejamento já podem ser observados antes mesmo

de sua execução falada, como através dos gestos e expressões faciais (SCHEGLOFF, 1984).

2.2 Diversidade dos participantes e confidencialidade

Pessoas têm seus próprios objetivos e estilos para comunicar valores, experiências e normas dos grupos aos quais pertencem (WOOD, 2010). Fisiologia, localidade, etnia, classe social e idade são somente alguns dos fatores que influenciam a maneira de um indivíduo se comunicar. Não é fácil para um sistema computacional lidar com tal diversidade. Ela influencia desde o reconhecimento automático de voz (AUBANEL; NGUYEN, 2010) até as expectativas sociais geradas por um ato comunicativo. Influencia também o próprio significado das palavras, que pode mudar até mesmo no transcorrer de uma única conversa (LIBERMAN, 2012) ou transformar-se em um código com significado único para um grupo específico de pessoas.

Em inteligência artificial, chama-se de *senso comum* o grupo de informações que o agente conversacional supõe que toda pessoa conheça, ao menos no público alvo, e que não possui qualquer problema de confidencialidade. Entretanto, pode ser desejável por parte dos participantes que se trate uma parte das informações trocadas como secreta. Portanto, um modelo computacional para agentes conversacionais deve considerar o grau de confidencialidade de uma informação antes de executar uma intervenção a ela relacionada.

Intui-se que, ao se desenhar um sistema de diálogo falado, é importante delimitar não só a funcionalidade, mas, mais do que em outros sistemas, o quanto ele é específico de um local ou cultura ou o quanto ele é interpessoal.

2.3 Multimodalidade

Um projeto que pretenda construir um robô capaz de se comunicar na forma mais similar possível à de um ser humano deve considerar um modelo multimodal de comunicação que inclua gestos, movimento da cabeça, expressões faciais, orientação corporal e direcionamento dos olhos (SCHEGLOFF, 1991), além da voz em suas partes verbais e prosódicas. O trabalho de (GOODWIN, 2000), relacionando análise da conversa e teoria da ação, foi, talvez, o marco inicial em prol da ideia de que a análise da ação humana deveria considerar o uso simultâneo de múltiplos recursos

semióticos pelos participantes de uma interação conversacional. Goodwin argumentou contra a ideia, bastante comum até então, de que tal análise deveria tratar a comunicação verbal falada como primária e o resto como contexto. Ele introduz o conceito de *campo semiótico*, que pode ser entendido como uma fonte distinta de sinais que colabora simultaneamente com outros campos para que as ações humanas possam ser compreendidas; podem-se citar diversos exemplos de campos semióticos, tais como a parte verbal da fala de uma pessoa, a prosódia de tal fala e até mesmo a posição dos participantes de um jogo de futebol em relação à quadra em que a partida é jogada. Um conjunto particular de campos semióticos aos quais os participantes de uma ação se orientam em um determinado momento é chamado de *configuração contextual*. Lógicas similares de análise têm levado, na última década, a análises da conversa cada vez mais complexas e sofisticadas, como, por exemplo, no trabalho de (MCCLEARY; LEITE, 2012).

Nos trabalhos pesquisados relacionados à modelagem computacional de processamento de interação multimodal entre homem e máquina, é comum a utilização das expressões *modalidade de entrada* e *modalidade de saída* para se referir aos campos semióticos; *fusão*, *cooperação* ou *integração* para se referir ao processo de integração dos vários sinais de entradas, sendo o primeiro termo o mais adotado (LALANNE et al., 2009); e *fissão* para se referir ao processo de gerar uma mensagem adequada para o usuário utilizando-se várias saídas multimodais (DUMAS; LALANNE; OVIATT, 2009). Os campos semióticos considerados em um trabalho de um robô sociável são restringidos tanto pela capacidade técnica e esforço necessário quanto pela própria falta de conhecimento de como o ser humano interpreta e sincroniza esses campos. Entretanto, tais dificuldades não têm limitado a pesquisa computacional na área, tanto para gestos quanto para expressões faciais, inclusive no Brasil (MADEO, 2013; ROMERO; SANTOS, 2012).

2.4 Função comunicativa e obrigações sociais

Tomasello (2008) descreve três tipos básicos de funções comunicativas: solicitar, informar e compartilhar. Elas podem ser utilizadas sem depender de um canal verbal. Destes três, podem-se derivar vários outros subtipos, como reforçar, negar, agradecer, etc. Quando se consideram a linguagem e as obrigações sociais do ser humano moderno, esta lista de funções comunicativas básicas aumenta. Uma das

mais discutidas estruturas da conversação, os pares adjacentes (SCHEGLOFF; SACKS, 1973) e suas diversas expansões constituem uma série de tipos básicos de funções comunicativas que precisam ser atendidas por agentes conversacionais e que são definidas tanto a partir de necessidades comunicativas como por convenções sociais. Como exemplos de pares adjacentes, pode-se citar pergunta e resposta, elogio e agradecimento e solicitação de saudação e resposta de saudação.

Existem diversas propostas para padronizar as anotações de diálogo, algumas delas recentes, como o ISO 24617-2 (BUNT et al., 2012) e o DIT++ (BUNT, 2009). O padrão ISO mencionado é baseado em *atos dialogais*, operações de atualização de estado de informação em cada um dos participantes do diálogo. Um mesmo *segmento funcional* (que pode ser entendido como a menor unidade de construção do turno que possua uma ou mais funções comunicativas) pode ser composto de vários atos dialogais, atualizando o estado das informações em diferentes *dimensões*, isto é, informações de diferentes tipos (BUNT, 2006). A Figura 1 ilustra essa situação.

Figura 1 – Exemplo de multidimensionalidade em um mesmo segmento funcional.

1. U: Você pode me dizer a que horas sai o primeiro trem para o aeroporto no domingo de manhã?
2. S: O horário do primeiro trem para o aeroporto no domingo de manhã é 5:32.
3. U: Obrigado.

Adaptado de (BUNT, 2006)

Como se pode observar, o segmento funcional equivalente ao turno 2 tem os seguintes atos dialogais: responde a questão levantada no turno 1 e dá *feedback* positivo sobre o entendimento da questão ao se repetir a que se refere a resposta. Note que o segmento funcional do turno 3 também atualiza informações em dimensões distintas, pois agradece e dá *feedback* de que o entendimento de S sobre 1 foi considerado satisfatório.

Desta forma, o padrão ISO define que todo ato dialogal é composto de uma dimensão e de uma função comunicativa. O padrão define também um modelo para qualificadores funcionais, relações retóricas e relações de dependência funcional e de *feedback*. *Qualificadores funcionais* são informações adicionais sobre o segmento

funcional, como, por exemplo, a emoção detectada e o grau de confiança do ouvinte de que o seu entendimento é correto. Relações de dependência funcional são aquelas que existem entre pares adjacentes como, por exemplo, entre pergunta e resposta. Relações de *feedback* mostram a que segmento um determinado *feedback* se refere. Por fim, *relações retóricas* são utilizadas para explicar as relações de sentido entre diferentes trechos do diálogo, como justificativas, exemplificações e contraposições.

Além das expectativas geradas pelos pares adjacentes de atos dialogais, existem também *expectativas sociais*. Quando um interlocutor pergunta o nome de alguém, espera-se não apenas que o ouvinte responda, mas que também pergunte o nome do interlocutor, se ainda não souber. Em datas especiais, como o natal em algumas comunidades cristãs, é esperado que se troquem votos de felicidade após ou conjuntamente com os pares adjacentes de saudação. Portanto, as expectativas geradas nos ouvintes não são derivadas apenas das funções comunicacionais, mas também da semântica e do contexto social.

2.5 Estrutura da conversa

Além dos pares adjacentes, existem outras estruturas da conversa relevantes para o diálogo, embora possam se notar diferenças entre os trabalhos de diferentes autores da análise da conversação. Em seu trabalho, Kerbrat-Orecchioni (2006) descreve um método de estruturação do diálogo utilizando sequência de trocas de fala e divide cada intervenção em *atos de fala*, deixando a representação de gestos em um patamar de menor importância. Calbris (2011) utiliza como estrutura básica do diálogo sequências de *unidades ideacionais*, que são fragmentos da conversa definidos em termos tanto da fala quanto de aspectos não verbais como postura corporal e gestos, e que claramente se opõe ao termo ato de fala utilizado pelo outro método. Nesta última linha, que é mais complexa e abrangente, um sistema computacional de diálogo deveria suportar um grande sincronismo entre as detecções visual, de movimentos e de voz, o que é condizente com uma proposta multimodal de arquitetura. Ao se projetar um sistema de diálogo falado, é necessário considerar a relação entre o método de análise utilizado e a representação computacional destas unidades (como os atos dialogais do ISO 24617-2).

Apesar das diferenças entre os métodos dos autores, é possível destacar algumas estruturas hierárquicas em comum:

- Interação: encontro, conversação, etc. Algumas vezes é difícil delimitar o início e o fim de uma interação, e como elas estão ligadas entre si. Por exemplo, uma situação de difícil delimitação acontece quanto os indivíduos A, B e C estão conversando e C se despede e deixa a conversa, mas volta depois de um tempo retomando a conversa anterior com A e B.
- Sequência: bloco de trocas conversacionais ligadas por um forte grau de coerência semântica ou pragmática, isto é, que trata de um mesmo assunto ou tarefa. Em uma interação, o início é marcado por uma sequência de abertura, com cumprimentos, e o fim por uma sequência de encerramento, com despedidas.
- Troca: a menor unidade composta por mais que um turno. Os pares adjacentes são casos típicos de trocas. Elas podem não ser verbais: um aperto de mão silencioso pode substituir a saudação inicial entre humanos.
- Ação ou intervenção: pode ser um turno de fala ou um gesto
- A menor unidade conversacional é equivalente à unidade ideacional. Ela pode substituir tanto um trecho de fala quanto um gesto ou movimento de cabeça. Normalmente equivale a um segmento funcional da ISO 24617-2.

2.6 Common Ground e Feedback[1]

Common ground (BAKER et al., 1999; TOMASELLO, 2008) é o nome dado ao conjunto de informações que um participante supõe que seu interlocutor saiba (inclusive aquelas que o interlocutor sabe sobre o próprio participante), e que são básicas para o bom relacionamento entre as pessoas no dia-a-dia. *Grounding* é a nome dado ao processo interativo pelo qual o *common ground* entre dois indivíduos é construído e mantido (CLARK; BRENNAN, 1991). Algum entendimento comum sempre irá existir ao início da conversa (quer seja fruto do senso comum ou de interações anteriores), mas aumentará com as novas informações decorrentes da atividade em andamento. Para um sistema computacional de diálogo falado, isto

[1] Optou-se por não traduzir estes termos. Nenhum outro trabalho pesquisado em português do Brasil traduz *common ground, feedback* e *grounding*. Os trabalhos de (GARCEZ; LODER, 2005) e a edição brasileira de (KNAPP, 2006) são exemplos.

reflete no fato de que perguntas já feitas e informações já dadas na interação corrente ou em anteriores não devem ser repetidas, a não ser que uma evidência negativa seja percebida, isto é, evidência de que algo comunicado, que se acreditava já consolidado, não foi lembrado ou entendido corretamente. Até o idioma utilizado faz parte do *common ground*, incluindo expressões ou palavras inventadas.

O processo de *grounding* é colaborativo, e requer certo esforço tanto do falante quanto do ouvinte para que ele seja realizado, através, principalmente, de *feedbacks* e reparos. Entre seres humanos o *feedback*, positivo ou negativo, é construído em tempo real, enquanto a mensagem ainda está sendo pronunciada (TRAUM et al., 2012) e pode continuar após o fim da mesma. Movimentos de cabeça, dos olhos, gestos, expressões faciais e voz (tanto de forma verbal quanto não verbal) evidenciam se o ouvinte está entendendo ou não o que está sendo dito e se ele concorda ou não. Entretanto, nem todo aspecto da interação precisa ser completamente *grounded*; apenas o que os participantes acreditam ser o suficiente para que se possa prosseguir rumo ao que é o propósito atual da conversa. Os participantes tendem a minimizar o esforço dispensado neste processo; essa regra é conhecida como *least collaborative effort* (menor esforço colaborativo).

Podem acontecer vários problemas para construir o *common ground.* Podemos identificar quatro níveis em que acontecem (BAKER et al., 1999):

- Contato: desejo e capacidade do participante de continuar a interação;
- Percepção: desejo e capacidade do participante de perceber a mensagem;
- Entendimento: desejo e capacidade do participante de entender a mensagem;
- Reação atitudinal: desejo e capacidade do participante de reagir adequadamente à mensagem.

Cada nível depende do anterior, isto é, não é possível ter a atitude correta se não houve entendimento, não é possível entender se não foi percebido, e não se pode perceber se não há contato. O *feedback* é uma evidência de percepção e, possivelmente, de entendimento. Problemas nos níveis de entendimento e de reação podem levar a uma ação de reparo de um dos participantes. Sob esta perspectiva, o *feeback* pode ser entendido como um mecanismo para evitar erros de *grounding*.

Nos sistemas de diálogo falado, existem algumas estratégias típicas para se obter e fornecer *feedback*. Se o grau de confiança sobre a última ação do usuário é baixo, ele pode solicitar a confirmação ou repetição do que foi dito antes de considerar o

conhecimento como *grounded*. Os tipos de estratégia mais comumente utilizados estão na Figura 2.

Figura 2 – Tipos mais comuns de estratégia para *feedback* em sistemas de diálogo falado**.**

Tipo de Estratégia	**Descrição**
Confirmação explícita	O sistema repete o turno anterior do usuário e pede para que confirme se o entendimento foi correto. Exemplo: Usuário: Gostaria de ir a um bar no centro de São Paulo. Sistema: Você gostaria de ir a um bar no centro de São Paulo?
Confirmação implícita	O sistema repete o turno anterior do usuário ao pedir algum dado adicional. Exemplo: Usuário: Gostaria de ir a um restaurante chinês. Sistema: Você gostaria de ir a um restaurante chinês em que local?
Reformulação estática	Utilizado quando o sistema não consegue repetir o que o usuário disse. Solicita ao usuário para reformular ou repetir a frase. Exemplo: Usuário: Gostaria de ir a um restaurante chinês. Sistema: Desculpe, você poderia reformular a frase?
Reformulação dinâmica	Também utilizado quando o sistema não consegue repetir o que o usuário disse. Solicita ao usuário para reformular a frase em um formato específico. Exemplo: Usuário: Gostaria de comer em um restaurante chinês. Sistema: Desculpe. Preciso que você reformule a frase usando a expressão: “Eu gostaria de ir a [TIPO_RESTAURANTE] em [LOCAL_RESTAURANTE]”.

Fonte**:** (CHEONGJAE LEE et al., 2010)

Entretanto, todas estas estratégias típicas, se usadas exclusivamente, podem tornar a utilização insatisfatória e não natural. Em sistemas de processamento incremental (SKANTZE, 2008) ou que utilizem mecanismos multimodais, é possível utilizar mecanismos mais naturais para obter e prover *feedback*. Neste último caso, podemos citar o trabalho de Kopp (2010), que discute como modelar um agente conversacional capaz de criar uma conexão harmoniosa com seu interlocutor através da coordenação de comportamentos, crenças e atitudes, utilizando não só a voz, mas também gestos, expressões faciais e postura corporal.

O *feedback* possui sua própria dimensão de funções comunicativas tanto para o sistema de análise da conversa do DIT++ quanto para o do ISO-24617-2, que permite classificá-lo como positivo ou negativo e se é referente a si próprio ou ao interlocutor.

Entretanto, utilizar todos esses mecanismos não evita que problemas de entendimento continuem a ocorrer naturalmente; neste caso, o sistema precisa estar preparado para reparar o problema.

2.7 O reparo

Reparo é o mecanismo utilizado na conversação para que um participante possa corrigir ou esclarecer uma contribuição anterior. Ressalta-se que o reparo não é sempre uma correção nas conversas entre humanos; por exemplo, ele pode acontecer quando o falante se lembra de uma palavra esquecida em turno anterior (SCHEGLOFF; JEFFERSON; SACKS, 1977).

Na sua forma mais geral, o reparo tem duas fases: a iniciação e o resultado[2]. A iniciação envolve a indicação de que há um problema de entendimento e pode ser feita tanto pelo próprio autor (*reparo auto-iniciado*) quanto pelo ouvinte (*reparo iniciado pelo outro*) do trecho problemático. O resultado do reparo também pode ser feito tanto pelo autor (*auto-reparo*) quanto pelo ouvinte (*reparo levado a cabo pelo outro* ou, de forma simplificada, *reparo pelo outro*).

É importante ressaltar que existem outras maneiras utilizadas pelos seres humanos para lidar com problemas de entendimento que não o reparo. Por exemplo, após o autor percebê-lo, ele pode desistir da estratégia anterior e optar por outra. Em outro exemplo, o autor do trecho problemático pode agir como se o ouvinte tivesse entendido corretamente e prosseguir a conversa (SCHEGLOFF, 1992).

O reparo pode ser classificado quanto à distância, em turnos, entre o trecho problemático e a ação de reparo:

- Reparo no mesmo turno do trecho problemático.
- Reparo na segunda posição: iniciado pelo outro no turno seguinte à fonte do problema;
- Reparo na terceira posição: Auto-iniciado no terceiro turno em relação à fonte do problema. Acontece tipicamente quando o outro responde como se tivesse entendido, mas então o autor percebe, pela resposta, que foi mal entendido.

---

[2] Neste trabalho optamos por utilizar o termo *resultado* de reparo como utilizado em (GARCEZ; LODER, 2005).

- Reparo na quarta posição: Iniciado pelo outro no quarto turno em relação à fonte do problema (SCHEGLOFF, 1992). Acontece quando, após uma troca completa o próprio autor não percebe o problema de entendimento, mas o outro.

Figura 3 - Exemplo de reparo na quarta posição.

1. A: Você foi à aula na sexta-feira?
2. B: Fui sim, e você?
3. A: Não fui, achei que não precisava da aula de reforço.
4. B: Que aula de reforço?
5. A: Da aula de inglês!
6. B: Ah, desculpe, você estava falando da aula de inglês! Eu não me matriculei para esse curso!

Importante notar que os dois últimos tipos, na terceira e na quarta posição, podem não acontecer exatamente nesta posição, como acontece no exemplo da Figura 3.

Note que o turno 1 possui uma referência problemática que só é definitivamente reparada em 6, que só pôde ser resolvida depois que uma segunda referência problemática, no turno 3, foi reparada em 5. Este tipo de situação requer que qualquer coisa já falada vá para o *espaço de reparação* (SCHEGLOFF, 1992), podendo vir a ser reparada muitos turnos depois. Esta característica tem efeitos importantes no *grounding* de sistemas de diálogo falado, inclusive levando à criação máquinas de estados que rastreiam o *grounding* de qualquer informação fornecida durante todo o diálogo, como em Young et al. (2010).

Para ser capaz de iniciar um reparo no próprio turno do trecho problemático, um sistema precisa ter capacidade de auto monitoração, isto é, de avaliar o efeito de sua fala ainda durante a execução da mesma, através, por exemplo, da avaliação do *feedback* simultâneo dado pelo interlocutor. Tal característica só é possível em sistemas de processamento incremental do diálogo, que podem começar a pronunciar um turno de fala e adaptá-lo ainda durante sua execução (SKANTZE; HJALMARSSON, 2010). É interessante observar que, diferente de um ser humano, um sistema computacional não precisa monitorar a ocorrência de erros de articulação, inexistentes na geração atual de softwares de síntese de fala. Se o trecho que

necessita reparo já foi falado, então ele é chamado de reparo evidente (*overt repair*); se ele ainda não foi, ele é chamado de reparo encoberto (*covert repair*). Esta diferenciação é fundamental para sistemas que trabalham o turno de forma incremental.

3 REVISÃO DA LITERATURA: O GERENCIAMENTO DO DIÁLOGO EM AGENTES CONVERSACIONAIS

O gerenciador de diálogos é o coração da arquitetura computacional de um agente conversacional embutido (ou *embedded conversational agent*, também chamados pela sigla *ECA*). Ele é o responsável pela tomada de decisão com propósito comunicativo. Em todas as arquiteturas que foram encontradas através da presente pesquisa, é possível identificar claramente um gerenciador de diálogos, ou módulo equivalente. Este capítulo faz um breve levantamento da literatura relacionada. Existem vários textos que descrevem um panorama mais profunda da área, como:

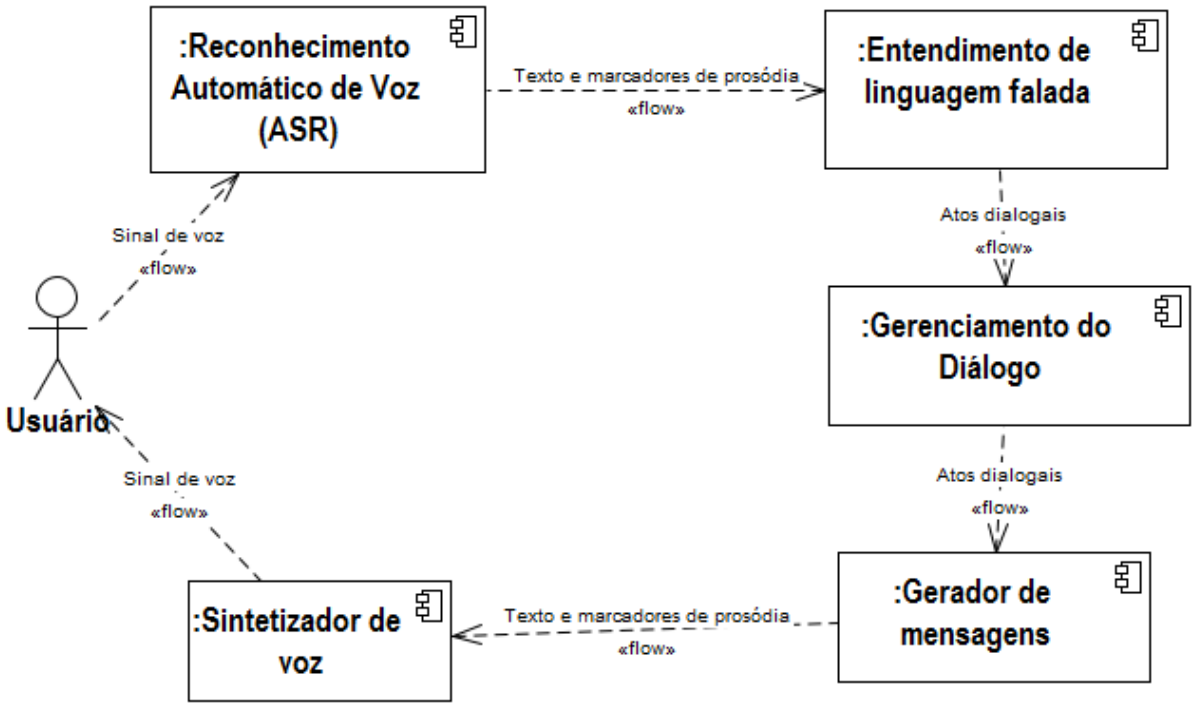
- McTear (2002): panorama geral das técnicas utilizadas até 2002 em sistemas de diálogo falado;
- Bui (2006): panorama geral das técnicas utilizadas em sistemas de diálogos multimodais;
- Thomson (2009): panorama de sistemas de diálogo falado com enfoque nas técnicas estatísticas e baseados em estado da informação;
- Dumas et al. (2009): panorama das técnicas e tecnologias utilizadas em aplicações multimodais, incluindo sistemas de diálogo;
- Petukhova (2011): amplo levantamento que inclui não só sistemas de diálogo falado, mas também modelos de diálogo de forma geral;
- Young et al. (2013): panorama dos trabalhos relacionados a gerenciamento de diálogo baseados em processo de decisão de Markov.

3.1 Arquitetura geral de um sistema de diálogo falado

A Figura 4 mostra a arquitetura típica de um sistema de dialogo falado. O reconhecedor de voz é o responsável por transformar os sinais de voz em texto e outras anotações relacionadas à prosódia. Entretanto, a prosódia e a parte verbal possam ser tratadas separadamente, como na arquitetura proposta por Alfenas & Pereira-Barretto (2012). O entendimento de linguagem falada é o responsável por separar o texto anotado em segmentos funcionais e definir os atos dialogais e a semântica de cada segmento. O gerenciamento do diálogo é o responsável por escolher uma resposta apropriada às entradas e tipicamente envolve dois tipos

distintos de tarefa: manter o estado do diálogo e a tomada de decisão propriamente dita; alguns trabalhos inclusive deixam explícita a separação (YOUNG et al., 2013). O gerador de mensagens transforma a saída do gerenciador de diálogo, tipicamente formada por atos dialogais e outros formatos estruturados, em texto em linguagem natural e anotações de prosódia, que são, finalmente, transformados em voz pelo sintetizador de voz. Lemon (2011) defende a ideia de que o gerenciamento de diálogo e a geração de mensagens em linguagem natural, se combinados, podem ser aperfeiçoados utilizando-se de planejamento baseado em de Processo de Decisão de Markov, o que resultaria em um modelo um pouco diferente do exibido na Figura 4.

Figura 4 – Arquitetura típica de um sistema de diálogo falado



Além desses componentes que cuidam de tarefas específicas do processo, existem outros componentes comuns a todo o ciclo de processamento, dos quais os mais importantes estão relacionados à fonte de conhecimento e de dados. Podemos listar, entre outros itens típicos deste grupo:

- Base de dados léxica, contendo nomes, verbos, adjuntos adverbiais e nominais, sinônimos, relações semânticas e léxicas, etc., do qual o WordNet (MILLER, 1995) é um exemplo;

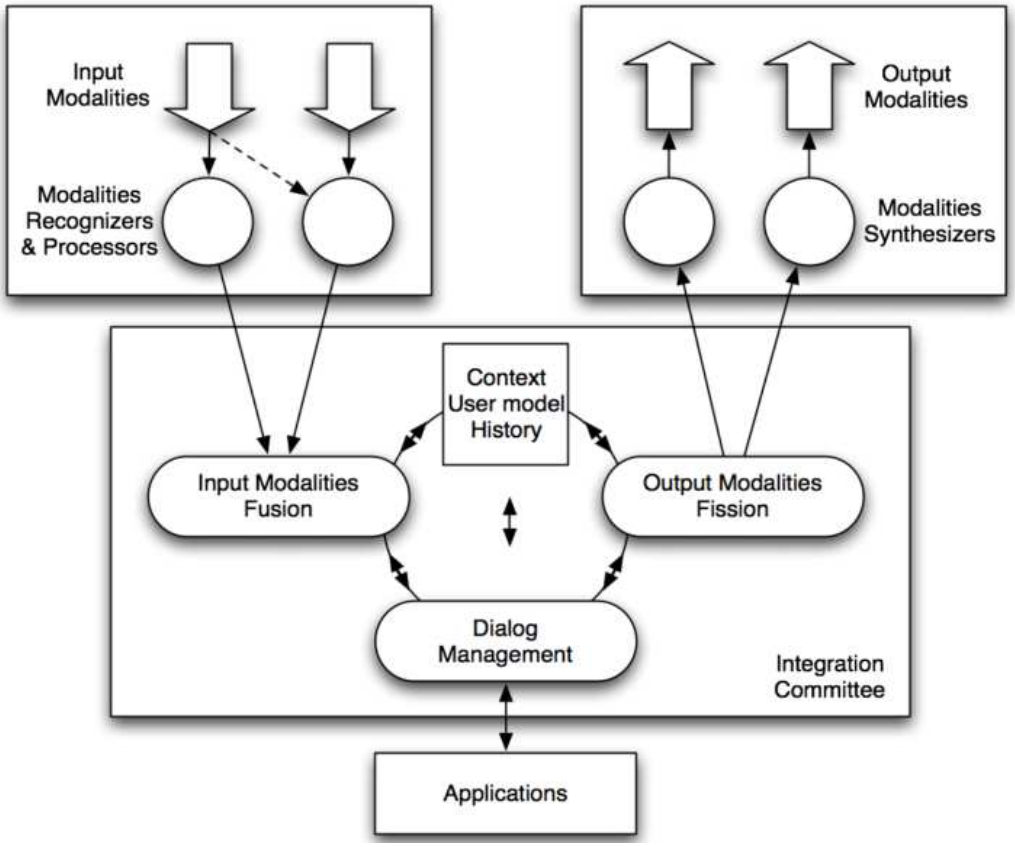
- Um modelo de conhecimento do mundo, tipicamente uma rede semântica contendo senso comum, como o ConceptNet (SPEER; HAVASI, 2012), ou do domínio específico de uma tarefa;
- Memória semântica, contendo as opiniões (também chamadas de *crenças*) sobre o mundo. Uma parte importante desta memória é o *modelo do usuário*, que contém as crenças do agente conversacional sobre um interlocutor específico e é atualizado turno a turno, contendo, comumente, a identificação, os objetivos e as intenções;
- Memória episódica, contendo o histórico de turno das conversas.

É importante ressaltar que um sistema de diálogo falado pode ter nenhum ou vários desses. Em alguns trabalhos revisados, como Kang et al. (2012) e Williams & Young (2007), existe um componente chamado *modelo do diálogo* que acumula várias das funções mencionadas.

O tipo de arquitetura apresentado na Figura 4 é, entretanto, típico de um processo baseado unicamente em voz. Embora alguns autores argumentem que seria fácil adicionar módulos adicionais neste ciclo, conectando entradas e saídas ao gerenciador de diálogos (THOMSON, 2009), manter o sincronismo entre diversas modalidades de entrada ao longo do tempo não é simples e comumente exige processos específicos que cuidem da fusão e da fissão multimodal. Uma arquitetura típica para sistemas multimodais em o aspecto exibido na Figura 5.

A principal função da fusão das modalidades de entrada (*Input Modalities Fusion* na figura) é extrair o significado dessas várias entradas. Ela pode ser realizada em três níveis: de dados, de características e de decisão. A fusão no nível de dados se dá entre múltiplos sinais de modalidades muito similares, como entre duas câmeras filmando a mesma cena, mas de pontos de vista diferentes. A fusão de características se dá entre diferentes modalidades que são altamente acopladas ou sincronizadas no tempo, como a voz e o movimento dos lábios. A fusão no nível de decisão é a mais comum, e envolve modalidades com menos acoplamento, como voz e teclado, e é tipicamente realizada após o processamento individual da semântica de cada uma.

Figura 5 - Arquitetura típica de um sistema multimodal.



Fonte: (DUMAS; LALANNE; OVIATT, 2009)

A fissão (*Output Modalities Fission* na Figura 5) é responsável por três tarefas. A primeira é construir uma mensagem com o conteúdo adequado a partir da ação (ou grupo de ações) decidida pelo gerenciador de diálogos. A segunda é a seleção das modalidades e canais de saída a utilizar para a transmissão da mensagem. A última é garantir que os diversos canais de saída estejam sincronizados no tempo e no espaço, de forma a tornar a comunicação natural.

O componente identificado pelas expressões *Context, User model* e *History* faz o papel da fonte de conhecimento para o gerenciamento de diálogo, e contém, respectivamente, contexto, modelo do usuário e histórico da conversa.

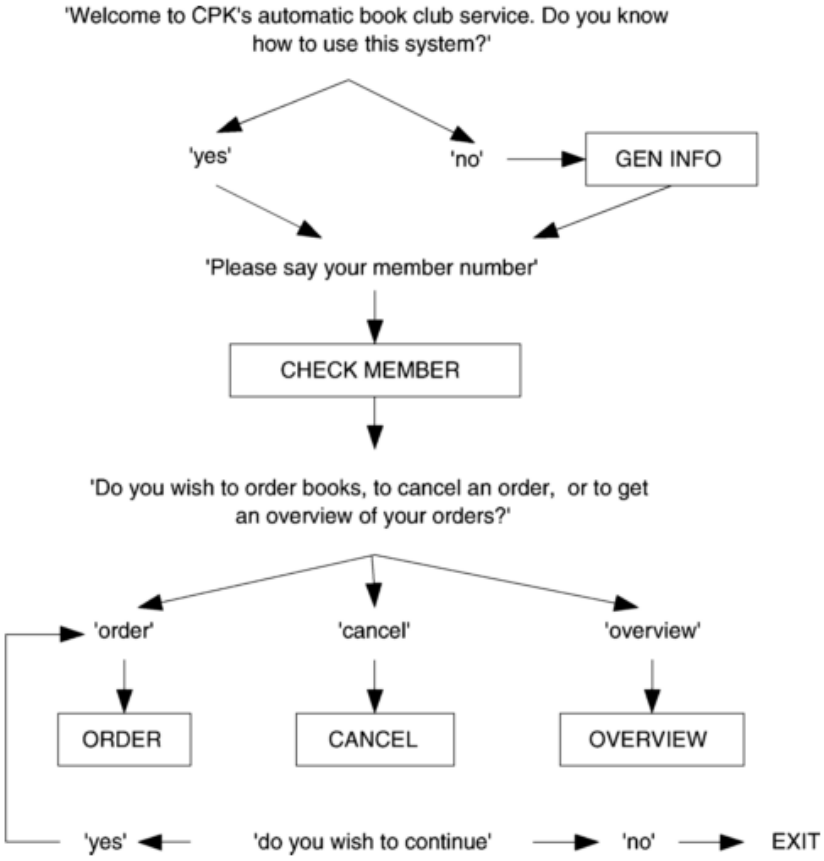
3.2 Classificações de sistemas de diálogo falado

Este tópico trata das classificações referentes a sistemas de diálogo falado e do gerenciamento do diálogo. Descrever estas classificações é importante, pois elas permitem situar as propostas feitas nesta dissertação.

3.2.1 Classificação de McTear quanto ao método de controle

Os sistemas de diálogo falado podem ser classificados conforme vários critérios. O método de controle utilizado no gerenciamento do diálogo descrito em McTear (2002) pode ser considerado um dos mais tradicionais e é utilizado por diversos dos trabalhos revisados, por exemplo Bohus & Rudnicky (2009) e Bui (2006). Esta classificação propõe a divisão dos sistemas de diálogo falado em baseados em máquinas de estados finitos, baseados em *frames*[3] e baseados em colaboração entre agentes.

Figura 6 - Exemplo de sistema baseado em máquina de estados finitos



Fonte: (MCTEAR, 2002)

Os sistemas baseados em máquina de estados finitos foram e talvez ainda sejam os mais utilizados em aplicações comerciais, conforme apontado por McTear. Neles, o

[3] O termo *frame* não foi traduzido por já ser utilizado na literatura em português. A tradução em expressões como moldura ou quadro poderia fazer com que se perdesse algum significado.

usuário é levado a uma sequência de passos pré-definidos. O fluxo é especificado como um conjunto de estados ligados por transições denotando os caminhos alternativos conforme as respostas do usuário às solicitações feitas pelo sistema, que detém a iniciativa. O modelo do diálogo está implícito nestes sistemas, pois a sequência e os itens de informação obtidos estão representados nos próprios estados e transições. A maior vantagem deste método é que o vocabulário e gramática podem ser especificados previamente, tornando mais simples o desenvolvimento, treino e teste do sistema. O maior problema é que ele dificulta a modelagem de situações em que o usuário obtém a iniciativa, isto é, situações em que o usuário faz as perguntas ou introduz um assunto. Outro problema é a impossibilidade de construir fluxos dependentes de contexto; por exemplo, não há maneira imediata de o sistema corrigir uma informação dada vários turnos antes, requerendo que o usuário solicite retornar ao passo anterior por várias vezes até atingir o ponto em que a informação errada foi dada, exceto se houver replicação a cada estado da transição que permite a correção da informação. Um exemplo de fluxo deste tipo de método é exibido na Figura 6, feito para um sistema automatizado simples para retirar livros de um clube de livros.

Nos sistemas baseados em *frames* ou *templates*[4], o sistema faz perguntas ao usuário que lhe permitem preencher lacunas (*slots*) em um modelo pré-definido para a tarefa corrente. Não existe uma ordem pré-definida neste caso, e o usuário pode fornecer mais informações do que as solicitadas pelo sistema, que escolhe suas ações baseado nas lacunas ainda não preenchidas. Outro critério utilizado para a tomada de decisão, utilizado em sistemas mais flexíveis, é o grau de confiança do sistema no seu entendimento sobre o que foi dito pelo usuário. Os sistemas desta categoria são bastante comuns comercialmente, principalmente quando os diálogos são focados em fornecimento de informações, como é o caso de sistemas de informação a turistas.

Os sistemas baseados em colaboração entre agentes (ou sistemas baseados em inteligência artificial) são mais complexos que os anteriores. O nome da categoria decorre do fato de que todos os participantes de uma mesma interação são considerados como agentes pelo sistema, sejam eles humanos ou artificiais. Estes sistemas são normalmente dotados de alguma capacidade de raciocínio sobre suas próprias atitudes e crenças e por vezes até mesmo sobre as dos demais agentes. O

[4] A tradução da expressão *templates* comumente usada para o português seria *modelos*, o que seria ambíguo neste texto. Desta forma, a expressão original em inglês foi mantida.

modelo computacional pode manter uma lista de expectativas sobre a próxima ação do usuário, o que permite construir um plano de intervenções do sistema para cada expectativa. A iniciativa pode partir tanto do sistema quanto do usuário.

3.2.2 Outras classificações mais recentes quanto ao método de controle

Nos últimos dez anos, a quantidade de trabalhos publicados relatando sistemas que não se baseiam em máquina de estados finitos ou em *frames* cresceu bastante. Novas categorias têm sido propostas para a classificação por método de controle, embora ainda não haja um consenso nos textos consultados.

Bui, em seu trabalho sobre a revisão do gerenciamento do diálogo multimodal (BUI, 2006), classifica os métodos em quatro categorias. A primeira é composta pelas categorias de sistemas baseados em máquinas de estados finitos e baseados em *frames* ou *templates* de McTear. A segunda compreende os métodos de atualização do estado da informação. A terceira inclui os métodos baseados em planejamento. A quarta se refere aos métodos baseados em colaboração entre agentes de McTear. O segundo e o terceiro métodos são descritos a seguir.

Um *sistema baseado na atualização do estado da informação* para o gerenciamento do diálogo (TRAUM; LARSSON, 2003) precisa ter cinco partes claramente definidas: (i) a descrição dos componentes informacionais (exemplos: participantes, crenças, *common ground*), (ii) uma representação formal de tais componentes informacionais, isto é, através de listas, conjuntos, estruturas com atributos tipados, etc., (iii) um conjunto de eventos de diálogo que causam atualização no estado destes componentes informacionais, (iv) as regras de atualização de estado e (v) a estratégia de atualização para decidir quais regras aplicar em resposta a cada evento ou conjunto de eventos. Existem alguns métodos estatísticos de gerenciamento nesta categoria, que utilizam o processo de decisão de Markov (doravante chamado de MDP, do inglês *Markov Decision Process*) e o processo de decisão de Markov parcialmente observável (doravante chamado de POMDP, do inglês *partially observable MDP*)[5]. A principal característica deles é a definição de uma função de recompensa para as ações do sistema baseada nas atitudes do usuário e uma política

[5] Mais uma vez, preferimos manter as siglas em inglês MDP e POMDP do que diferir de outros trabalhos em português do Brasil.

de escolha de ação que procure aperfeiçoar o ganho destas recompensas. Estes sistemas serão discutidos com mais detalhe adiante neste capítulo.

Os sistemas *baseados em planejamento* seguem os princípios das teorias baseadas em planejamento do diálogo e da ação comunicativa, que defendem a ideia de que o ato discursivo do interlocutor é parte de um plano e que é trabalho do ouvinte identificar e responder apropriadamente a este plano. Este modelo relaciona um plano no nível do domínio da tarefa com um plano comunicativo, como, por exemplo, em Kang et al. (2012). Os planos são estruturados de forma a definir explicitamente restrições, pré-condições, decomposição em outros planos e ações, efeitos e objetivos relacionados, entre outros atributos que um sistema deste tipo pode considerar. De acordo com Bui (2006), este modelo é capaz de suportar diálogos mais complexos que os anteriores e de acordo com Petukhova (2011) eles permitem um tratamento mais racional. Mas ele também possui várias desvantagens, como intratabilidade (ou até mesmo indecidibilidade) no pior caso e a falta de uma teoria sólida que a suporte.

Um único sistema de diálogo falado pode ser classificado em mais de uma das categorias acima, isto é, elas não são exclusivas. Bui (2006) exemplifica com o projeto Smartkom, que é ao mesmo tempo baseado em colaboração entre agentes, planejamento e atualização do estado da informação.

Petukhova (2011) faz uma classificação dos modelos de diálogo que não é focada em sistemas de diálogo falado, mas mesmo assim válida para eles. São três as categorias citadas neste trabalho: métodos baseados no estado da informação, métodos baseados em planejamento e métodos baseados em gramáticas. Este último é baseado na observação dos padrões do diálogo, como nos pares adjacentes e em outras estruturas do diálogo. Os sistemas baseados em máquinas de estados finitos e em frames se encaixam nessa classificação. Uma crítica feita a estes métodos é que eles ignoram o conteúdo semântico e o caráter multifuncional das falas. Embora Petukhova mencione este último como defeito desta categoria, estes defeitos também podem acontecer em sistemas que utilizam outros métodos, pois esta característica depende da implementação em particular feita pelos desenvolvedores, que podem ignorar o caráter multifuncional, quer seja o sistema baseado em planejamento ou no estado da informação.

3.2.3 Classificação quanto às modalidades de comunicação

Os sistemas de diálogo falados também podem ser classificados em três grandes grupos quanto às modalidades de comunicação suportadas (SCHULLER et al., 2013). O primeiro é o grupo dos que suportam apenas comunicação verbal e que compreende a maioria dos sistemas comerciais, quer o canal de comunicação seja de texto ou de voz. O segundo é o grupo dos sistemas chamados de *paralinguísticos*, que são baseados na análise de um único canal de comunicação, tipicamente a voz, mas considerando tanto os aspectos verbais quanto os não verbais, como prosódia, timbre, risadas e mesmo grunhidos (SCHULLER et al., 2013), o que permite uma análise mais profunda sobre o estado físico e mental do interlocutor. O último grupo compreende os sistemas *multimodais*, que envolvem vários canais de comunicação, como a voz, a visão, o posicionamento e outras informações de contexto que não linguístico. Tanto os sistemas paralinguísticos quanto os multimodais compartilham as características arquiteturais de fusão e fissão já mencionadas.

3.2.4 Baseado em conhecimento versus baseado em dados

Um sistema é classificado como baseado em conhecimento quando as regras nele utilizadas são feitas artesanalmente (do inglês *handcrafted rules*), isto é, são definidas cuidadosamente por um humano especialista no domínio de negócio do sistema. Geralmente, tal sistema está limitado a executar tarefas altamente estruturadas, restringindo o diálogo a uma linguagem bastante regulada e sem surpresas, com o sistema retendo a iniciativa. Os sistemas baseados em máquinas de estados finitas e muitos outros baseados em gramáticas caem nessa categoria. Pouco ou nenhum raciocínio e aprendizado são feitos pelo gerenciador de diálogos. Estes sistemas são muito pouco flexíveis para o usuário e as regras são pouco portáveis, isto é, portar o sistema de um domínio de negócio para outro requer que grande parte dele seja refeito. Mas, por outro lado, este modelo é bastante simples de entender, de desenvolver e de testar, devido à sua previsibilidade. Entretanto, vários esforços têm sido feitos para reduzir os problemas deste modelo (CHEONGJAE LEE et al., 2010). Um destes esforços é no sentido de separar claramente os aspectos dependentes de um domínio específico daqueles aspectos do diálogo que são independentes e colocar estes últimos em uma plataforma reutilizável que acelere o desenvolvimento em novos

domínios. Tal é o caso do Ravenclaw (BOHUS; RUDNICKY, 2009). Questões como gerenciamento de turnos, controle do *grounding* das informações e tratamento de erros (*feedback*, solicitações de *feedback* e repetições) são realizados pelo Ravenclaw, enquanto características específicas do domínio da tarefa são definidas pelo autor do sistema final através de arquivos de configuração e uma extensão da linguagem C++. Algumas das características do Ravenclaw pertencem à categoria de sistemas baseados em planejamento, pois ele força que o desenvolvimento seja feito seguindo uma modelagem hierárquica do planejamento das tarefas, e outras características pertencem a categoria dos sistemas baseados em frames, pois as informações que precisam ser preenchidas para a execução da tarefa ficam claramente definidas. Este sistema é discutido com mais detalhes adiante neste mesmo capítulo.

Um sistema baseado em dados é capaz de aprender novas regras a partir um conjunto de diálogos de treinamento devidamente anotados. Desta forma, o custo de desenvolver as regras para um sistema em outro domínio é aproximadamente o mesmo de coletar novos dados neste domínio e anotá-los. Uma das desvantagens é que esta técnica remove muito do controle desejado pelo desenvolvedor e torna o refinamento do diálogo difícil, aumentando consideravelmente o esforço necessário para testes. Essas desvantagens podem ser reduzidas pela aplicação de técnicas de aprendizado supervisionado e de aprendizado online, isto é, o sistema é disponibilizado para uso final após um treino mínimo e é atualizado com os próprios dados de produção, devidamente supervisionados.

Trabalhos mais recentes têm proposto, cada vez mais, técnicas híbridas, baseadas tanto em conhecimento quanto em dados (CHEONGJAE LEE et al., 2010). Por exemplo, técnicas de simulação de usuário são utilizadas para gerar um grande número de diálogos a partir de um *corpus* pequeno baseado em um modelo específico ou criado diretamente por um especialista. Esses diálogos são, então, utilizados para treinar um sistema baseado em dados.

3.2.5 Modelos incrementais e não incrementais

O processamento do diálogo entre humanos sempre é incremental, isto é, as pessoas não esperam pelo fim do turno do falante para iniciar a compreensão do que está sendo falado. Entretanto, a maioria dos trabalhos pesquisados relatam sistemas de

diálogo falado não incrementais, que processam os turnos de uma única vez após seus términos.

Entre as propostas de processamento incremental, foram revisados os descritos em Schlangen & Skantze (2009), Skantze & Hjalmarsson (2010) e Traum et al. (2012). O que fica claro a partir destes trabalhos é que este tipo de processamento requer um modelo especial para a arquitetura do sistema como um tudo. Cada componente precisa definir sua unidade incremental mínima, que é necessariamente mais segmentada do que um turno completo, podendo ser frases, palavras ou mesmo sílabas. Para tirar proveito real, algum tipo de predição deve ser feito pelos componentes que trabalham de forma incremental, o que implica também na necessidade de um mecanismo mais robusto para revogar hipóteses, pois muitas delas são criadas e abandonadas antes do final do turno, como consequência de predições que não se realizam.

No trabalho de Schlangen & Skantze (2009) é definido um protocolo genérico de atualização de hipóteses, utilizado entre todos os módulos. Os tipos de mensagens incluem a revogação (*purge*), que avisa outros componentes que uma hipótese previamente considerada está sendo descartada, a atualização (*update*), que avisa sobre novas hipóteses e atualizações, e o compromisso (*commit*), em que um módulo avisa os demais que está utilizando uma determinada hipótese de forma definitiva em detrimento de outras. Esta ideia é evoluída no trabalho de Skantze & Hjalmarsson (2010), que descreve a plataforma Jindingo para sistemas com processamento incremental. Nele, Skantze também descreve um experimento com um sistema de itens usados chamado DEAL, desenvolvido sobre o Jindingo, em que os usuários o classificaram como mais eficiente e mais educado quando comparado a um sistema equivalente com arquitetura não incremental. Uma das maiores razões para este resultado é que um sistema incremental tem melhores opções de estratégia de *feedback*, simultâneas à fala, mais naturais e diferentes daquelas listadas na Figura 2. Traum et al. (2012) também defende esta ideia.

3.3 Detalhes dos componentes de arquitetura

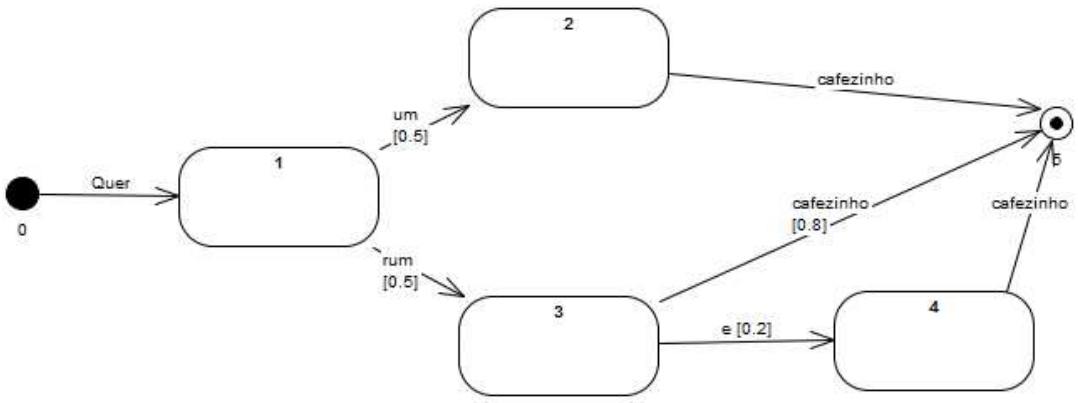
Esta seção descreve os componentes que compõe um sistema de diálogo falado exibidos na Figura 4.

3.3.1 Reconhecimento de voz

O reconhecimento de voz é o responsável por transformar os sinais de voz, normalmente acompanhados de ruído, em texto, algumas vezes acompanhado de uma medição de confiança na interpretação e de análise de prosódia (SAINI; KAUR, 2013). Existem diversos softwares comerciais e públicos, inclusive com suporte para o português brasileiro e que disponibilizam uma interface de programação para integração com outros softwares.

Embora esses softwares tenham melhorado muito nos últimos anos, a taxa de erros ainda é bastante significativa. O desempenho do reconhecedor está ligado diretamente a vários fatores, como quantidade de dados no domínio de treinamento, dificuldade da tarefa à qual é proposto e variação de características dos interlocutores, como timbre, sotaque, idade e sexo. Se o treinamento for feito com um domínio limitado, mas o usuário final puder falar livremente, a taxa de erro será bastante alta (THOMSON, 2009). Muitos reconhecedores têm como saída apenas a hipótese mais provável sobre a fala, mas alguns mais recentes são capazes de fornecer uma lista das hipóteses mais prováveis, junto aos respectivos graus de confiança. Esta lista costuma ser chamada de lista de *N-melhores* (adaptado do inglês *N-best list*) hipóteses.

Figura 7 – Exemplo de rede de entrelaçamento de palavras usada como saída de um reconhecedor de voz



Alguns reconhecedores possuem como saída um grafo direcionado e acíclico de *entrelaçamento de palavras* (do inglês *word lattice*). Cada nó está associado a um instante da fala, e cada transição está associada a uma possível hipótese de palavra

pronunciada e a probabilidade desta hipótese. Um exemplo pode ser visto na Figura 7. Thomson argumenta que sempre é possível converter uma rede deste tipo em uma lista de N-melhores hipóteses.

3.3.2 Entendimento de linguagem falada

Em um sistema de diálogo, é mais importante entender a intenção do usuário do que a representação semântica exata do que foi dito (MCTEAR, 2002; THOMSON, 2009). Se o usuário diz “Qual é o endereço do restaurante?” ou “Eu gostaria do endereço do restaurante.”, a intenção do usuário e a expectativa sobre a próxima intervenção do sistema é a mesma. Por esta razão o entendimento da linguagem falada, em alguns casos também chamado de *analisador semântico* (ALFENAS; PEREIRA-BARRETTO, 2012), tem que ter como saída os atos dialogais relativos aos segmentos funcionais analisados e o conteúdo semântico apropriado aos atos dialogais. No exemplo citado, o sistema deveria compreender ambas as ações do sistema como um ato dialogal de função comunicativa *request*[6], cujo conteúdo semântico associado é apenas uma referência para a informação desejada, no caso um endereço pertencente a um restaurante que já foi *grounded*. O conteúdo semântico, quando acompanhando um ato dialogal, é chamado de *item do ato dialogal* (THOMSON, 2009).

Os atos dialogais suportados por um sistema devem ser definidos a priori, e podem conter um subconjunto ou uma extensão daqueles definidos pela ISO 24617-2. O formato de saída deve considerar que um mesmo turno falado do usuário pode conter vários segmentos com atos dialogais e conteúdos semânticos distintos, e que possui relações funcionais, de feedback e retóricas entre si e com segmentos de outros turnos. Existem várias outras dificuldades nestas tarefas, como definir entidades nomeadas e relacioná-las a itens em memória e resolver anáforas e elipses. Além disso, o modelo precisa lidar com múltiplas hipóteses paralelas, tanto devido aos erros e múltiplas hipóteses de entradas recebidas do reconhecedor de voz, quanto aos erros introduzidos no próprio processo de análise semântica decorrentes de ambiguidades. Conforme Thomson, o processo de definição do ato dialogal é chamado de *decodificação semântica*. Ainda, segundo ele, existem várias técnicas disponíveis

[6] Manteremos o uso dos nomes em inglês dos atos dialogais do ISO-24617-2 sempre que possível.

para esta decodificação, tais como: baseadas em conhecimento, baseadas em reconhecimentos de modelos (*templates*) e gramáticas e baseadas em dados. Exemplos deste último incluem máquinas de vetores de suporte (SVM, *support vector machines*) e transdutores de estado finito com peso.

3.3.3 Geração de respostas e síntese de voz

Existem três categorias principais de métodos para transformar a saída do gerenciador de diálogo, tipicamente atos dialogais acompanhado de itens semânticos (LEMON, 2011), em linguagem natural. A primeira, bastante comum em sistemas comerciais, é o de preenchimento de lacunas seguindo um modelo (*template*). Esta técnica, se utilizada de forma pura, requer bastante trabalho manual, pois é necessário mapear cada saída do gerenciador para um *template*. A segunda categoria é a de geração de texto em linguagem natural *convencional*, que requer a definição de três módulos principais: um planejador que seleciona o conteúdo a ser utilizado e define a estrutura geral do discurso de saída (do ponto de vista retórico); um planejador de sentença que decide quais expressões de referência usar (isto é, se será usado um nome, um pronome, um atributo, etc. para referenciar os sujeitos e objetos) e agrega estruturas retóricas em sentenças; e, finalmente, um módulo que converte estas sentenças estruturadas em linguagem natural. Esta categoria convencional de geração ainda requer bastante trabalho manual para criação das regras de geração e é de modelagem mais complexa que os baseados em *template*, embora seja mais fácil de adaptar e reutilizar. A última categoria é de geradores *treináveis*, e é baseado em técnicas estatísticas de aprendizado supervisionado. Lemon (2011) cita algumas técnicas possíveis nessa categoria, como a modelagem de uma linguagem de bigramas, a extração de regras a partir de um *corpus* e o planejamento de sentenças baseados em atos dialogais e as relações retóricas entre eles. É interessante citar que pouco trabalho foi encontrado sobre a geração de respostas que considerasse características não linguísticas, como emoções. Se for necessário, é uma pesquisa que precisa ser mais bem realiza na continuidade deste trabalho.

A síntese de voz é atualmente bastante usada em aplicações comerciais. Sistemas operacionais como as versões mais novas do Microsoft Windows, já possuem um sintetizador embutido, considerado essencial para tornar o sistema mais acessível. Existem sistemas que suportam anotações de prosódia na saída.

3.4 Gerenciamento do diálogo

A principal função do gerenciador de diálogo é controlar o fluxo do diálogo. Existem vários tipos de técnicas utilizadas no gerenciamento, como as já citadas baseadas em máquinas de estados finitos e baseadas em agentes, entre outras. Neste tópico é feita uma revisão detalhada de dois métodos de gerenciamento, o utilizado pela plataforma de desenvolvimento Ravenclaw e o baseado em POMDP, como representantes, respectivamente, dos sistemas baseados em planejamento modelados a partir do conhecimento e dos sistemas baseados na atualização do estado da informação modelados a partir de dados.

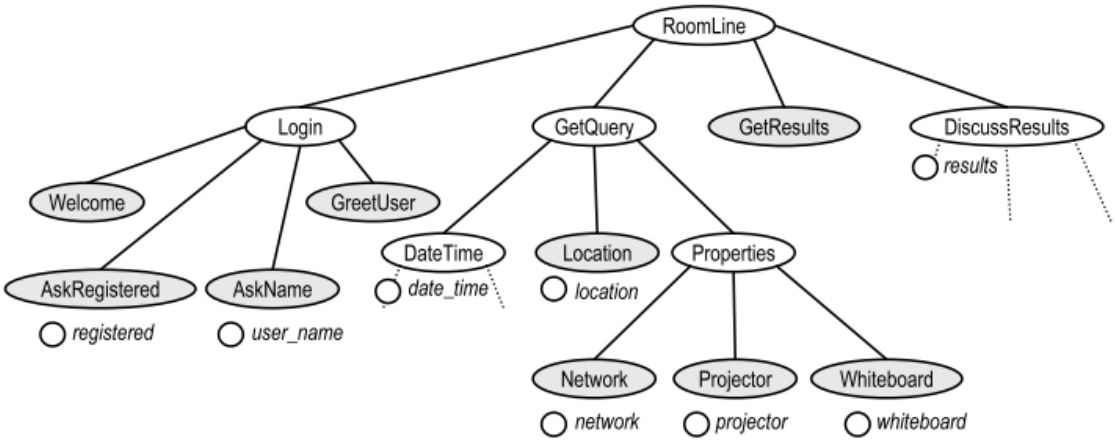
3.4.1 Ravenclaw

Esta plataforma para criação de gerenciadores de diálogo falado já foi utilizada em muitas aplicações diferentes, como acesso a informação, diagnóstico médico e guia de procedimentos (BOHUS; RUDNICKY, 2009). O principal propósito de seu projeto foi a separação clara entre tarefas específicas de um domínio e aspectos independentes, e que estes pudessem ser reutilizados em vários sistemas, aplicados a áreas diferentes de negócio. A lógica de controle deve ser especificada através de um plano hierárquico das tarefas específicas do domínio, representada no sistema por uma árvore de *agentes dialogais*.

A Figura 8 exibe um exemplo desta árvore, referente a uma parte da tarefa de alocação de salas de reunião para o sistema *RoomLine*. O nó principal tem quatro filhos: o *login*, responsável por obter a identificação do usuário no sistema; o *GetQuery*, responsável por obter os parâmetros de busca de sala; o *GetResults*, que faz a busca da sala no banco de dados da aplicação; e o *DiscussResults*, que apresenta o resultado obtido para o usuário. Alguns desses nós são decompostos em outras tarefas e assim sucessivamente. Existem duas categorias de agentes neste sistema: os *fundamentais*, marcados em cinza na figura, e as *agências dialogais*. A primeira categoria é formada apenas pelas folhas da árvore (isto é, estão em posições terminais e não têm outros agentes como filhos) e representam ações atômicas. Há quatro tipos de agentes fundamentais: *Inform*, que produz uma saída falada informativa; *Request*, que solicita uma informação ao usuário; *Expect*, que representa

informações que o usuário pode dar livremente, mas que não são solicitadas pelo sistema em caso contrário; e *Execute*, que executa uma tarefa específica do domínio da aplicação, como uma pesquisa em banco de dados. A categoria agência, em contraposição, contém agentes que ocupam as posições não terminais, e seu propósito é controlar a execução das ações nele contidas. Cada agente pode especificar pré-condições, situações de disparo e critérios de sucesso ou de falha, utilizados pela plataforma e agências durante o planejamento de execução dos vários agentes na árvore.

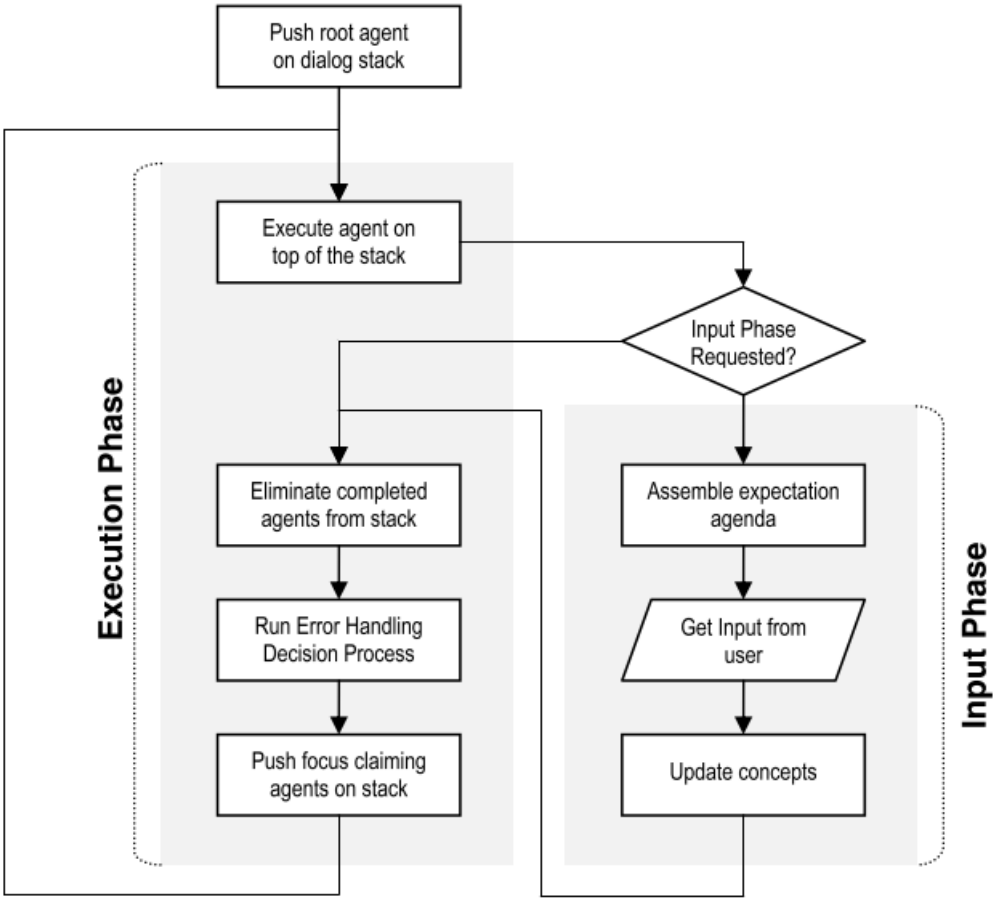
Figura 8 - Exemplo de uma árvore de agentes dialogais descrevendo uma tarefa para o sistema *RoomLine*, utilizando o Ravenclaw



Fonte: (BOHUS; RUDNICKY, 2009)

Os agentes podem manipular dados, que são encapsulados no sistema através de *conceitos*. Na Figura 8, por exemplo, o conceito *registered* está associado ao agente *askRegistered*. Os conceitos podem ser de vários tipos básicos nativos da plataforma, como booleanos, *Strings* (cadeias de caracteres), inteiros e ponto-flutuante, embora o desenvolvedor possa especificar tipos mais complexos, como estruturas e vetores. O Ravenclaw mantém, para cada conceito, uma lista de pares *(valor, grau de confiança)* que servem de parâmetro para o sistema controlar o *grounding* das informações requeridas para a execução de cada tarefa do domínio. Na prática, as tarefas são especificadas através de uma extensão da linguagem C++, contendo um conjunto de macros pré-especificadas.

Figura 9 – Fluxo de execução do Ravenclaw



Fonte: (BOHUS; RUDNICKY, 2009)

A estrutura interna do Ravenclaw é baseada em duas estruturas principais: uma pilha de discurso, que contém a estrutura do discurso em tempo de execução, e uma agenda de expectativas, que lista as possíveis ações do usuário esperadas pelo sistema a cada instante. A agenda de expectativas ajuda a resolver ambiguidades semânticas, pois ela oferece informação sobre a probabilidade das respostas que o usuário pode dar. O diálogo é controlado interpolando-se fases de entrada com fases de execução, conforme a Figura 9. Durante a fase de execução, a plataforma processa o agente no topo da pilha de discurso. Por exemplo, agentes do tipo *Inform* e *Request* irão causar uma saída falada, enquanto o *Request* irá também esperar por uma ação do usuário, fazendo com que o sistema vá para a fase de entrada. Ao término do processamento do agente, a plataforma elimina da pilha todos os agentes cujas condições de finalização foram satisfeitas, mesmo que ainda não tenham sido processados, o que pode acontecer no caso do usuário fornecer mais informações do que o solicitado pela última ação do sistema. Em seguida, o sistema executa o

processo para tratamento de erros. Se qualquer erro precisa ser tratado no diálogo, um agente manipulador de erros é colocado no topo da pilha do discurso. Finalmente, no último passo de execução, o sistema procura agentes na pilha que estejam solicitando o foco de execução (baseado nas condições de disparo especificadas), e os move para o topo. Durante a fase de entrada, o sistema utiliza a agenda de expectativas para preencher a lista de conceitos a partir das observações feitas sobre a última ação do usuário.

O Ravenclaw classifica os erros que podem acontecer em dois tipos: mal-entendidos e desentendimento total. O último acontece quando o sistema falha em obter qualquer informação útil do usuário e o primeiro acontece quando há entendimento parcial ou total, mas que pode ser incorreto (isto é, com grau de confiança abaixo de um aceitável) ou não ter relação com as expectativas atuais. A plataforma possui uma lista de estratégias de recuperação de erro e um processo de tomada de decisão que dispara estas estratégias no momento apropriado. Existe uma categoria de estratégia para cada um dos tipos de erro; para mal-entendidos pode-se optar por uma confirmação explícita ou implícita do que foi entendido. As estratégias são codificadas como agentes e já estão embutidas na plataforma. Além de lidar com erros, o Ravenclaw também possui estratégias embutidas para lidar com outras situações, como estouro de tempo limite para resposta do usuário, reiniciar a conversa, suspender e retornar a conversa, etc.

3.4.2 Baseados em Processos de Decisão de Markov Parcialmente Observáveis

O método para gerenciamento de diálogo baseado em processos de decisão de Markov parcialmente observáveis, como proposto por Williams & Young (2007), tem como propósito lidar com três questões essenciais: aprimorar o processo de detecção de erros em relação a outros modelos; ser capaz de planejar suas ações em prazos longos; e admitir hipóteses paralelas sobre a ação do usuário e sobre os itens do modelo do diálogo que precisam ter sua situação de *grounding* controlada. Para entender este modelo é necessário, primeiro, um entendimento dos modelos do POMDP e mesmo de MDP (*Markov Decision Process*, Processo de Decisão de Markov). Em seu trabalho, Williams e Young fazem um bom resumo, mas uma revisão mais detalhada sobre ambos os modelos encontra-se em Shani et al. (2012), que também avalia os métodos existentes para a escolha da melhor política de decisões.

Em seu trabalho, Williams e Young propõem a seguinte representação para o gerenciamento do diálogo. O usuário possui algum estado interno correspondente a seu objetivo atual g. Baseado nele, o usuário executa a ação a_u, que passa pelo reconhecedor de voz ou de gestos e módulos de entendimento semântico, gerando um conjunto de hipóteses H sobre a_u, cada uma composta por um par (observação o, grau de confiança c). Além disso, o histórico do diálogo, contendo as ações do usuário, do sistema, situação de *grounding*, etc., é mantido em um histórico do dialogo s_d. O gerenciador utiliza então H para tentar determinar o estado real do diálogo, definido em função de s_d, a_u e g. Uma vez que não é possível determinar com certeza o estado do diálogo, o estado do sistema é uma distribuição com a probabilidade de se estar em cada um dos estados do diálogo, chamado *estado de crença*, ou b_s. Baseado em b_s, o gerenciador escolhe e executa a ação do sistema a_s, que levará o usuário a possivelmente alterar o objetivo de g para g' e a escolher uma nova ação a_u', reiniciando o ciclo.

Agora podemos definir o modelo do POMDP para sistemas de diálogo falado (doravante chamado de *SDS-POMDP*) como $M = (Q, A_s, T, R, O, Z, \lambda, b_0)$, em que:

- Q é o conjunto de estados, cada um definido na forma (s_d, a_u, g)
- A_s é o conjunto de todas as ações a_s que o sistema pode realizar
- T é o conjunto de probabilidades de transição entre estados. Cada probabilidade depende apenas do estado corrente s e da última ação do sistema a_s, e pode ser representada na forma $P(s'|s,a_s)$, em que s' é o estado de destino.
- R define as expectativas de recompensa imediata em cada estado, cada uma na forma r (s_d, a_u, g,,a_s), isto é, a função de recompensa r depende apenas de s_d, a_u, g e,a_s.
- O é o conjunto de observações possíveis sobre a ação do usuário
- Z é a distribuição de probabilidades de observação $P(o|s',a_s)$, isto é, a probabilidade de se observar o depende apenas de a_s e do próximo estado s.
- λ é o fator de desconto geométrico para cálculo da expectativa de recompensa em longo prazo.
- b_0 é o estado de crença inicial.

Williams e Young demonstram que a forma final da transição do modelo do SDS-POMDP é dada por:

$$p(s'|s,a) = p(g'|ga_s)p(a'_u|g'a_s)p(s'_d|g'a'_u s_d a_s) \quad (1)$$

Como a observação depende apenas da última ação do usuário, a distribuição de probabilidades de *o'* pode ser dada por:

$$p(o'|s',a_s) = p(o',c'|a'_u) \quad (2)$$

(1) e (2) representam o modelo estatístico do SDS-POMDP proposto por Williams e Young. Ambos os modelos têm que ser estimados, possivelmente a partir de um *corpus* de interações anotadas. Entretanto, é importante observar que o *corpus* precisa ser baseado em interações homem-máquina e não interações reais entre humanos. Para isto, existem técnicas de auxílio ao desenvolvimento, chamadas de *simulação de usuário* e *mágico de Oz*, ambas discutidas no tópico "Aspectos importantes da metodologia de desenvolvimento", ao final deste capítulo

A partir das definições acima, é possível calcular o estado de crença a cada passo, conforme demonstrado por Williams e Young, pela equação:

$$b(g',a'_u,s'_d) = k \times p(o',c'|a'_u) \times p(a'_u|g',a_s) \quad (3)$$
$$\times \sum_{g \in G} p(g'|g,a_s) \sum_{s_d \in S_d} p(s'_d|g'a'_u s_d a_s) \sum_{a_u \in A_u} b(g,a_u,s_d)$$

onde *k* é a constante de normalização. Esta equação deve ser calculada para cada estado (g', a'_u, s'_d) que precisa ser atualizado no estado de crença.

Uma vez atualizado o estado de crença, o gerenciador escolhe uma nova ação a'_s, utilizando para isso uma política de escolhas $\pi(b)$. Esta política é chamada de ótima quando ela maximiza a expectativa de recompensas; representá-la-emos por $\pi^*(b)$. Encontrar a política ótima é um problema geralmente intratável, mas existem métodos que calculam soluções aproximadas para $\pi^*(b)$. Discutir essas técnicas está fora do escopo desta revisão.

Entre as vantagens deste modelo básico do SDS-POMDP podemos destacar a naturalidade para considerar diversas hipóteses paralelas de uma maneira bastante precisa e viabilizar um método de automatização e inferência do planejamento através da política de escolha das ações. Entretanto, ele apresenta alguns problemas. O maior é, possivelmente, a definição da função de recompensa. A maneira mais simples é definir uma recompensa positiva no caso de um fechamento positivo do diálogo, uma recompensa negativa no caso de um fechamento negativo e zero como recompensa nos demais casos, ou uma recompensa negativa mínima para evitar que o diálogo se prolongue indefinidamente.

Outra desvantagem é que o tamanho do estado de crença cresce muito rapidamente. Em um sistema que tenha em sua base mil cidades como possíveis destinos de viagem, o sistema precisa manter uma distribuição considerando que o usuário tem como possível objetivo cada uma destas mil cidades. O modelo do *estado da informação oculta*, ou, como comumente chamado, *HIS* (do inglês *Hidden Information State*) (YOUNG et al., 2010), é o mais conhecido a propor uma solução para este problema. Ele propõe que estados similares devem ser agrupados em partições, e uma única probabilidade de crença é mantida para a partição. Desta forma, estados que ainda não foram evidenciados não são rastreados de forma específica.

Outra proposta interessante do modelo do estado da informação oculta para diminuir o tamanho do POMDP e, por consequência, a complexidade do processo de tomada de decisão é a criação de um *espaço sumarizado de crença*, uma versão compacta do estado de crença do SDS-POMDP tradicional, chamado por Young de *espaço mestre de crença*. O processo de tomada de decisão inicia, portanto, com o estado de crença sendo mapeado no espaço sumarizado, para ser mapeado em seguida em uma *ação sumarizada* através da política $\pi(b)$. A ação sumarizada é então mapeada de volta no espaço mestre para finalmente ser enviada para a geração da mensagem de saída. Cada ponto no espaço sumarizado é um vetor composto pelas duas hipóteses mais prováveis no espaço mestre e a última ação do usuário. O HIS utiliza uma função quadrática de distância para mapear um estado do espaço mestre em um ponto do espaço sumarizado.

Mesmo no caso do HIS, se um mecanismo de recombinação das partições não for utilizado, a quantidade de partições pode crescer exponencialmente em relação ao tamanho da lista de N-melhores hipóteses de entrada (WILLIAMS; AVE; PARK, 2010). As técnicas para recombinação das partições são chamadas de técnica de *poda* (*prunning*) da árvore de partições. A mais simples consiste em remover as partições com menor probabilidade, recombinando-as à partição ancestral da qual se originaram. Um método mais sofisticado e mais eficaz em domínios com conceitos que possuem muitos atributos é proposto em (YOUNG; GASIC, 2011).

3.5 Aspectos importantes da metodologia de desenvolvimento

O processo de desenvolvimento de um sistema de diálogo falado pode ser visto como um caso especial de engenharia de software (MCTEAR, 2002), com alguns métodos

de desenvolvimento e avaliação específicos. O processo de desenvolvimento precisa definir quais tarefas o sistema deve ser capaz de realizar, especificar uma estrutura de diálogo para suportar cada uma dessas tarefas, o vocabulário a ser reconhecido e as estruturas linguísticas que devem ser atendidas. Alguns métodos comuns ao processo tradicional de levantamento de requisitos se aplicam: revisão de literatura, entrevistas com os usuários, observações de campo e experimentos ou provas de conceito. Entretanto, a observação de interações entre humanos fornecem apenas uma ideia geral sobre as tarefas e vocabulário e sobre como tornar o diálogo mais natural, não servindo, por exemplo, como dados para treinamento prático de sistemas de diálogo falado, que tem várias restrições sobre o diálogo.

O método *Mágico de Oz* (MCTEAR, 2002) é comumente usado para investigar como os usuários lidariam com um sistema computacional de diálogo falado. Nele, um humano simula o papel do sistema automatizado ou de partes dele, provendo respostas com voz sintetizada, induzindo o usuário a acreditar que está interagindo de fato com um computador. Esta técnica tem duas vantagens. A primeira é que ele permite que o comportamento do sistema seja avaliado antes que recursos significativos tenham sido gastos na construção do sistema. A outra é que ele permite um desenvolvimento interativo em que, aos poucos, componentes ou ações simuladas vão sendo substituídas por partes automatizadas. A desvantagem é exatamente a dificuldade para um humano passar por um computador, antecipando as limitações que o sistema irá ter.

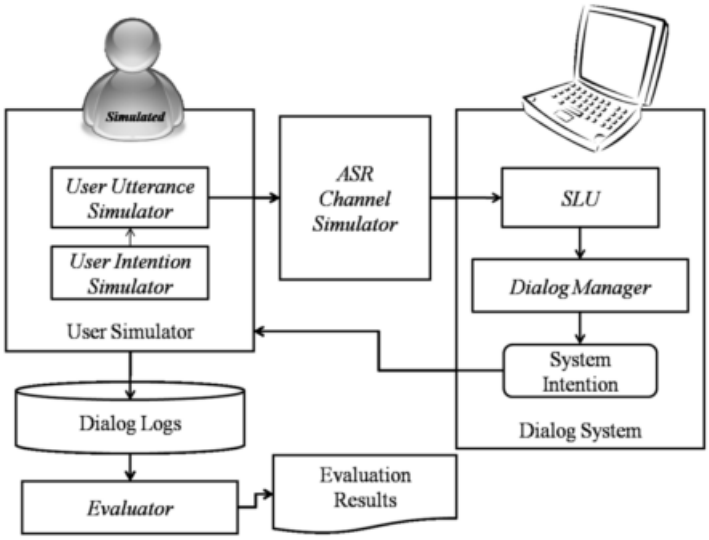
Uma técnica interativa para construção do sistema é o método chamado de *sistema em loop* (MCTEAR, 2002). Nela, versões com funcionalidades limitadas do sistema são utilizadas para coletar dados, a serem utilizados para aperfeiçoamento de novas versões, que são por sua vez utilizadas para coletar novos dados e assim sucessivamente até que o sistema esteja completo. Este método costuma ser trabalhado conjuntamente com o método Mágico de Oz. Por exemplo, uma versão inicial do sistema utilizada para coletar dados pode incluir apenas versões iniciais de componentes de reconhecimento e síntese de voz, enquanto os demais módulos são simulados por um humano.

Existem várias métricas para avaliar um sistema de diálogo, a maioria delas específicas de um componente, como a precisão na detecção das palavras e sentenças, próprias para o reconhecedor de voz, e a precisão na detecção de conceitos, própria para o entendimento de linguagem falada. Para o sistema de

diálogo como um todo, podemos destacar (CHEONGJAE LEE et al., 2010; MCTEAR, 2002):

- Taxa de sucesso para a realização de cada tarefa;
- Tamanho médio do turno;
- Quantidade de turnos necessários para a realização de cada tarefa;
- Taxa de adequação contextual, isto é, proporção de turnos que são, de alguma forma, inapropriados para o contexto. Os turnos podem ser classificados como *totalmente inapropriado*, *apropriado*, *parcialmente inapropriado* e *incompreensível*, por exemplo;
- Taxa de turnos de correção: proporção de turnos em que o sistema e o usuário tratam erros, incluindo feedbacks e solicitações de *feedback*. Um diálogo com alta taxa de turnos de correção pode ter altos custos para adaptação do usuário por ser pouco natural

Figura 10 – Arquitetura geral de uma simulação de diálogo utilizando simulação de usuário.



Fonte: (CHEONGJAE LEE et al., 2010)

De forma geral, essas métricas são levantadas a partir da opinião dos usuários. Entretanto, nem sempre há alguém qualificado disponível e com o treinamento correto. Para evitar esses problemas, técnicas de simulação de usuário têm sido usadas de forma abrangente (THOMSON, 2009), e podem ser colocadas em três categorias: simulação de intenção, simulação de fala e simulação de erros.

A arquitetura geral de um sistema de simulação de usuário pode ser vista na Figura 10. O simulador de intenção (*user intention simulation* na figura) analisa o contexto do diálogo[7] e define a intenção do usuário. O simulador de fala (*user utterance simulator* na figura) constrói uma expressão em linguagem natural e a pronuncia. O simulador de reconhecimento de voz (*ASR channel simulator*) adiciona ruído à fala do usuário, isto é, adiciona erros ao reconhecimento de voz. Uma boa revisão das técnicas de simulação de usuário pode ser encontrada em Cheongjae Lee et al. (2010).

[7] Para entender o que faz parte do contexto do diálogo na simulação de usuário, seria necessária uma pesquisa das referências utilizadas por (CHEONGJAE LEE et al., 2010), o que está fora do escopo deste trabalho de mestrado.

4 REVISÃO DA LITERATURA: A ADAPTATIVIDADE

A *adaptatividade* é uma técnica formal utilizada para que dispositivos mais poderosos, em relação à capacidade de expressão e de computação, possam ser obtidos gradualmente a partir dos recursos oferecidos por um dispositivo mais simples, de forma que sejam capazes de, sem a interferência de qualquer agente externo, tomar a decisão de modificar seu próprio comportamento (JOSÉ NETO, 2007), em tempo de execução. O termo *dispositivo*, como utilizado neste capítulo, refere-se a alguma abstração formal, cujo comportamento é regido por um conjunto finito explícito de regras que especificam, para cada situação em que se encontre, sua nova situação e também as correspondentes alterações esperadas no conjunto de regras que o definem. Em um dispositivo não-adaptativo, o conjunto de regras é invariável durante toda a sua execução. Um dispositivo adaptativo contém um conjunto especial de regras, que ativam, quando executadas, ações adaptativas, as quais por sua vez modificam o conjunto original de regras através de adições, alterações e remoções de regras. A camada composta desse conjunto de ações adaptativas é chamada de *camada adaptativa*; aquela composta das regras originais tem o nome de *camada subjacente*. Uma camada é dita *adaptativa de segundo nível* quando contém regras que alteram uma camada subjacente que também é adaptativa.

A adaptatividade foi primeiramente apresentada e utilizada na definição de autômatos adaptativos (JOSÉ NETO, 1993). Posteriormente, foi utilizada também em vários outros formalismos, tais como gramáticas (BRAVO, 2004), árvores e tabelas de decisão (PISTORI; JOSÉ NETO; PEREIRA, 2006; TCHEMRA, 2009) e *statecharts* (JOSÉ NETO; ALMEIDA JUNIOR; SANTOS, 1998). Uma formulação da adaptatividade, que não dependa do tipo de formalismo não adaptativo subjacente, foi feita primeiramente em José Neto (2001).

Os formalismos subjacentes dos dispositivos adaptativos podem ser classificados em variadas categorias, conforme sua forma de operação, algumas mais relevantes que outras para o gerenciamento de diálogo. Entre outras, têm-se as seguintes:

- Dispositivos de reconhecimento, da classe dos autômatos, baseados na sucessão de mudanças de estados;
- Dispositivos de geração, da classe das gramáticas;
- Dispositivos para a representação de sistemas assíncronos, tais como *statecharts*;

- Dispositivos estocásticos, como as cadeias de Markov, capazes de representar fenômenos aleatórios;
- Dispositivos de auxílio à tomada de decisão, como tabelas e árvores de decisão;
- Dispositivos de processamento, como as linguagens para a codificação de programas adaptativos.

Uma visão ampla acerca da adaptatividade encontra-se em José Neto (2007), e uma discussão centrada na utilização dos dispositivos adaptativos no gerenciamento do diálogo é feita em Alfenas et al. (2013). Neste capítulo, faz-se uma descrição detalhada de um dispositivo adaptativo, o Sistema de Markov Adaptativo, a fim de exemplificar detalhadamente um formalismo adaptativo que pode ter aplicações no gerenciamento de diálogo. Introduz-se também outros dois dispositivos de interesse de forma resumida, os autômatos adaptativos e as tabelas de decisão adaptativas.

4.1 Sistema de Markov Adaptativo

Um sistema de Markov adaptativo é um tipo de dispositivo formado por um conjunto de máquinas de Markov adaptativas (BASSETO, 2000) e um conjunto de regras especiais que permitem que as máquinas se relacionem. Uma máquina de Markov adaptativa é um dispositivo adaptativo cuja camada subjacente é formada por uma cadeia de Markov. Este dispositivo já foi utilizado com sucesso na geração de música em tempo real através do sistema LASSUS (BASSETO, 2000). Uma plataforma aberta de desenvolvimento de sistemas de Markov adaptativos, o *MMAdapt*, foi apresentada em Alfenas et al. (2012).

A formulação aqui apresentada é baseada neste último texto. A partir da definição básica de cadeias de Markov, vão sendo definidos dispositivos cada vez mais complexos, baseados no dispositivo imediatamente anterior, até que possamos definir um Sistema de Markov Adaptativo. Ao final, discute-se como estes dispositivos podem ser aplicados.

4.1.2 Cadeias de Markov

Define-se uma cadeia de Markov como uma tripla

$$K = (Q, q_0, T) \quad (4)$$

em que Q é um conjunto finito com n estados, q_0 é o estado inicial da rede e T é um conjunto de transições entre estados. Cada transição γ_{ij} é caracterizada como uma tripla

$$\gamma_{ij} = (q_i, q_j, \rho_{ij}) \quad (5)$$

em que ρ_{ij} é a probabilidade do estado q_j ser atingido estando em q_i. Observe que cada transição γ_{ij} é única em T, isto é, não pode haver duas transições diferentes partindo de um mesmo estado q_j e chegando a q_j. Além disso, seja Γ_q o conjunto de todas as transições que se originam em um estado q. Então:

$$\forall q \in Q, \sum_{\gamma \in \Gamma_q} \rho_\gamma = 1 \quad (6)$$

isto é, a soma das probabilidades de todas as transições iniciando em q é exatamente um, para todo q pertencente a Q.

A definição dada por (4), (5) e (6) garante que a probabilidade de um dado estado ser atingido depende exclusivamente do estado corrente da rede, isto é, a cadeia de Markov aqui definida é de primeira ordem. Pode-se provar também que, partindo-se do estado inicial q_0 e após um número suficientemente grande de iterações da rede, a probabilidade da rede encontrar-se em um de seus n estados é constante (HOWARD, 1971). Cabe ressaltar uma diferença em relação à representação utilizada na definição original (HOWARD, 1971): enquanto aqui T é um conjunto de transições, como na definição de autômatos, o texto original o apresenta como uma matriz n x n, com as mesmas propriedades. Desta forma, uma transição com ρ igual a zero é interpretada diferentemente de uma transição inexistente, o que será importante para a definição das ações elementares de consulta das máquinas de Markov adaptativas.

4.1.3 Máquinas de Markov

Uma máquina de Markov M é definida pela quádrupla

$$M = (Q, \Psi, T, q_0) \quad (7)$$

em que Q, T e q_0 são como definidos anteriormente para cadeias de Markov e Ψ é o alfabeto de saída. A expressão (5), então, dá lugar a

$$\gamma_{ij} = (q_i, q_j, \rho_{ij}, p_{ij}) \quad (8)$$

como definição de transição. Quando γ_{ij} é acionada, o símbolo $p_{ij} \in \Psi \cup \{\varepsilon\}$ é inserido na cadeia de saída de *M*.

Desta forma, é possível utilizar *M* como um dispositivo gerador de linguagem; a linguagem formada pelas possíveis cadeias de saída de *M* é regular. É possível mostrar, também, que o mecanismo de geração de linguagens por meio da máquina de Markov é formalmente equivalente ao das gramáticas lineares (BASSETO, 2000).

4.1.4 Máquina de Markov Adaptativa

Uma máquina de Markov adaptativa *M* é definida por uma quíntupla

$$M = (Q, \Psi, T, q_0, F) \quad (9)$$

em que Q, Ψ, T, e q_0 são como definidos anteriormente para a máquina de Markov não adaptativa e F é um conjunto de funções adaptativas. Toda função adaptativa pertencente a F pode ser definida como uma quádrupla

$$f = (\Lambda, V, G, C) \quad (10)$$

em que:

- Λ é uma sequência de m parâmetros formais (λ_1, λ_2, ..., λ_m);
- V é um conjunto de identificadores de variáveis {v_1, v_2, ..., v_y}, cujos valores são desconhecidos no instante de chamada de f mas que uma vez preenchidos terão seus valores preservados durante toda a execução da função;
- G é um conjunto de identificadores de geradores {g^*_1, g^*_2,..., g^*_z}, variáveis especiais que são preenchidas com novos valores, nunca antes utilizados pelo autômato, a cada vez que a função é chamada;
- C é uma sequência de ações adaptativas elementares executadas em f. Ressalta-se que a imposição de uma ordem nas ações não faz parte da definição original do formalismo adaptativo.

A expressão (8) é substituída por sua forma final

$$\gamma_{ij} = (q_i, q_j, \rho_{ij}, p_{ij}, a_{ij}) \quad (11)$$

como definição de transição, em que q_i, q_j, ρ_{ij} e p_{ij} são como definidos anteriormente e a_{ij} é uma ação adaptativa da forma

$$a_{ij} = f(\omega_1, \omega_2, ..., \omega_m) \cup \{\varepsilon\} \quad (12)$$

sendo f uma função adaptativa pertencente a F e (ω_1, ω_2, ..., ω_m) uma lista de m argumentos que correspondem posicionalmente à lista de parâmetros Λ declaradas para f. {ε} representa a cadeia vazia, isto é, é possível que uma transição não tenha ação adaptativa associada.

Definidos desta forma, o procedimento de mudança de estado de uma máquina de Markov adaptativa M pode ser descrito como segue:

1. Obtém-se a lista de transições possíveis Γ_{qi} a partir do estado ativo q_i em M;
2. Escolhe-se de forma aleatória uma única transição γ_{ij} a disparar, utilizando-se as probabilidades ρ_{ik} das transições;
3. A máquina muda para o estado q_j;
4. O símbolo p_{ij} é inserido na cadeia de saída de M;
5. Se $a_{ij} \neq \varepsilon$, então a função adaptativa f é chamada com os argumentos ω_1, ω_2, ..., ω_m, realizando a ação adaptativa sobre M. Se não, nada é feito.

As ações adaptativas elementares utilizadas em C podem ser dos seguintes tipos:

- Consulta de transição: busca transições conforme um filtro recebido como parâmetro e as armazena em uma variável v. A imposição de uma sequência na execução das ações elementares de f remove o não determinismo, pois diferentes sequências de preenchimento de variáveis de retorno poderiam levar a diferentes resultados nas ações de consultas seguintes em f que também usam v;
- Adição de transição: cria uma nova transição entre dois estados existentes. Se já existir uma transição entre eles, ela é substituída;
- Remoção de transições: remove uma ou mais transições recebidas por parâmetro.

A partir desses três tipos de ações elementares é possível construir outras ações que aumentam a expressividade de construção de uma máquina de Markov adaptativa, sem, entretanto, aumentar seu poder de computação. Podemos citar, por exemplo, uma ação adaptativa que executa outra ação dependendo do resultado da comparação de valor entre duas variáveis (ALFENAS et al., 2012). A linguagem *L(M)* gerada por uma máquina de Markov adaptativa é sensível ao contexto (BASSETO, 2000).

Quando se executa uma ação de alteração de probabilidade ou remoção ou inserção com $\rho > 0$ para uma transição γ_{ij}, é necessário reajustar as probabilidades para as

demais transições pertencentes a Γ_q de tal forma que (6) continue válida. Seja ρ' a nova probabilidade (considere ρ' igual a zero no caso de remoção) e ρ_{ij} a probabilidade antiga (considere ρ_{ij} igual a zero no caso de inserção). Então

$$\forall \gamma_{ik} \in \Gamma_q, k \neq j, \rho_{ik} = \rho_{ik} \bullet \frac{(1-\rho')}{1-\rho_{ij}} \qquad (13)$$

4.1.5 Sistema de Markov Adaptativo

Define-se um Sistema de Markov Adaptativo como um conjunto de máquinas adaptativas de Markov que podem se relacionar através de novas ações adaptativas, introduzidas a seguir. Assim, pode-se estabelecer uma relação de escopo sobre as ações adaptativas definidas para cada máquina, classificando-as em: ações que modifiquem apenas a topologia local da máquina e ações que podem interferir no comportamento de outras máquinas pertencentes ao sistema. Desta forma, cada máquina M^k de um sistema de Markov adaptativo é definida como uma quíntupla:

$$M^\kappa = (Q^\kappa, \Sigma, T^\kappa, q_0{}^\kappa, F^\kappa) \qquad (14)$$

em que Σ é o alfabeto de saída, comum a todas as máquinas do sistema, e Q^κ, T^κ, $q_0{}^\kappa$, F^κ são como na definição de Q, T, q_0 e F, respectivamente.

Além dos tipos de ações elementares definidos para uma máquina de Markov adaptativa, as máquinas pertencentes a um Sistema de Markov Adaptativo podem executar outros tipos, que operam sobre as outras máquinas do sistema. São eles:

- Dos mesmos tipos definidos para M, porém operando sobre outras máquinas do sistema;
- Desativa M^w, $w \neq \kappa$. Desabilita a máquina M^w. Uma máquina desabilitada permanece em seu estado corrente, não acionando qualquer transição nem executando qualquer ação adaptativa, até que seja novamente habilitada;
- Ativa M^w, $w \neq \kappa$. Habilita a máquina M^w;
- Consulta estado de M^w, $w \neq \kappa$. Consulta o estado corrente de M^w e o retorna.

4.1.6 Aplicação dos Sistemas de Markov Adaptativos

Os sistemas de Markov adaptativos foram aplicados com sucesso na síntese automática de música, como demonstrado em dois trabalhos distintos, Basseto (2000)

e Alfenas et al. (2012). No primeiro caso, do sistema LASSUS, a composição gerada pertence ao estilo barroco, a quatro vozes. No segundo caso, o sistema Markovianas, gera composições de rock, com guitarra, bateria e baixo. Ele foi criado sobre a plataforma MMAdapt, que permite construir máquinas de Markov a partir de arquivos de configuração (XML). Nos dois casos, as diversas vozes ou instrumentos são definidos a partir de máquinas específicas controladas por uma máquina compositora, cujas ações adaptativas alteram o comportamento das demais. O MMAdapt permite estender o seu comportamento através de uma interface de programação em linguagem Java. O desenvolvedor pode alterar tanto o método de interpretação da cadeia de saída quanto o mecanismo que controla o intervalo de tempo entre os acionamentos das transições.

4.2 Outros dispositivos adaptativos de interesse

Os dispositivos a seguir são descritos de forma breve. Eles possuem certas características importantes para o gerenciamento de diálogo.

4.2.1 Autômatos adaptativos

Os autômatos adaptativos são dispositivos com poder computacional equivalente ao das Máquinas de Turing, capazes de reconhecer cadeias pertencentes às linguagens sensíveis ao contexto. Os autômatos adaptativos apresentados em José Neto (1993) e José Neto (1994) possuem como dispositivo subjacente autômatos de pilha estruturado (JOSÉ NETO, 1987). Uma regra deste dispositivo adaptativo pode ter até duas ações adaptativas associadas, uma anterior e outra posterior à mudança de estado da transição. As ações adaptativas elementares são dos mesmos tipos definidos para máquinas de Markov adaptativas: ações de consulta, inserção e remoção. O autômato adaptativo pode ser determinístico ou não determinístico. Ele é considerado determinístico se e somente se, a cada passo de sua operação, o dispositivo adjacente for determinístico (CASTRO JUNIOR; JOSÉ NETO; PISTORI, 2007). O dispositivo subjacente é determinístico se há garantia de que em cada passo de execução exista apenas uma transição que possa ser utilizada (SHIBATA, 2008).

Um transdutor adaptativo é uma variação do autômato adaptativo, cujas transições podem definir um símbolo de saída. Quando uma dessas transições é disparada, o símbolo de saída nela definido é colocado em uma cadeia de saída.

Os autômatos adaptativos já têm sido empregados no processamento de linguagem natural. Podemos destacar o trabalho de Shibata (2008), que aplica com sucesso os autômatos adaptativos à tradução grafema-fonema para o português brasileiro, que é uma das principais etapas da síntese de voz.

4.2.2 Tabelas de decisão adaptativas

As tabelas de decisão adaptativas (JOSÉ NETO, 2001; STANGE; NETO, 2011) são dispositivos adaptativos que têm como dispositivo subjacente, como se poderia supor, tabelas de decisão, dispositivos bastante usados em aplicações de tomada de decisão. Uma tabela de decisão tradicional é composta de regras que relacionam condições a ações a serem executadas. Uma tabela de decisão adaptativa, conforme pode ser visto na Figura 11, estende o dispositivo original de duas formas:

- Permite especificar um conjunto de funções adaptativas, utilizando como base os mesmos três tipos de ações elementares de consulta remoção e inserção, utilizados nos dispositivos especificados anteriormente.
- Permite associar uma chamada de função adaptativa anterior e de uma chamada de função adaptativa posterior à execução da regra.

Figura 11 – Estrutura geral de uma tabela de decisão adaptativa

<table>
<tr><td colspan="2"></td><td>Colunas das funções adaptativas</td><td>Colunas das regras normais</td></tr>
<tr><td></td><td>Cabeçalhos</td><td rowspan="11">Declaração das funções adaptativas</td><td rowspan="11">Declaração das regras normais</td></tr>
<tr><td rowspan="3">Condições</td><td>Critério 1</td></tr>
<tr><td>Critério 2</td></tr>
<tr><td>...</td></tr>
<tr><td rowspan="3">Ações</td><td>Ação 1</td></tr>
<tr><td>Ação 2</td></tr>
<tr><td>...</td></tr>
<tr><td rowspan="5">Funções adaptativas a chamar</td><td>Nome</td></tr>
<tr><td>Anterior ou Posterior</td></tr>
<tr><td>Parâmetro 1</td></tr>
<tr><td>Parâmetro 2</td></tr>
<tr><td>...</td></tr>
</table>

Embora esteja fora do escopo deste trabalho, é importante mencionar a existência das tabelas de decisão multicritério (TCHEMRA, 2009), uma extensão da proposta original, que permitem a modelagem de problemas de decisão mais complexos e semiestruturados, incorpora algoritmos clássicos de métodos multicritérios e fornece ao decisor condições para expressar seus julgamentos acerca dos critérios.

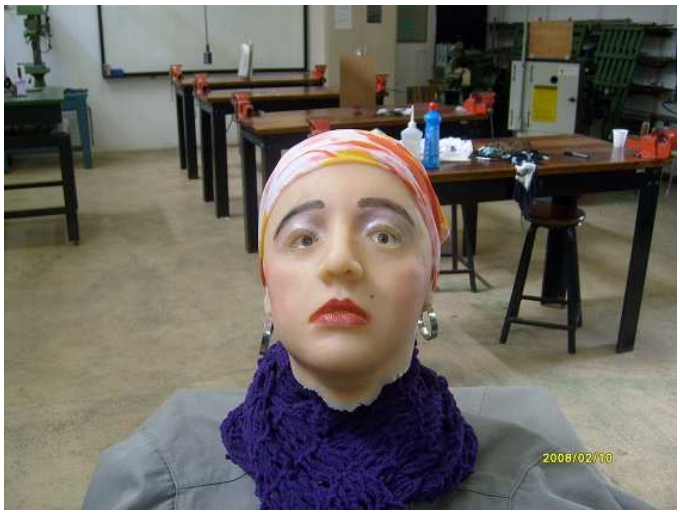
5 A ARQUITETURA DO SISTEMA

O gerenciador de diálogo (doravante chamado apenas de GD) proposto neste trabalho faz parte de um sistema de diálogo falado, que possui outros componentes. Este capítulo descreve, em alto nível, este sistema e sua arquitetura, e detalha o modelo de entradas e saídas que permite a interação entre o GD e demais componentes.

5.1 O robô sociável Minerva

O GD proposto é parte do sistema de diálogo do robô sociável Minerva, cuja foto pode ser vista na Figura 12. É um robô humanoide parcial, que possui apenas pescoço e cabeça. A Minerva, portanto, não é capaz de se locomover sozinha ou de gesticular. Também não tem olfato e, na fase atual de seu projeto, nada possui que seja equivalente ao tato. As capacidades de que é dotada são:

Figura 12 – A Minerva, em fotografia de 2010



- Reconhecimento de voz: capacidade diretamente relacionada ao GD através dos processos de entendimento de linguagem natural.
- Síntese de voz: todas as falas elucubradas pela Minerva são consequências diretas das saídas do GD.
- Visão e movimento dos olhos e do pescoço: na maior parte do tempo, o GD não envia comandos de controle diretamente aos módulos de visão e de movimento dos olhos e do pescoço. Pode-se dizer que o GD não tem

conhecimento de tais capacidades na maioria das situações. Entretanto, se alguém pede ao robô para olhar para algo ou para alguma direção, cabe ao GD decidir o que fazer, naturalmente enviando algum sinal para que a visão e os olhos sejam direcionados ao local solicitado pelo interlocutor.

- Movimento dos músculos da face: as mesmas observações feitas para a visão se aplicam ao movimento dos músculos da face.

Posto desta forma, o comportamento da Minerva não é diferente daquele de um agente virtual representado por um rosto e que se comunica com o interlocutor através de um computador com câmera, caixas de som e microfone. Estas habilidades foram analisadas durante o desenvolvimento do GD de forma a determinar o conjunto de observações que podem ser feitas sobre o mundo e o conjunto de ações que o GD pode usar como saída.

O sistema pode estar ciente de necessidades reais do sistema robótico, como aquelas decorrentes de energia próxima do fim e de rotinas de manutenção do sistema decorrente do uso diário, como atualizações ou reorganização da memória permanente. O GD proposto neste trabalho considera que em tais situações lhe será enviado um aviso pelos módulos de monitoração, para que o encerramento do diálogo seja feito de forma adequada.

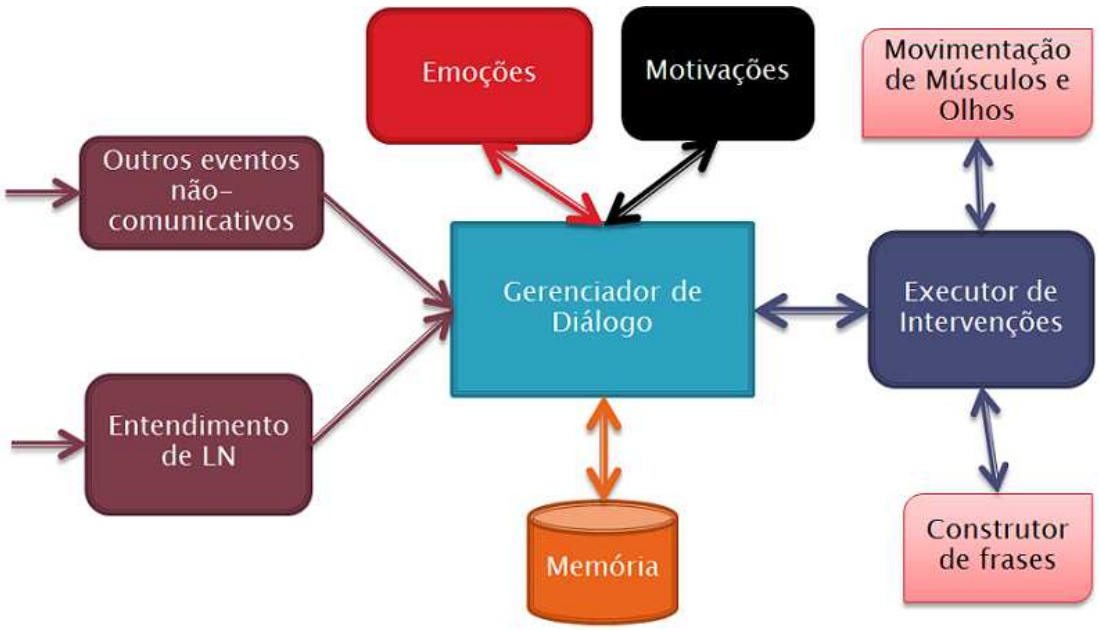
A Minerva foi criada com o propósito de ter interações voltadas às coisas em geral, como conversações sobre fatos recentes e coisas que as pessoas gostam, desgostam ou possuem, em vez de suporte à execução de tarefas (ALFENAS; PEREIRA-BARRETTO, 2012). Além disso, o projeto do sistema deve buscar métodos que proporcionem à Minerva um comportamento natural, isto é, ela deve se comportar como um ser humano. Utilizando termos coloquiais, a intenção do projeto é criar um robô capaz de “bater papo”. Entretanto, para os fins deste trabalho de mestrado, esta linha não é adequada, pois dificulta a comparação com trabalhos existentes. Na maioria deles, os sistemas de diálogo são utilizados para fornecer informações. Um exemplo bastante comum nas propostas é um sistema de informação a turistas sobre bares, restaurantes ou museus. Outros exemplos incluem sistemas de fornecimento de rotas de ônibus, de navegação veicular, de suporte à equipe de manutenção de equipamentos e de comunicação em grupo de robôs caça-tesouro, todos produzidos sobre a plataforma do Ravenclaw (BOHUS; RUDNICKY, 2009). Os sistemas baseados em POMDP, além de utilizados em sistemas para fornecimento de informação, também tem sido utilizados em sistemas de controle de equipamentos de

serviços domésticos (YOUNG et al., 2013). Desta forma, como detalhado no capítulo de resultados, o GD proposto é utilizado na execução de uma tarefa de vendas.

5.2 Visão de arquitetura do sistema de diálogo falado da Minerva

Do ponto de vista do GD, a arquitetura geral do sistema da Minerva pode ser vista na Figura 13. Embora existam outros componentes na arquitetura além dos exibidos, estes não têm relação direta com o GD. Além disso, os componentes exibidos podem ser, quando implementados, divididos em vários; a figura apenas ilustra as responsabilidades e, como se verá a seguir, o objetivo é que os mecanismos de arquitetura permitam desacoplamento máximo.

Figura 13 – Arquitetura de sistema da Minerva, do ponto de vista do GD



5.2.1 Descrição geral dos componentes

Os dois componentes à esquerda na Figura 13, “Entendimento de LN” e “Outros eventos não comunicativos”, são responsáveis pelas entradas ativas que o GD recebe. O primeiro envia hipóteses sobre as observações feitas quanto às falas e gestos do interlocutor. O segundo envia hipóteses sobre informações com propósito não comunicativo, como identificação do interlocutor, distância do interlocutor e o sinal para solicitação de encerramento gerado pela própria Minerva.

A memória é o componente responsável pelo armazenamento de crenças, episódios e decisões do GD. É utilizado também para consultas, que são importantes para a tomada de decisão. É bastante conveniente que os formatos de dados utilizados tanto na memória quanto no GD sejam o mesmo.

Os dois componentes acima do GD, “Emoções” e “Motivações”, influenciam as decisões tomadas de forma passiva. O componente de emoções é responsável por informar o estado emocional do interlocutor. O componente de motivações poderia ser utilizado no GD como suporte para determinação dos objetivos do robô durante a conversa.

Finalmente, o último componente de interesse é o executor de intervenções, responsável pela fissão multimodal. Ele deve sincronizar o controle dos mecanismos físicos utilizados na movimentação do pescoço, dos olhos, da boca e dos demais músculos faciais com a geração e a síntese da fala, de forma que o GD não tenha que controlar tais mecanismos.

5.2.2 Desacoplamento

Como dito acima, a arquitetura deve prover um método para o desacoplamento entre os componentes. Dois mecanismos básicos são utilizados para prover tal desacoplamento: (i) declaração obrigatória de uma interface para cada tipo de dados trocados entre os componentes e (ii) um mecanismo de quadro negro[8] para comunicação de hipóteses entre os componentes.

O quadro negro é um *singleton*[9] em que ficam disponíveis todas as entradas e saídas compartilhadas entre os componentes do sistema. Inicialmente, os componentes registram no quadro negro quais são os tipos de eventos que devem receber; então, quando um novo dado é colocado no quadro negro, um evento é disparado, e todos os componentes que se registraram para o tipo de evento selecionado são avisados. Os tipos de eventos incluem, entre outros, os mesmos tipos utilizados em arquiteturas incrementais, isto é, eventos de atualização de hipóteses referentes a um determinado

[8] Do padrão de arquitetura *blackboard* (DEUGO; WEISS; KENDALL, 2001).

[9] Singleton é um padrão para garantir que uma única instância de uma classe existe em qualquer instante (GAMA et al., 1994).

tipo de dado, de revogação de uma hipótese específica e de compromisso com uma hipótese específica.

5.3 Entradas do gerenciador de diálogos

Nesta seção, especificamos o modelo de entrada de dados do GD, o que compreende a modelagem das hipóteses, dos segmentos funcionais, dos gestos, dos atos dialogais, do conteúdo semântico e das relações funcionais, retóricas e de *feedback*, além de entradas referentes a dados não comunicativos. Este modelo deve ser compartilhado por toda a arquitetura da Minerva e é derivado dos modelos observados nos modelos pesquisados na literatura.

5.3.1 Modelo de hipótese múltiplas sobre ações faladas

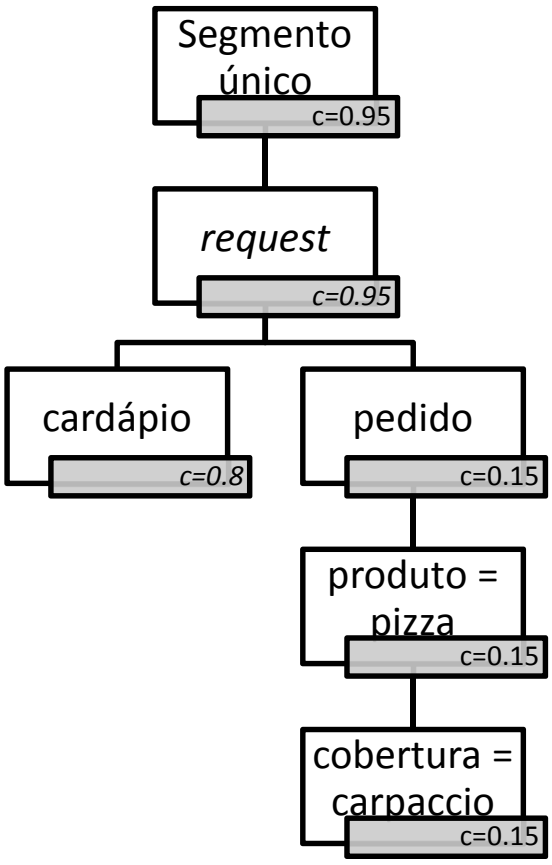
A estrutura utilizada neste projeto para o modelo de funções comunicativas e de segmentação funcional é baseada na norma ISO 24617-2 e pode ser dada pelas definições abaixo:

1. A ação do interlocutor é dividida em segmentos funcionais.
2. Um segmento funcional possui um conjunto de atos dialogais. No máximo um ato dialogal é permitido por dimensão comunicativa.
3. Um ato dialogal pode possuir um conjunto de itens semânticos relacionados cuja interpretação depende da função comunicativa do ato. Algumas funções comunicativas não necessitam de um componente semântico, como o *stalling* (protelamento – ver descrição na Figura 19).
4. O ato dialogal pode possuir relações funcionais, retóricas e de *feedback* com outros atos dialogais.
5. Um segmento funcional possui qualificadores de emoção. Esta regra é diferente do modelo original do ISO 24617-2, em que os qualificadores de emoção são aplicados no próprio ato dialogal.

Este modelo tem dois problemas. Primeiro, ele não considera que existem diversas hipóteses sobre a ação do usuário. Segundo, ele não é suficiente para um sistema multimodal, em que o usuário pode gesticular simultaneamente à fala. Desta forma, algumas definições adicionais são necessárias.

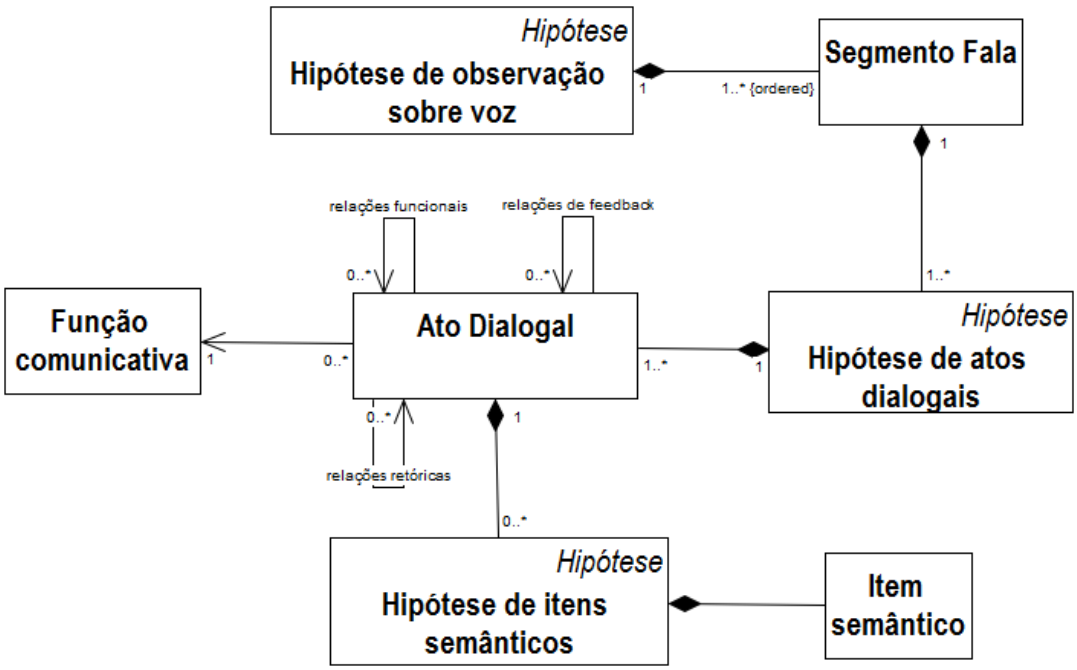
A entrada do GD deve compreender tanto o modelo de intervenções baseadas em voz quando o modelo básico de hipóteses. O modelo de hipóteses determina que cada ação a_u do interlocutor é recebida pelo sistema como um conjunto de n hipóteses, cada uma na forma de um par (observação o_i, grau de confiança c_i), $1 \leq i \leq n$. Além disso, a representação do conjunto de hipóteses deve seguir a modelagem da própria ação, de forma que o processo de tomada de decisão possa capturar a origem da incerteza, se assim for necessário. Por exemplo, suponha que o GD tenha recebido duas hipóteses sobre a_u, “Gostaria de ver o cardápio” com 80% de probabilidade e “Gostaria de comer carpaccio” com 15%. Ambas as hipóteses são formadas por um único segmento funcional contendo um único ato dialogal de requisição, isto é, referente à função comunicativa *request*. A diferença acontece apenas no conteúdo semântico. Se o modelo de hipóteses seguir a estrutura definida pelo próprio modelo para intervenções baseadas em voz, é possível capturar a forma como cada nível (segmentação funcional, ato dialogal e itens semânticos) gera incerteza na hipótese de entrada.

Figura 14 – Exemplo com duas hipóteses observadas sobre a mesma ação do usuário



Em um primeiro nível, as probabilidades devem ser distribuídas entre as diferentes hipóteses de segmentação funcional. O segundo nível deve dividir as probabilidades entre as diversas hipóteses sobre os atos dialogais. O terceiro nível deve seguir a estrutura dos itens semânticos do ato dialogal de cada hipótese. Seguindo o exemplo anterior, o componente de entendimento de linguagem natural deve gerar como entrada para o GD a estrutura exibida na Figura 14. A quebra entre as hipóteses acontece apenas no nível semântico, que diferencia cardápio e carpaccio, sendo que este último foi entendido como a solicitação de um pedido de pizza com cobertura de carpaccios, o que reflete na estrutura dos níveis seguintes.

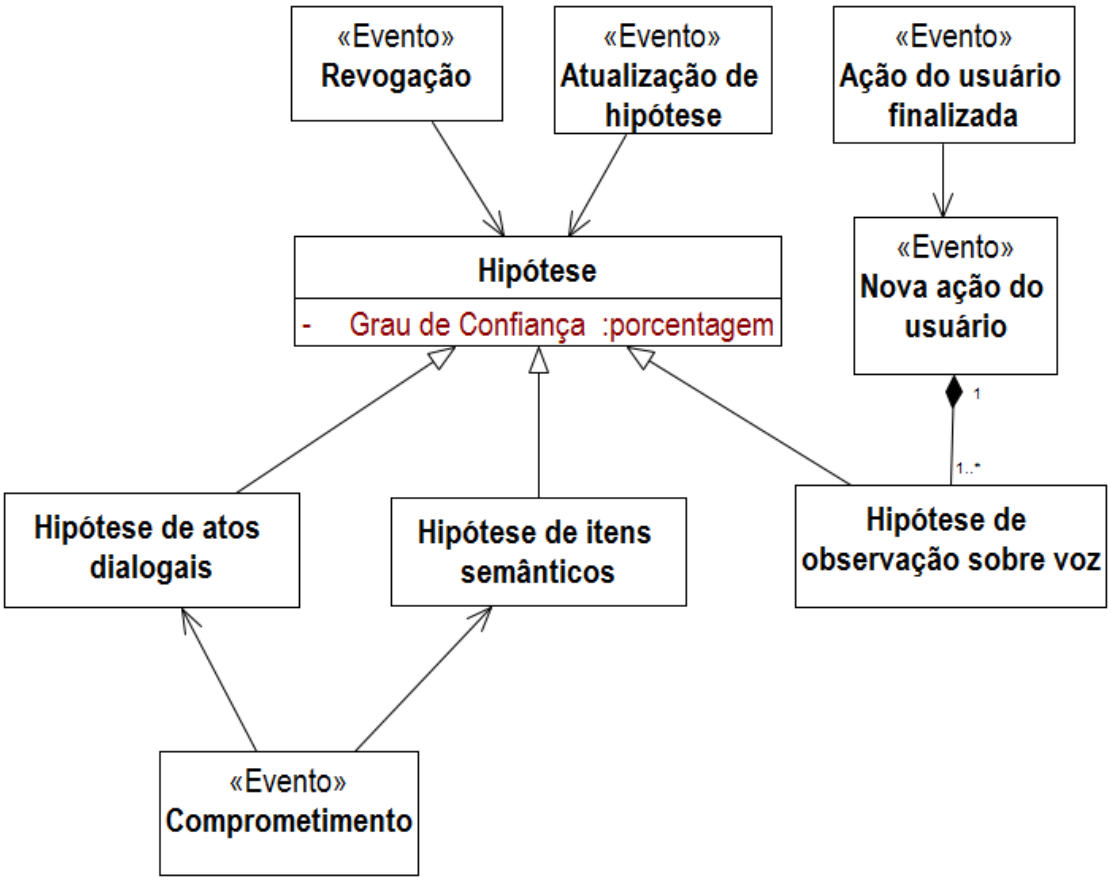
Figura 15 – Modelo conceitual para as entradas do GD considerando voz e múltiplas hipóteses



O modelo conceitual desta proposta pode ser visto na Figura 15. Um aspecto importante dele é que atos dialogais referentes à mesma função comunicativa, mas com relações diferentes, determinam hipóteses de atos dialogais diferentes. O problema deste modelo hierárquico é que ele replica os mesmos atos dialogais e itens semânticos em hierarquias de hipóteses distintas, o que dificulta relacionamentos entre atos dialogais de diferentes turnos. Entretanto, como é detalhado no capítulo 7, o modelo de gerenciamento de diálogo proposto não mantém múltiplas hipóteses sobre ações passadas do usuário, apenas da ação atual.

É necessário também definir como funcionam as atualizações de hipóteses já existentes, no caso de um tipo de dados que suporte o modo incremental. Um exemplo de situação em que os incrementos são relevantes para a tomada de decisão acontece no *feedback* simultâneo e natural do usuário. Se ‘o usuário começar a falar enquanto o robô já estiver falando, três interpretações são possíveis: (i) o usuário quer fazer algum tipo de reparo e a expectativa é que o sistema deve parar de falar para escutá-lo; (ii) o usuário está dando um *feedback* positivo, como “ok”, “u-hum”, e o sistema não deve parar de falar, pelo contrário, o natural é mantê-la; (iii) o usuário está dando um *feedback* negativo e o sistema não precisa terminar o turno, mas eventualmente perguntar o que há de errado ou tentar inferir o que pode estar errado e corrigir o turno. Portanto, o sistema não deve parar de falar de imediato quando o usuário começa a falar simultaneamente, mas reavaliar a ação do usuário de forma incremental, através de eventos de atualização recebidos pelo GD, que devem especificar a qual ação e a qual hipótese se relacionam.

Figura 16 - Modelo conceitual de hipóteses sobre as ações do usuário e eventos de entrada do GD



O diagrama exibido na Figura 16 mostra os tipos de eventos e as relações destes com os tipos de hipótese já apresentados. A atualização, a revogação e o compromisso podem ser compreendidos como tipos de eventos de atualização dos dados no quadro negro, em que os dados são de tipos que interessam ao GD. Além dos três mencionados, a arquitetura define dois outros tipos de eventos de interesse do GD:

- Nova ação do usuário: informa ao GD que o usuário iniciou uma ação. O seu uso mais importante é para informar que o usuário começou a falar.
- Ação do usuário finalizada: informa ao GD que o usuário terminou de agir, o que pode indicar que o turno está disponível.

Na inicialização do sistema, o GD registra no quadro negro que deseja ser informado sobre estes cinco tipos de eventos, relativos à fala e ao gesto e já analisados semanticamente. Nota-se que é possível, embora não desejável, que o GD receba eventos sobre uma ação do usuário mesmo após ela ter sido finalizada.

5.3.2 Incorporação de gestos ao modelo

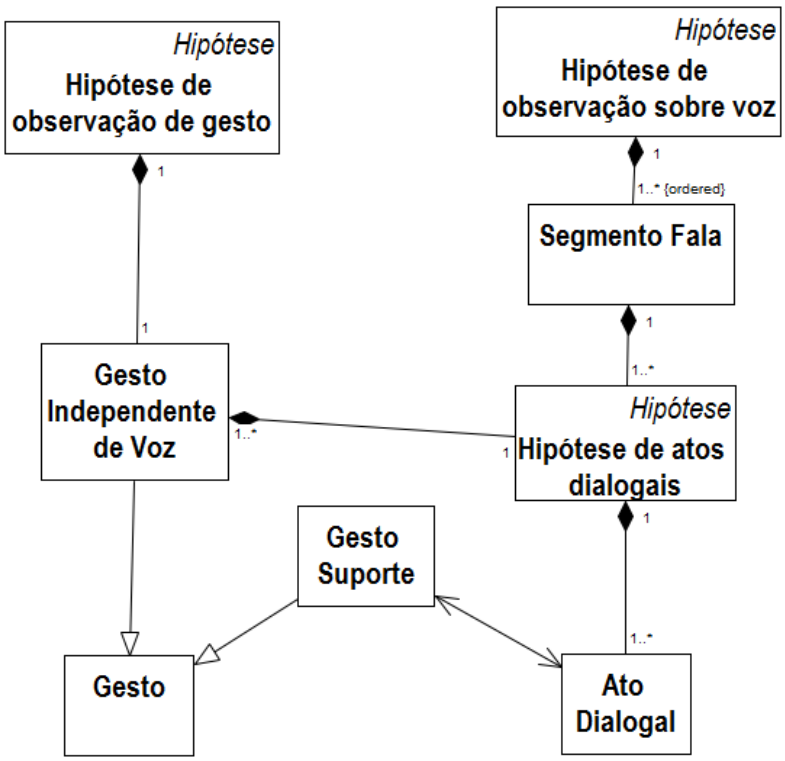
Resta apenas incorporar no modelo aspectos multimodais da comunicação não associados à voz, isto é, relacionados aos gestos, que podem ser separados em dois grupos:

- Gestos de suporte à voz: são gestos cuja interpretação depende da fala simultânea ou que replicam informação dada também em canal de voz. O ato dialogal que é suportado tanto por voz quanto por um gesto deve estar relacionado a ambos e os eventos de entrada estarão relacionados a uma ação baseada em voz, não baseada em gesto.
- Gestos independentes de voz: são gestos que não tem relação direta com a fala do mesmo participante. É o caso de muitos gestos de *feedback* do usuário simultâneos à fala do robô. Neste caso, o evento de entrada estará ligado diretamente a hipóteses sobre o gesto.

A detecção da Minerva em relação a gestos é bastante limitada e esta questão não é tratada com a profundidade adequada nesta proposta. O projeto da Minerva prevê detecção apenas de expressões faciais e movimentos da cabeça do interlocutor, o que deve permitir identificar gestos de negação e de afirmação da cabeça. Isso não

quer dizer que outros gestos não seriam úteis em uma aplicação. A utilização de gestos manuais para controle do turno, por exemplo, seria bastante útil.

Figura 17 – Modelo de entrada do GD para gestos



A Figura 17 exibe o modelo de hipóteses atualizado para incorporação de gestos. A primeira observação importante sobre a proposta é que cada hipótese compreende apenas um gesto, ao invés de compreender uma sequência de gestos ligados por um mesmo movimento. Esse modelo tampouco diferencia as fases do gesto. Um modelo incremental de interpretação dos gestos pode se beneficiar da incorporação destes dois aspectos. Por exemplo, um evento de entrada poderia comunicar que existe um gesto em andamento, mas, ao também sinalizar que ainda não houve retorno para a posição de descanso das mãos, avisa o GD que ainda podem acontecer sinalizações de novos gestos dentro da mesma unidade de movimento. Entretanto, a proposta de projeto da Minerva é adequada para a captação dos gestos de negação e de afirmação feitos com a cabeça do interlocutor.

Figura 18 - Exemplos de atos dialogais de propósito geral, conforme norma ISO 24617-2

Função comunicativa	**Descrição**	**Itens relacionados**
Question	Pergunta. Ex: “Qual o preço da pizza de pepperoni?”	Identificação do conteúdo que deseja descobrir. Ex: pizza (nome=*pepperoni*, preço=?)
CheckQuestion	Pergunta que deveria ser respondida com sim ou não. Ex: “Você entrega em dez minutos?”	Identificação do conteúdo que se deseja descobrir. Ex: tempoEntregaProximoPedido <= 10
Inform	Ato de informação. Ex: “Quero a de pepperoni”	Conteúdo semântico relacionado.
Answer	Ato de informação que responde a uma pergunta.	Conteúdo semântico + a_S com *Question* sendo respondida
Suggestion	Diretiva feita pelo usuário sugerindo algo.	Conteúdo semântico.

5.3.3 Funções comunicativas suportadas

O GD deve suportar algumas funções comunicativas referentes à fala e a gestos. Exemplos de funções de propósito geral, isto é, que podem ser utilizadas em diversas dimensões, inclusive específicas do domínio de negócio, estão na Figura 18. Exemplos de atos dialogais de uma dimensão específica estão na tabela Figura 19.

5.3.4 Entradas não comunicativas

As entradas não comunicativas seguem o mesmo modelo básico de múltiplas hipóteses. Isto é, cada entrada não comunicativa é um par (observação *o*, grau de confiança *c*).

Os tipos de entradas não comunicativas considerados pelo sistema dependem da aplicação e das capacidades do sistema. Por exemplo, a identificação do usuário pode ser um evento não comunicativo, originado a partir da análise do timbre de voz ou de um sistema externo de autenticação biométrica, ou pode ser comunicado verbalmente através de funções comunicativas do próprio diálogo.

Desta forma, o modelo do gerenciador deveria permitir a declaração de novos tipos de entradas e ser robusto para que consiga fazer a tomada decisão com eles.

Figura 19 - Exemplos de atos dialogais de propósito específico, conforme norma ISO 24617-2

Função comunicativa	**Descrição**	**Itens relacionados**
Initial Greeting	Saudação inicial.	Cumprimento utilizado.
Return Greeting	Resposta de saudação.	Cumprimento utilizado + a_s de InitialGreeting.
Well-being check	Saudação onde se pergunta sobre o bem-estar do outro.	Cumprimento utilizado. Exemplo: “Tudo bem?”
Thanking	Agradecimento	Expressão utilizada.
Thanking downplay	Minimização do agradecimento. Exemplo: “De nada”	Expressão utilizada + a_s de Thanking.
Apology	Desculpas. Comumente utilizado em reparos.	Eventualmente a a_s pelo qual o sistema se desculpa.
Apology downplay	Minimização do ocorrido. Exemplo: “Tudo bem.”	a_s de *Apology*.
Stalling	O usuário repete o mesmo som durante um tempo, conseguindo mais tempo para formular a frase. Exemplo: “huuummmmm”	Irrelevante.
Contact Check	Verificação se o ouvinte está atento. Ex: “Minerva?”, “Alô?”	Expressão utilizada.
AutoPositive	*Feedback* dado pelo usuário quando ele entendeu a última intervenção do robô. Ex: “OK”, gesto com a cabeça.	a_s ao qual o feedback referencia.
AlloNegative	*Feedback* dado pelo usuário quando ele acredita que o robô não entendeu sua última fala. Ex: “Não”, gesto com a cabeça.	a_s ao qual o feedback referencia.

5.4 Modelo Semântico

Nesta dissertação, a principal diretriz de modelagem semântica foi a criação de uma estrutura básica bastante flexível, isto é, uma estrutura sobre a qual se pudesse descrever uma grande variedade de conhecimento. Por outro lado, este é um problema cuja solução foge do escopo deste trabalho. Portanto, seria suficiente, para a demonstração do método de gerenciamento de diálogo proposto, criar ou utilizar uma estrutura básica que viabilizasse a modelagem semântica das aplicações de exemplo.

Os trabalhos de gerenciamento de diálogo pesquisados não detalham o seu modelo semântico, normalmente se limitando a relatar a arquitetura em alto nível desta memória e, no caso dos trabalhos baseados em POMDP, como ela é utilizada para calcular o estado de crença do diálogo. Desta forma, o padrão de memória semântica a ser utilizado foi pesquisado a partir padrões de modelagem semântica de propósito mais geral, tais como o UNL (UCHIDA; ZHU, 2001) e o ConceptNet (SPEER; HAVASI, 2012), ao invés dos modelos já utilizados em sistemas de diálogo falado.

Nesta pesquisa, foram considerados alguns requisitos de relação entre a memória e o GD. O primeiro foi a utilização do mesmo modelo na memória semântica e nos itens semânticos dos atos dialogais, o que removeria a necessidade de um conversor de formatos. Apesar disso, a interpretação do item semântico deveria depender da função semântica do ato dialogal relacionado. Por exemplo, se o usuário disser “Eu não gosto de cebola”, isso deve ser traduzido em um ato dialogal de função *inform* e conteúdo semântico na forma (relação = ”não gosta de”, indivíduo agente = Interlocutor, classe objeto = cebola). Se o usuário perguntar “A pizza de calabresa tem cebola?”, isso deve ser traduzido em um ato dialogal de função *propositional question* (um tipo de pergunta cuja expectativa de resposta é sim ou não) com conteúdo semântico na forma (relação = ”tem parte”, individuo agente = ”pizza de calabreza”, classe objeto=”cebola”). Se o usuário disser “Quero uma de calabresa sem cebola”, isso deve ser traduzido em um ato dialogal de função *request* com conteúdo semântico na forma (ação = ”fazer pedido”, produto (nome=”pizza de calabresa”, restringido por (relação = ”não tem parte”, classe agente = ”pizza de calabreza”, classe objeto = ”cebola”)).

Outro requisito é que esta ligação do item semântico com as memórias semânticas e episódicas deve ser feita antes da recepção do evento pelo GD ainda durante a análise semântica, isto é, a ligação não é uma responsabilidade do GD. A memória semântica é formada por uma ontologia inicial, presente antes que ocorra o primeiro diálogo, informações armazenadas como decorrência das tomadas de decisão do GD e, eventualmente, informações inferidas e armazenadas após o diálogo. A memória episódica mantém o histórico completo do diálogo.

Seria importante também considerar a linguagem a ser utilizada na declaração do modelo. Seu poder expressivo deveria ser equivalente ao da lógica descritiva (KRÖTZSCH; SIMANCÍK; HORROCKS, 2013) ou, ao menos, ser capaz de permitir:

- a declaração de indivíduos, isto é, objetos ou instâncias de classes;

- a declaração de classes, isto é, agrupamento de indivíduos através de coleções enumeradas ou com características comuns;
- a declaração de relações entre os indivíduos e classes;
- a utilização de expressões lógicas complexas utilizando indivíduos, classes e relações, incluindo combinações de conjunções, disjunções, negações e operadores existenciais, de cardinalidade e um que relacione uma classe como subclasse de outra;
- a declaração de axiomas, isto é, expressões lógicas admitidas como verdadeiras no domínio da aplicação.

Além disso, é um diferencial importante a existência de ferramentas que auxiliem na definição e na utilização da ontologia, tais como um editor, um *reasoner* (software que auxilia a inferir consequências lógicas a partir dos axiomas no domínio da aplicação, incluindo verificar se o domínio tem alguma inconsistência, isto é, se existem axiomas que não poderiam ser simultaneamente verdadeiros) e uma interface de programação de aplicativos.

Todas esses requisitos são atendidos pela linguagem escolhida, a linguagem de definições de ontologia OWL (do inglês *Web Ontology Language*), mais especificamente da sua versão 2 (BOCK et al., 2012). Ela possui uma interface de programação na linguagem Java, chamada de OWL API (HORRIDGE; BECHHOFER, 2011). Possui uma ferramenta para edição de ontologias em OWL (TUDORACHE; VENDETTI; NOY, 2008). E possui muitos *reasoners*, para cada um dos vários subconjuntos da linguagem. Também atende a todos os requisitos feitos para a linguagem, já que ela é baseada na lógica descritiva (DENTLER et al., 2011). E é possível utilizar o mesmo modelo nos itens semânticos, no GD e na memória semântica, inclusive é possível utilizar a interface de programação para fazer as chamadas ao *reasoner* (ou *reasoners*) escolhido a partir do GD para fazer consultas, adicionar e remover axiomas e verificar a consistência da ontologia.

5.5 Saídas do gerenciador de diálogos

A principal saída do GD é a intervenção a ser executada pelo robô, definida em forma de atos dialogais com itens semânticos, seguindo o mesmo modelo das entradas. O conjunto de funções comunicativas que podem ser utilizadas na saída difere do

conjunto que pode ser utilizado na entrada. Por exemplo, *stalling* é utilizado apenas na entrada e não na saída.

Outra saída importante do GD é uma lista de expectativas para a próxima ação do usuário, na forma de atos dialogais acompanhados de itens semânticos e de uma probabilidade de acontecimento (conforme a crença do GD). Esta informação pode ser utilizada em outros módulos para resolver ambiguidades ou o complemento da ação do usuário em uma arquitetura incremental. Opcionalmente, o GD pode disponibilizar como saída um planejamento completo cobrindo muitos turnos, mostrando quais intervenções pretende executar mediante cada expectativa de ação do usuário.

A intervenção a executar, as expectativas relativas a esta intervenção e, eventualmente, planos de ações futuras do sistema, são saídas que são disponibilizadas no quadro negro. A atualização e disponibilização destas informações geram eventos que podem ser aproveitados por qualquer componente do sistema. Estas três saídas podem ser chamadas de saídas *diretas*.

A memória episódica e a memória semântica também podem ser atualizadas, porém sem gerar eventos. Essas são as saídas *indiretas* do GD. A cada turno, automaticamente, o GD insere informações na memória episódica que permitem restaurar tudo o que aconteceu no diálogo. Na memória semântica, ao contrário, as informações não são atualizadas automaticamente; devem ser atualizadas através de instruções específicas associadas às regras do dispositivo, de maneira que o desenvolvedor do sistema final tenha controle sobre o fluxo de vida delas.

6 CRITÉRIOS DE ESCOLHA DO MODELO

No início deste trabalho de pesquisa, verificou-se que a adaptatividade tem potencial para ser utilizada em várias tarefas específicas dentro do gerenciamento de diálogo, e com diferentes tecnologias. Para chegar ao modelo proposto adiante, foi necessário comparar os modelos iniciais entre si conforme um conjunto de critérios.

Para compor esse conjunto de critérios, obteve-se da literatura revisada algumas características desejáveis do sistema de diálogo falado que tem impacto direto no gerenciamento de diálogo. São elas:

- Múltiplas hipóteses paralelas: considerá-las parece ser uma estratégia mais robusta para identificar e manipular ambiguidades e erros inseridos pelos módulos anteriores.
- Processamento incremental ou não incremental das ações do usuário: um modelo incremental traz mais naturalidade, mas também mais complexidade de processamento no gerenciamento.
- Multimodalidade de entrada: um modelo multimodal de entrada traz mais naturalidade, mas traz também a possibilidade de ações paralelas do usuário entre voz e gestos.
- Prosódia e emoção na voz: são variáveis adicionais que podem ser utilizadas beneficamente na tomada de decisão, não necessariamente tornando mais complexo o modelo do gerenciamento.
- Iniciativa: apenas do sistema, apenas do usuário ou mista. Esta última dá mais flexibilidade ao usuário, sem ser obrigatoriamente mais complexa, dependendo apenas do método utilizado no gerenciamento.
- Sequenciamento do diálogo: alguns métodos de gerenciamento utilizam estimativa de estado ou uma agenda para que permitir a modelagem de um diálogo que siga naturalmente uma determinada sequência, de tal forma que a resposta a uma mesma ação do usuário possa ser diferente dependendo da sequência que levou a esta ação (contexto).
- Suporte ao reparo: é desejável que o método utilizado permita o reparo de qualquer informação no espaço de reparo, não apenas referente à última ação do usuário ou do sistema, sem ser necessário que o usuário solicite o retrocesso a estados anteriores onde ele possa fazer a correção.

- Facilidade de desenvolvimento: facilidade com que o desenvolvedor constrói e testa as regras de gerenciamento de diálogo em um novo domínio de aplicação.

Entretanto, após o início do detalhamento dos modelos, pode-se observar que esses critérios não eram suficientes, pois os modelos ainda se mostraram insuficientes para lidar com o processamento da própria parte linguística em alguns cenários específicos. Estes cenários e todas os outros que, idealmente, deveriam ser suportadas para a parte verbal, são detalhados neste capítulo, com exemplos de situações do cotidiano. Eles foram utilizados no refinamento do modelo e são adotadas como critérios complementares para avaliação dos resultados finais. Apresenta-se, inicialmente, uma notação mais formal para que possam ser entendidos com clareza.

6.1 Notação para descrição formal do modelo e dos cenários

O texto será descrito sempre do ponto de vista da problemática do GD, no momento em que se está recebendo a última ação do usuário, a_u, para decidir qual a próxima ação do sistema, a_s. As ações do sistema anteriores e posteriores serão descritas na forma a_s^t. O histórico de ações do sistema em um diálogo específico tem, portanto, a forma de uma série temporal

$$(a_s^{-\Delta},\ a_s^{-\Delta+1}, ..., a_s^{-1}, a_s, a_s^1, ...) \qquad (15)$$

em que Δ é a quantidade total de ações do sistema geradas desde o início do diálogo e que precedem a_s. De forma semelhante, as ações do usuário anteriores e posteriores a a_u têm a forma a_u^t, formando uma série temporal

$$(a_u^{-U},\ a_u^{-U+1}, ..., a_u^{-1}, a_u, a_u^1, ...) \qquad (16)$$

Em que U é a quantidade total de ações do usuário recebidas pelo gerenciador desde o início de diálogo e que precedem a_u. Tanto (15) quanto (16) são armazenados na memória episódica.

Cada uma das n observações (hipóteses) paralelas sobre a_u^t serão denotadas por o_i^t, em que i, $1 < i < n$, é utilizado para diferenciar entre si hipóteses da mesma ação. O grau de confiança em uma hipótese o_i^t será indicado por c_i^t. O conjunto de todas as observações possíveis que o sistema pode fazer sobre a ação do usuário será denotado por O. A parte verbal de cada o_i^t pode ser composto por m segmentos funcionais. Quando necessário, cada segmento será identificado por $\theta_{i,j}^t$, em que j, 1

< j < m, é utilizado para diferenciar entre si os segmentos funcionais de uma mesma hipótese o_i^t.

Cada segmento funcional pode, ainda, conter diferentes hipóteses sobre o conjunto de atos dialogais e itens semânticos. Para tornar a notação mais simples, não serão diferenciadas as várias camadas de hipóteses do modelo de entrada descrito em 3.3, mantendo apenas *i* para diferenciar todas as combinações de hipóteses de segmentação funcional, de atos dialogais e conteúdo semântico. Nos cenários descritos, tal diferenciação não mudaria ou tornaria mais clara a descrição. Desta forma simplificada, cada segmento funcional $\theta_{i,j}^t$ de a_u^t pode ter *p* atos dialogais, cada um definido na forma $\varphi_{i,j,k}^t$, 1 < *k* < *p*.

Cada ação do sistema a_s^t não tem um modelo de múltiplas hipóteses nem de vários segmentos funcionais, sendo composta diretamente por *p* atos dialogais, cada um na forma φ_{sk}^t, 1 < *k* < *p*. Esses atos dialogais geram *L* expectativas sobre a próxima ação do usuário, cada uma na forma $e_{k,l}^t$, 1 < *l* < *L*. Finalmente, o sistema pode associar a cada expectativa um grau de confiança $c_{k,l}^t$, que deve ser interpretado como a crença do sistema na concretização da mesma. Define-se Ψ_s como o conjunto de todos os atos dialogais que podem ser gerados pelo sistema.

6.2 Descrições dos cenários

Descreve-se neste tópico os cenários utilizados como critério para avaliar o modelo proposto utilizando-se a notação do tópico anterior. É necessário, inicialmente, descrever o cenário de *situação básica*; todos os demais cenários serão definidos como variações deste. Ela acontece quando:

- Uma única observação o_1 sobre a_u, com c_1 maior que um grau de confiança mínimo aceitável c_{min}, é recebida no GD
- Todos os atos dialogais podem ser vinculados com uma expectativa $e_{k,l}^{-1}$
- Apenas um ato dialogal $\varphi_{1,1,k}$ requer resposta do sistema
- Nenhum ato dialogal φ_{sk}^{-1} referente a a_s^{-1} que requeria resposta do usuário ficou sem resposta do mesmo
- A resposta a_s gerada pelo sistema contém apenas um ato dialogal φ_{sk} que requer resposta do usuário

Em alto nível, a situação básica se refere a um evento de entrada com apenas uma observação válida sobre a ação do usuário; o GD é capaz de manipular todos os atos dialogais deste segmento, mas apenas um deles requer resposta do sistema; todas as respostas requeridas do usuário foram dadas, não ficando nenhuma pendência; e a nova ação gerada pelo sistema contém apenas um ato dialogal que requer resposta do usuário. Um exemplo deste cenário pode ser visto na Figura 20.

Figura 20 – Exemplo de situação básica

a_s^{-1}: *Qual pizza você gostaria de pedir hoje?* a_u: *Tem alguma promoção hoje?* a_s: *O preço das pizzas de muzzarela e de calabresa estão promocionalmente em R$20,00. Gostaria de alguma delas?*

Observe que esse exemplo só pode ser considerado uma situação básica se o ato dialogal “Tem alguma promoção hoje?” estiver estre as expectativas de resposta para “Qual pizza você gostaria de pedir hoje?”. Observe também que dos dois segmentos funcionais de a_s apenas o último requer resposta do usuário, a pergunta “Gostaria de alguma delas?”.

Ressalta-se que a quase totalidade dos exemplos encontrados na literatura descrevem este cenário, o que tornou difícil aplicar os modelos propostos a situações reais, pois só então foi possível perceber que as propostas não eram totalmente adequadas.

6.2.1 Situação 1: Nenhuma hipótese tem grau de confiança maior que c_{min}

As hipóteses de entrada do gerenciador de diálogo precisam ter um grau de confiança maior que um grau de confiança mínimo c_{min} configurado pelo desenvolvedor da aplicação. Se nenhuma das hipóteses de entradas recebidas atende esta condição, deve-se disparar um procedimento de tratamento de erro, pois um de dois casos pode ter acontecido: (i) ou o usuário não falou (sendo a entrada possivelmente derivada de algum ruído do ambiente, como outra pessoa distante falando) ou (ii) o usuário disse algo cujo significado não pode ser determinado com confiança pelos módulos anteriores. O procedimento de tratamento deve incluir uma combinação de funções

comunicativas de *apology*, *negative-auto-feedback* e *contact-check*, como, por exemplo, em “Desculpe, você disse algo?” ou “Desculpe, você disse algo? Se disse, eu não entendi”.

6.2.2 Situação 2: Duas ou mais hipóteses têm grau de confiança maior que c_{min}

Neste caso, o gerenciador de hipóteses precisa decidir qual das hipóteses deve ser utilizada. Existem duas maneiras de o gerenciador fazer esta decisão:

- Descarta as hipóteses de entrada antes de realizar o processamento responsável decidir qual a ação a ser tomada;
- Realiza o processamento para decidir qual ação a ser tomada considerando todas as hipóteses. Somente então é feita a escolha da hipótese do usuário a ser considerada, com base na probabilidade das hipóteses de saída.

Idealmente, as hipóteses descartadas deveriam ser mantidas na memória de alguma forma, de tal maneira que se o usuário sinalizar, no turno a_u^1 ou posteriores, que a interpretação de a_u foi errônea, o gerenciador de diálogo possa utilizar as hipóteses descartadas para tentar reparar o erro de interpretação.

6.2.3 Situação 3: Duas ou mais hipóteses têm graus de confiança iguais ou separados por uma diferença menor que um c_{mindif} mínimo

Esta é uma variação do cenário da situação 2. Para selecionar a hipótese mais provável dentre duas ou mais hipóteses sobre a_u, a diferença entre o grau de confiança da hipótese mais provável e os graus de confiança das demais deve ser maior ou igual que uma diferença mínima exigida c_{mindif}. Se essa condição não for satisfeita, o gerenciador de diálogo deve disparar um procedimento de tratamento de erro, utilizando para isso uma combinação das funções comunicativas *apology*, *negative-auto-feedback* e *choice-question*, como no exemplo: “Desculpe, não tenho certeza se entendi. Você quis dizer ‘cabana ou ‘bacana’?”

6.2.4 Situação 4: a_u *com ao menos dois atos dialogais não relacionados, que requerem resposta do sistema*

Esta situação é uma variação do cenário da situação básica, em que dois ou mais atos dialogais requerem resposta do sistema, mas nenhum deles depende da interpretação do outro para ser compreendido pelo gerenciador, isto é, podem ser processados de forma independente.

Um exemplo dele é exibido no diálogo da Figura 21. Nela, o usuário reponde a uma pergunta com dois segmentos distintos, $\theta_{1,1}$ (“Quero uma de catupiry”) e $\theta_{1,2}$ (“Vocês aceitam cartão de crédito?”).

Figura 21 – Exemplo de situação em que a_u contém dois atos dialogais não relacionados que requerem resposta do sistema

a_s^{-1}: *Qual pizza você gostaria de pedir hoje?* a_u: *Quero uma de catupiry. Vocês aceitam cartão de crédito?* a_s: *OK, uma pizza de catupiry. Desculpe, não aceitamos cartão de crédito. Deseja saber outras formas de pagamento?*

Em $\theta_{1,1}$ temos dois atos dialogais: o primeiro de função comunicativa *answer* (relativa a a_s^{-1}) e o segundo de função comunicativa específica do domínio da tarefa em questão, *makeOrder* (relativa a uma pizza de catupiry). Do ponto de vista do sistema, é importante que o sistema confirme o entendimento da tarefa a ser realizada através de um *positive-auto-feedback* antes de realizá-la, no caso, registrar uma pizza de catupiry como parte do pedido; portanto este primeiro semento requer uma resposta do sistema.

Já em $\theta_{1,2}$ temos um único ato dialogal, de função comunicativa *check-question*, em que se deseja saber se é verdadeiro o axioma *sistema aceita ‘cartão de crédito’ como pagamento* (que em OWL poderia ser descrito de várias formas, como, por exemplo, *sistema temModoPagamento* ‘Cartão de crédito’, em que temModoPagamento é uma propriedade relacionando um sistema a um modo de pagamento). Esta pergunta foi direcionada ao sistema e requer uma resposta. Portanto, o diálogo da Figura 21 atende a todas as condições desta situação.

O gerenciador de diálogo deve ser capaz de responder a ambos, ou sinalizar que não pode responder a uma ou nenhuma das perguntas.

6.2.5 Situação 5: a_u *com ao menos dois atos dialogais que requerem resposta do sistema, sendo que um deles é relacionado às expectativas geradas após o processamento do outro*

Uma variação do cenário da situação 4 em que um dos atos dialogais do usuário que requerem resposta do sistema depende da interpretação de um outro ato dialogal do mesmo turno para ser interpretado corretamente. Um exemplo desta situação é exibido na Figura 22.

Figura 22 – Exemplo de situação em que a_u contém dois atos dialogais que requerem resposta do sistema, e a interpretação do segundo depende da interpretação do primeiro

a_s^{-1}: *Qual pizza você gostaria de pedir hoje?*
a_u: *Quero uma de calabresa. Sem cebola, por favor.*
a_s: *OK, uma pizza de calabresa sem cebola. Deseja mais alguma coisa?*

Para entender corretamente a_u, o sistema precisa processar primeiro processar o primeiro segmento “Quero uma de calabresa”, pois a restrição “sem cebola”, mencionada no segundo segmento, se aplica à pizza mencionada no primeiro. O modelo de gerenciamento de diálogo deve permitir ao desenvolvedor lidar também com esta situação.

6.2.6 Situação 6: a_s *com dois ou mais atos dialogais exigindo resposta do usuário*

Este cenário é caracterizado por ter dois ou mais atos dialogais em a_s que requerem resposta do usuário. Esta situação deveria, idealmente, ser evitada pelo GD. O principal motivo é que a resposta do usuário pode responder a ambas e ainda adicionar mais um ato dialogal que requeira resposta do sistema em a_s^1, podendo levar a um aumento sucessivo de complexidade. O melhor seria se o GD, ao detectar tal situação, executasse apenas um dos atos dialogais neste turno e postergasse a execução do outro ato dialogal para turnos futuros, podendo inclusive sinalizar ao usuário que este outro assunto será retomado adiante.

6.2.7 Situação 7: a_u *contém um ato dialogal que o GD não manipula*

Variação do cenário da situação básica, em que ao menos um ato dialogal $\varphi_{1,j,k}$ de entrada não pode ser vinculado com qualquer expectativa $e_{k,l}^{-1}$, e o GD não pode encontrar qualquer regra que manipule $\varphi_{1,j,k}$, mesmo entre aquelas fora da lista de expectativas.

Neste caso, o GD deve disparar um procedimento de tratamento de erro cuja ação do sistema envolva as funções comunicativas *apology*, *negative-auto-feedback* e *request-to-repeat* (solicitação para repetir) ou *request-to-rephrase* (solicitação para reformular), como no exemplo “Desculpe, posso não ter entendido o que você disse. Você pode reformular ou repetir mais lentamente?”

6.2.8 Situação 8: a_u *contém um ato dialogal que o GD pode manipular, não está entre as expectativas atuais mas certamente é relevante para ações futuras do sistema*

Variação do cenário da situação básica, em que ao menos um ato $\varphi_{1,j,k}$ não pode ser vinculado qualquer expectativa $e_{k,l}^{-1}$, mas o GD encontrou ao menos uma regra que que sabe manipular $\varphi_{1,j,k}$, e esta regra é um requisito para a finalização com sucesso do diálogo. Neste caso, o GD deveria permitir a execução da regra encontrada, se todas as condições para a execução da mesma tiverem sido satisfeitas, e adaptar os planos existentes para que a informação já obtida, por antecipação do usuário, não seja solicitada.

Ressalta-se também que a situação 8 e a situação 5 podem ocorrer simultaneamente.

6.2.9 Situação 9: a_u *contém um ato dialogal que o GD pode manipular, não está entre as expectativas atuais e não se sabe ou não é relevante para ações futuras do sistema*

Variação do cenário da situação básica, em que ao menos um ato $\varphi_{1,j,k}$ não pode ser vinculado com qualquer expectativa $e_{k,l}^{-1}$ e o GD encontrou ao menos uma regra que

que sabe manipular $\varphi_{1,j,k}$, mas ela certamente não será executada ou não se sabe se ela será executada.

No primeiro caso, em que a regra certamente não será executada, deve-se disparar um procedimento de tratamento de erro (de forma similar a situação 7) ou um procedimento de reparo; este último somente se for detectado que a aparente fala desconexa do usuário poderia ter sido causada por um erro de interpretação. No caso em que a execução da regra é incerta, pode-se proceder da mesma forma do primeiro caso ou ainda de outras duas formas distintas: (i) altera-se o plano de execução para adicionar a informação dada pelo usuário somente no caminho de regras que levaria futuramente a tal regra ou (ii) assume-se que o usuário já escolheu, implicitamente, o caminho específico que levaria a esta regra e o sistema faz, portanto, o mesmo, o que pode ser um tanto sutil. Um exemplo de utilização desta última forma pode ser visto na Figura 23.

Figura 23 – Exemplo de situação em que um ato dialogal do usuário não estava entre as expectativas do usuário, mas entre as expectativas de um dos possíveis caminhos do diálogo

a_s^{-1}: *Você gostaria da pizza de calabresa ou da pizza de catupiry?* a_u: *Sem cebola, por favor. Quanto fica?* a_s: *OK, uma pizza de calabresa sem cebola. O preço é R$22,00 com a entrega. Deseja mais alguma coisa?*

Neste exemplo, o sistema assume que o usuário selecionou implicitamente a pizza de calabresa em a_u, pois não existe regra relacionada à expectativa de resposta “Sem cebola” no caminho que se segue à seleção de pizza de catupiry. O sistema, entretanto, deixa explícito seu entendimento através do *feedback* “OK, uma pizza de calabresa sem cebola”.

O método de GD deveria permitir ao desenvolvedor de uma aplicação lidar com essa situação, através de ao menos uma das formas mencionadas.

6.2.10 Situação 10: a_s^{-1} *requeria ações do usuário que não foram atendidas*

Esta situação acontece quando ao menos um ato dialogal φ_{sk}^{-1} referente a a_s^{-1} que requeria resposta do usuário ficou sem resposta em a_u. O usuário pode não ter

entendido plenamente a ação anterior do sistema ou ter entendido mas a considerar irrelevante para seus objetivos quanto ao diálogo com o sistema. O GD deveria disparar um procedimento específico para tratar esta situação, envolvendo (i) atos dialogais com as funções comunica de *apology*, (ii) repetição dos atos dialogais não atendidos e, se possível, (iii) informar o usuário que ele pode desistir desta parte da conversa, como no exemplo “Desculpe, eu perguntei se você queria pizza de calabresa. Deseja desistir de pedir tal pizza?”

6.2.11 Situação 11: Capacidade de postergar quando a_u *gera expectativas sobre* a_s *que não podem ser respondidas no tempo de* a_s

Esta situação pode ser entendida como um caso especial dos cenários das situações 8 e 9. Ela acontece tipicamente quando o sistema requer o fornecimento de alguma informação específica antes que o objetivo principal do usuário possa ser realizado, como no caso da Figura 24. Neste exemplo, a Pizzaria do Fulano requer a identificação completa do usuário antes de registrar um pedido do mesmo, pedindo possivelmente informações como endereço e nome completo se for o seu primeiro acesso. Como o usuário teve a iniciativa de pedir a pizza antes que o sistema pudesse identificá-lo, o GD precisou postergar a realização do pedido.

Figura 24 – Exemplo de situação em que o sistema posterga a resposta a uma requisição do usuário

a_s^{-1}: *Pizzaria do Fulano, boa noite. Com quem falo?*
a_u: *Com o Ciclano. Queria uma pizza de calabresa, por favor.*
a_s: *Desculpe, antes de continuar com o pedido, preciso de sua identificação completa. Este é a primeira vez que pede pizza por telefone conosco?*

6.2.12 Situação 12: Capacidade de retomar um assunto postergado quando se percebe que o mesmo pode ser respondido

Pode ser entendido como um complemento da situação anterior. Acontece quando o sistema retoma um assunto postergado anteriormente, ao perceber que todas as pré-condições estão agora satisfeitas. Um exemplo está na Figura 25.

O modelo do gerenciador de diálogos deveria ser capaz de não só permitir ao desenvolvedor criar aplicações que posterguem assuntos, mas também que os retomem quando uma pré-condição for atingida.

Figura 25 – Exemplo de situação em que o sistema retoma um assunto, postergado no exemplo da Figura 24.

a_s^{-1}: *Confirma que seu nome é Ciclano Fulanóide, morador da Rua dos Ciclanos, número 23 e seu telefone de contato é 1111-1111?* a_u: *Confirmo.* a_s: *Cadastro realizado com sucesso. Confirma o pedido de uma pizza de calabresa?*

6.2.13 Situação 13: Usuário repara a_s^{-1} *em* a_u

Esta situação equivale ao reparo de uma ação do sistema na segunda posição, isto é, quando o usuário repara a ação do sistema já no turno imediatamente seguinte. É a situação mais comum de reparo. A Figura 26 contém um exemplo.

Figura 26 – Exemplo de situação de reparo de uma ação do sistema na segunda posição.

a_s^{-1}: *Confirmando uma pizza de calabresa. Mais alguma coisa?* a_u: *Não não, eu disse que desisti da pizza de calabresa.* a_s: *Desculpe-me. Cancelando pizza de calabresa. De que outra pizza o senhor gostaria?*

É importante notar que o usuário pode reparar um ato dialogal do sistema que requeria uma ação do usuário como resposta. O projeto de gerenciamento precisa ser feito cuidadosamente para detectar tal caso e trata-lo adequadamente para que, ao mesmo tempo em que se faz o reparo, não se cobre do usuário a resposta do que foi reparado.

6.2.14 Situação 14: O sistema repara a_s^{-1} *em* a_s

É o reparo da ação do sistema na terceira posição, isto é, pelo próprio sistema. Requer alguma inferência por parte do gerenciador do diálogo para perceber sozinho que é necessário esclarecer ou corrigir um turno anterior. Uma forma de fazer isso é detectar que o usuário não entendeu a ação anterior, pois a_u contém um ou mais atos dialogais que só fariam sentido se a_s^{-1} tivesse sido diferente.

Figura 27 – Exemplo de situação de reparo de uma ação do sistema na terceira posição.

a_s^{-2}: *Você gostaria de uma pizza portuguesa ou de uma francesa?*
a_u^{-1}: *Da última que você disse.*
a_s^{-1}: *OK, francesa. Deseja mais alguma coisa?*
a_u: *Por favor, sem cebola na pizza.*
a_s: *A nossa pizza francesa não contém cebola. Você tinha escolhido portuguesa ou francesa?*

Um exemplo está na Figura 27. A resposta do usuário em a_u só faria sentido se o sistema tivesse entendido a_u^{-1} de forma errônea ou se o próprio usuário estivesse confuso sobre a cobertura da pizza francesa, fazendo-se necessário um esclarecimento.

6.2.15 Situação 15: Usuário repara a_s^{-2+} *em* a_u

Esta situação se refere ao reparo da ação do sistema no quarto turno, isto é, quando o usuário repara em a_u o turno a_s^{-2} ou anterior. Para ser capaz de suportar esta situação, o sistema precisa manter o espaço de reparação completo, o que permitiria ao usuário reparar qualquer ação anterior do sistema.

Se houver dependência semântica de outras ações já realizas pelo sistema com a semântica do turno reparado, pode ser necessário repetir essas ações para corrigir as dependências. Um exemplo acontece ao se reparar uma ação a_u^{-3} em a_u, quando as informações de a_u^{-2} e a_s^{-2} dependem semanticamente de a_u^{-3} O mais sensato seria, então, o GD descartar a informação obtida em a_u^{-2} e refazer o planejamento de suas ações.

Deve-se observar que talvez não seja possível que o GD detecte automaticamente essa dependência semântica entre as ações, sendo necessário ao desenvolvedor da aplicação informar como proceder em tais casos. Não fez parte deste trabalho de pesquisa a discussão dessa detecção automática.

6.2.16 Situação 16: O sistema repara a_s^{-2+} *em* a_s

É o reparo da ação do sistema no quinto turno ou posterior, isto é, realizado pelo próprio sistema. Esta situação tem a mesma dificuldade de detecção da situação 14 (reparo no terceiro turno) aplicada a todos os turnos anteriores, e não somente ao último. Inclusive pode-se entender a situação 14 como um caso especial do cenário da situação 16.

6.2.17 Situação 17: O sistema repara a_u *em* a_s

É o reparo pelo sistema da ação do usuário no segundo turno. Nela, o GD precisa inferir que o usuário precisa de esclarecimento ou de correção. Isso pode ser feito ao se verificar que o ato dialogal do usuário ou a parte semântica do ato dialogal não correspondem às expectativas anteriores. Novamente, como na situação 13, é necessário cuidado extra para não se cobrar do usuário respostas exigidas pela própria ação sendo reparada.

6.2.18 Situação 18: O usuário repara a_u^{-1+} *em* a_u

É o reparo pelo usuário da sua própria ação no terceiro turno após a ação reparada, ou posterior. Pode ser causada porque o usuário falou algo errado ou incompleto ou mesmo porque o usuário mudou sua intenção. Na prática, é similar à situação 13, pois comumente o GD não poderá determinar com 100% de confiança se a origem do erro foi do usuário em a_u^{-1} ou se foi erro de interpretação enquanto decidindo a ação a_s^{-1}.
A Figura 28 contém um exemplo desta situação. Repare que o exemplo utiliza as hipóteses consideradas de ação do usuário, e não as ações propriamente ditas, para ressaltar o ponto de vista do GD. Nele, não se pode afirmar se o usuário realmente falou o_1^{-1} ou se o sistema apenas entendeu errado, sendo difícil classificar este

exemplo como situação 13 ou situação 18. Entretanto, para fins práticos, o comportamento do sistema deve ser o mesmo.

Figura 28 – Exemplo de situação de reparo em que o gerenciador de diálogo não pode determinar a origem do erro reparado pelo usuário

o_1^{-1}: *Quero então uma pizza de quatro queijos um.* a_s^{-1}: *OK, quatro queijos um. Mais alguma coisa?* o_1: *Eu disse quatro queijos dois.* a_s: *Desculpe, confirmando um pedido de pizza quatro queijos dois. Mais alguma coisa?*

6.2.19 Situação 19: O sistema repara a_u^{-1+} *em* a_s

Esta é, talvez, a situação de reparo mais complexa. É o reparo de uma ação do usuário pelo sistema no quarto turno ou posterior. Requer inferência de para ser detectada, e se feito de forma indevida, como todo reparo de ação do usuário, pode levar à insatisfação do mesmo. Um exemplo está na Figura 29. O GD percebe em a_s que existe uma contradição entre o_1^{-1} e o_1, e utiliza atos dialogais com funções comunicativas *apology* ("Desculpe"), um *inform* sobre o entendimento de o_1^{-1} ("*entendi que você queria uma pizza sem carne*"), um *inform* sobre o entendimento de com uma relação retórica de contraste com o ato dialogal anterior ("*mas que está agora pedindo com cobertura extra de bacon*") e finalmente um *check-question* para saber se o entendimento está correto ("*Entendi corretamente?*").

Figura 29 – Exemplo de reparo de ação do usuário pelo sistema vários turnos depois, decorrente de mudança de intenção do usuário

o_1^{-1}: *Gostaria de uma pizza com champignon mas sem carne.* a_s^{-1}: *OK, temos várias pizzas com champignon mas sem carne. Por exemplo temos a pizza vegetariana dois, com brócolis e (...)* o_1: *Vou querer uma pizza vegetariana dois então. Com cobertura extra de bacon.* a_s: *Desculpe, entendi que você queria uma pizza sem carne mas que está agora pedindo com cobertura extra de bacon. Entendi corretamente?*

6.2.20 Situação 20: Manutenção de iniciativa

O GD deve ser capaz de manter a iniciativa sempre que necessário. Existem duas situações em que isto acontece:

- Quando a_u está de acordo com todas as expectativas do sistema mas não gera expectativas sobre a próxima ação do sistema;
- a_s não requer ação do usuário como resposta e o usuário não toma a iniciativa.

O GD deve manter um plano ou uma lista de objetivos que auxilie a decidir o caminho a seguir nestas situações. Se todos os objetivos já tiverem sido atingidos pode ser necessário ter a iniciativa para encerrar a conversa, como exemplificado na Figura 30. A ação do usuário obtida da hipótese o_1 não gera nenhuma expectativa sobre a próxima ação do sistema, mantendo a iniciativa com o sistema, que opta por encerrar a conversa.

Figura 30 – Exemplo de da iniciativa do sistema para encerrar o diálogo

a_s^{-1}: *Tudo certo para o seu pedido então. Ele será entregue em até trinta minutos. Deseja mais alguma coisa?*
o_1: *Não, obrigado.*
a_s: *A pizzaria do Fulano agradece a preferência e tenha uma boa noite.*

6.2.21 Situação 21: A hipótese mais provável não pode ser vinculada com alguma expectativa do sistema, mas outra hipótese menos provável encaixa

Quando o GD recebe duas hipóteses com grau de confiança maior que c_{min}, e a hipótese mais provável não pode ser vinculada com nenhuma das expectativas, mas a outra encaixa, pode-se optar pela menos provável ou perguntar ao usuário qual das hipóteses é a correta, mesmo que a diferença entre o grau de confiança de ambas seja maior que c_{mindif} (o que é diferente da situação 3). Deve-se tomar cuidado para não se supervalorizar aquela que tem vínculo com as expectativas, pois pode ser que os componentes anteriores do sistema, como a análise semântica ou o reconhecedor de voz, já tenham aumentado o grau de confiança em tal hipótese apenas por ela estar entre as expectativas, já que eles também têm acesso ao quadro negro onde as saídas do GD ficam armazenadas.

7 ADAPTALKER: UM MODELO PARA O GERENCIAMENTO ADAPTATIVO DE DIÁLOGO

O modelo proposto e detalhado a seguir, chamado de *Adaptalker*, é baseado em máquinas de estado. Naturalmente, herdam-se as vantagens e desvantagens deste modelo de base. Ao escolhê-lo, considerou-se mais importante mostrar como a adaptatividade é útil no gerenciamento de diálogo do que propor um modelo que atendesse bem em todos os critérios listados previamente. Ainda assim, é mostrado que o Adaptalker satisfaz muito bem a maior parte dos critérios.

Dos critérios básicos obtidos da literatura, optou-se por utilizar um modelo não incremental, por ser menos complexo. Um modelo incremental seria mais trabalhoso e seria irrelevante para os pontos em que a adaptatividade é utilizada nesta dissertação. Também se optou por não utilizar a multimodalidade pelos mesmos motivos.

O Adaptalker é formado por duas camadas distintas. A primeira, chamada de *filtro*, não contém adaptatividade e é responsável por filtrar os eventos que deverão realmente ser processados e por manipular as hipóteses que serão trabalhadas pela camada seguinte. A segunda camada é formada por vários *Dispositivos Adaptativos de Tomada de Decisão* (doravante chamados apenas por DATD), transdutores adaptativos que, descritos de forma resumida, mapeiam atos dialogais do usuário e informações na memória semântica em atos dialogais do sistema e atualizações da memória semântica. O filtro dispara o processamento de vários DATD simultaneamente, um para cada hipótese com grau de confiança maior que um c_{min} configurado. Cada DATD irá realizar o processo de tomada de decisão e retornar um conjunto de *p* atos dialogais na forma φ_{sk}^{t}, com um grau de confiança C_i^t, e um conjunto de expectativas $e_{k,l}^{t}$. A camada de filtro decide, então, qual das respostas será utilizada e a armazena no quadro negro.

O Adaptalker foi organizado de forma que um desenvolvedor precise apenas personalizar o conjunto de regras de tomada de decisão do DATD para criar um novo GD para sua aplicação. Para viabilizar isto, mesmo um erro detectado na camada de filtro é enviado para o dispositivo adaptativo através de atos dialogais com função comunicativa reservada a uso interno, isto é, cujos atos dialogais relacionados não podem ser gerados fora do GD. O desenvolvedor pode reutilizar em diferentes aplicações as funções comunicativas e as regras de tomada de decisão referentes a

situações comuns na maioria dos diálogos, tais como aquelas referentes ao cumprimento, encerramento, reparo e gerenciamento da comunicação. Pode-se, entretanto, definir funções comunicativas exclusivas de um sistema, como, por exemplo, para permitir a comunicação ao GD de eventos de identificação biométrica do usuário.

Uma característica importante da camada de tomada de decisão é a organização das regras do DATD em submáquinas. Esta organização não aumenta o poder computacional do dispositivo nem viabiliza a resolução de algum problema específico do gerenciamento de diálogo, mas torna o dispositivo mais expressivo, pois facilita o entendimento e a manutenção das regras, e diminui o espaço de procura de regras, o que é especialmente útil na execução de ações adaptativas.

7.1 Filtro

A primeira tarefa do filtro é decidir quais eventos devem ser utilizados ou descartados. Como o modelo proposto é não incremental, apenas os eventos de nova ação e de ação finalizada são utilizados, e os demais são descartados. Os eventos de ação finalizada são utilizados para disparar o processo de tomada de decisão na camada seguinte.

Os eventos de nova ação são utilizados para evitar turnos simultâneos do sistema e do usuário. Entretanto, ser não incremental traz uma desvantagem aqui. Para explicar o porquê, deve-se detalhar os dois casos que levam o usuário a falar ao mesmo tempo que o sistema:

1. O usuário já utilizou o sistema muitas vezes e sabe qual é o próximo a_s. Para não perder tempo, ele responde com a_u^1 antes de o sistema terminar de pronunciar a_s. Neste caso, o GD deveria considerar a_s como totalmente comunicada ao usuário.
2. O usuário na verdade não havia terminado o seu turno ou o retomou para reparar algo, sem ter conhecimento de como seria a_s. Neste caso, o GD deveria considerar a_s como não comunicada ao usuário e reprocessar as informações passadas em a_u e a_u^1 conjuntamente para gerar uma nova ação a_s^1 que substitui totalmente a_s. Repara-se que se houvesse processamento incremental, talvez fosse possível determinar a_s como parcialmente comunicado.

Não é tarefa simples decidir qual dos dois casos deve ser considerado. Na dúvida, deve-se sempre assumir o segundo caso; pois, se no primeiro o usuário está adiantando informações e o GD suporta as situações 5, 8 e 9 listadas no Capítulo 6 (o que é o caso do Adaptalker), não haverá problemas em processar ambos os turnos do usuário como se fossem um único.

A segunda tarefa é a de manipulação das múltiplas hipóteses. Se acontece a situação 1 (nenhuma hipótese tem grau de confiança maior que um c_{min} configurado) então será passado para o DATD um ato dialogal substituto de função comunicativa reservada *noUnderstanding* indicando erro total de compreensão. Se acontece a situação 2 (duas ou mais hipóteses tem grau de confiança maior que c_{min}) então criam-se múltiplas cópias do dispositivo adaptativo atual, uma para cada hipótese paralela de entrada. Quando todas as cópias do DATD terminam o processamento, o filtro decide qual hipótese será utilizada com base na probabilidade a posteriori da hipótese, que é obtida a partir da multiplicação do grau de confiança na hipótese de entrada c_i^t com o grau de confiança do GD na resposta C_i^t. Se a diferença entre a probabilidades a posteriori é menor que c_{mindif}, então o GD irá restaurar o DATD original do início do turno e enviará a ele um ato dialogal substituto de função comunicativa reservada *doubtfulUnderstanding*, cujo item semântico é composto pelas hipóteses competidoras.

Existe a possibilidade de uma hipótese ser enviada para o DATD contendo uma hipótese de ato dialogal ou de item semântico menor que c_{min}. Neste caso, apenas os atos dialogais relacionados à hipótese problemática serão substituídos por atos dialogais de função *partialUnderstanding* indicando erro parcial de compreensão e com a hipótese original passada como item semântico.

7.2 Dispositivo adaptativo de tomada de decisão: definição formal

Define-se o DATD como uma quíntupla

$$DATD = (M, O, \Psi_s, M_a, M_0) \tag{17}$$

em que:

- M é o conjunto de todas as submáquinas que podem ser utilizadas no DATD;
- O é o alfabeto de entrada, isto é, o conjunto de todos os atos dialogais que podem ser observados a partir de uma ação do usuário;

- Ψ_s é o alfabeto de saída, isto é, o conjunto de todos os atos dialogais que podem ser utilizados como saída pelo sistema;
- M_a é uma lista ordenada de submáquinas ativas. Cada vez que uma nova máquina é ativada, ela entra na posição inicial da lista. Quando a máquina é inativada, ela é removida da lista. Se M_a ficar vazia, não será possível a continuidade do diálogo utilizando o dispositivo;
- M_0 é a configuração inicial de M_a ao início de um diálogo.

Cada submáquina $\mu \in M$ é definida por uma sétupla

$$\mu = (Q, B, T, F, X, q_0, Q_F) \quad (18)$$

em que:

- Q é o conjunto de estados de μ;
- B é o conjunto de estados de *binding*. São estados especiais utilizados para o vínculo de $\varphi_{i,j,k}^{t}$ com $e_{k,l}^{t-1}$;
- T é o conjunto de transições de estado de μ;
- F é o conjunto de funções adaptativas que podem ser utilizadas em μ;
- X é o conjunto de variáveis que podem ser utilizadas em μ para armazenar resultados de consultas entre as transições;
- q_0 é o estado inicial de μ, $q_0 \notin B$;
- Q_f é o conjunto de estados de aceitação de μ.

Cada transição pertencente a T é definida por uma óctupla

$$\gamma_{i,\pi} = (q_i, q_j, \pi, c_\pi, \alpha_a, \alpha_p, \beta, foco) \quad (19)$$

em que:

- q_i é o estado inicial da transição;
- q_j é o estado final da transição. Se $q_i \in B$, então $q_j \notin B$;
- π é a condição para que a transição seja disparada, utilizando comparadores lógicos, identificadores de variáveis $x \in X$ e literais. Se $q_i \in B$, então π também é definido através de funções comunicativas e de filtros semânticos (descritos através de expressões na linguagem escolhida para a memória semântica) para o item semântico de $\varphi_{i,j,k}^{t}$. A transição pode não ter qualquer condição, isto é, a condição é a cadeia vazia $\{\varepsilon\}$;
- c_π é a expectativa que o gerenciador de diálogos tem de que a condição π será satisfeita. Sua definição é opcional e permitida apenas se $q_i \in B$. É utilizada para definição do grau de confiança $c_{k,l}^{t}$ na expectativa $e_{k,l}^{t}$;

- α_a é uma ação adaptativa executada antes da transição de estado;
- α_p é uma ação adaptativa executada após a transição de estado;
- β é a tarefa que será executada ao final da transição;
- *foco*: diz quem receberá o foco de execução das regras após a execução da tarefa β, dentre o usuário, a submáquina atual e outra submáquina.

Os seguintes valores são possíveis para o foco:

- Foco no usuário: interrompe a execução da submáquina antes da troca de estado para obtenção da resposta do usuário;
- Foco na submáquina: requer a manutenção do foco na submáquina corrente para a execução da próxima regra do dispositivo;
- Liberação de foco: libera o foco para outra submáquina. Se o foco é liberado em um estado final, a submáquina corrente será inativada, isto é, removida de M_a;
- Foco neutro: deixa o DATD decidir se a submáquina irá ou não manter o foco. Para ter foco neutro, a submáquina deve ser capaz de manter a iniciativa se necessário, isto é, deve ter uma transição começando em q_j que pode ser acionada incondicionalmente.

As tarefas utilizadas em β podem ser de quatro tipos:

- Tarefa de requisição ao usuário: gera um ou mais atos dialogais φ_{sk}^{t} que requerem resposta do usuário, com função comunicativa pertencente a Ψ_s. Pode ser utilizada se e somente se $q_j \in B$. Permite apenas foco no usuário;
- Tarefa de informação ao usuário: gera um ou mais atos dialogais φ_{sk}^{t} que não requerem resposta do usuário, com função comunicativa pertencente a Ψ_s. Pode ser utilizada para qualquer q_j. Permite os seguintes valores de foco: na submáquina, neutro ou liberação;
- Tarefa de sistema: executa um comando de sistema, como uma consulta ou gravação na memória ou integração com um sistema externo. Permite apenas foco na submáquina, neutro ou liberação;
- Chamada de submáquina: procura uma submáquina e a coloca no topo de M_a, passando a execução para ela após transitar para q_j. Aceita apenas foco de liberação ou neutro.

Uma ação adaptativa é utilizada para chamar uma função adaptativa com parâmetros específicos. Uma função adaptativa $f \in F$ é definida como uma quádrupla

$$f = (\Lambda, V, G, A_f) \quad (20)$$

em que:

- Λ é uma sequência de parâmetros formais (λ_1, λ_2, ..., λ_m);
- V é um conjunto de identificadores de variáveis {v_1, v_2, ..., v_y} internas de f cujos valores são desconhecidos no instante de chamada de f mas que uma vez preenchidos terão seus valores preservados durante toda a execução da função;
- G é um conjunto de identificadores de geradores {g^*_1, g^*_2, ..., g^*_z}, variáveis especiais que são preenchidas com novos valores, ainda não utilizados pelo autômato, a cada vez que a função é chamada;
- A_f é uma sequência de ações adaptativas elementares executadas em f. Ressalta-se que a imposição de uma ordem nas ações não faz parte da definição original do formalismo adaptativo, tal como relatado para Maquinas de Markov Adaptativas em 4.1.4.

Desta forma, uma ação adaptativa α pode ser definida formalmente como

$$\alpha = f\ (\omega_1, \omega_2, ..., \omega_m) \cup \{\varepsilon\} \quad (21)$$

sendo f uma função adaptativa pertencente a F e (ω_1, ω_2, ..., ω_m) uma lista de m argumentos que correspondem posicionalmente à lista de parâmetros Λ declaradas para f. {ε} representa a cadeia vazia, isto é, é possível que uma transição não tenha ação adaptativa associada.

7.3 Ações adaptativas elementares do DATD

Diversas ações adaptativas elementares podem ser utilizadas em A_f. Elas são utilizadas nas funções adaptativas para viabilizar alterações no conjunto de regras do DATD.

7.3.1 Ação elementar de consulta

A ação elementar de consulta do DATD é equivalente a ação elementar de consulta do formalismo original de autômato adaptativo (JOSÉ NETO, 1993), embora com algumas pequenas diferenças. Ela busca transições dentro do dispositivo conforme

um filtro recebido através de parâmetros e as armazena em uma variável $v \in V$. Pode ser definida formalmente como

$$\beta_c = (v, \mu_c, q_i, q_j, \pi, \beta) \quad (22)$$

em que

- $v \in V$;
- $\mu_c \in M \cup \{\varepsilon\}$ é a submáquina onde a busca será realizada. Se igual a $\{\varepsilon\}$, a consulta será feita em todas as submáquinas do dispositivo;
- q_i é o filtro de estado inicial da transição. Se não definido, são buscadas transições iniciando em qualquer estado;
- q_j é o filtro de estado final, se não definido, são buscadas as transições terminando em qualquer estado;
- π é o filtro de condição. Pode ser vazio ou um conjunto formado por variáveis $x \in X$ e funções comunicativas pertencentes a O. Busca-se então transições que utilizem essas variáveis e funções comunicativas em suas condições;
- β é filtro de tarefa. Pode ser vazio ou um conjunto de funções comunicativas pertencentes a Ψ_s. Busca-se então transições que utilizem essas variáveis e funções comunicativas em suas condições.

7.3.2 Ação elementar de adição

A ação elementar de adição do DATD é equivalente à ação elementar de adição do formalismo original de autômato adaptativo. Ela requer a definição de (μ_i, q_i, q_j, π, c_μ, α_a, α_p, β, *foco*); π, β, c_μ, α_a e α_p são opcionais; μ_i é a máquina onde a inserção será feita. Podem-se utilizar os valores dos estados, condições, tarefas e ações adaptativas obtidas por uma ação de consulta prévia como parâmetros da adição.

Em uma mesma submáquina, pode existir mais de uma transição iniciando e terminando nos mesmos estados q_i e q_j, mas não com as mesmas condições. Ao se tentar inserir uma transição com o mesmo estado inicial e a mesma condição de uma transição existente acontecerá um erro e a inserção será cancelada; entretanto, o processamento do dispositivo não será encerrado, continuando normalmente.

7.3.3 Ação elementar de remoção

A ação elementar de remoção do DATD é equivalente a ação elementar de remoção do formalismo original de autômato adaptativo. Ela recebe como parâmetro uma variável $v \in V$ contendo zero ou mais transições que serão removidas do dispositivo. Se uma ou mais transições armazenadas em v não fizerem parte do dispositivo, estas não são removidas.

7.3.4 Outras ações elementares

Existem outras ações elementares que são utilizadas no DATD:

- Alteração: altera q_i, q_j, π, c_μ, α_a, α_p, β ou o *foco* de qualquer transição.
- Comparação entre duas variáveis: esta ação recebe quatro parâmetros (ω_1, ω_2, $a_=$, $a_\neq$). Se ω_1 e ω_2 forem iguais, executa-se $a_=$; caso contrário, executa-se $a_\neq$.
- Chamada de ação adaptativa não-elementar: recebe como parâmetro uma função adaptativa f e uma lista de argumentos (ω_1, ω_2, ..., ω_n) que correspondem posicionalmente à lista de parâmetros Λ declaradas para f.

Todas elas são derivadas das ações elementares de consulta, adição e remoção, conforme (ALFENAS et al., 2012).

7.4 Algoritmo de execução do DATD

O algoritmo geral de execução do DATD pode ser observado no diagrama de atividades da Figura 31. Antes de cada diálogo, o Filtro coloca o DATD em sua configuração inicial (“Inicialização”). Em seguida ou após cada ação do usuário, o DATD é acionado para gerar um candidato à próxima ação do sistema. Ele primeiramente seleciona uma submáquina para processar toda ou parte da ação do usuário. Regras da submáquina selecionada serão processadas até que a própria submáquina libere o processamento. O “Loop de seleção de submáquina” decide, então, se as saídas geradas pelo DATD já são candidatas suficientemente boas à próxima ação do sistema: o próximo passo poderá ser a chamada de uma submáquina já selecionada, a atividade de seleção de submáquina ou retornar o processamento para o Filtro. As seções abaixo detalham o funcionamento de cada uma das atividades exibidas na figura.

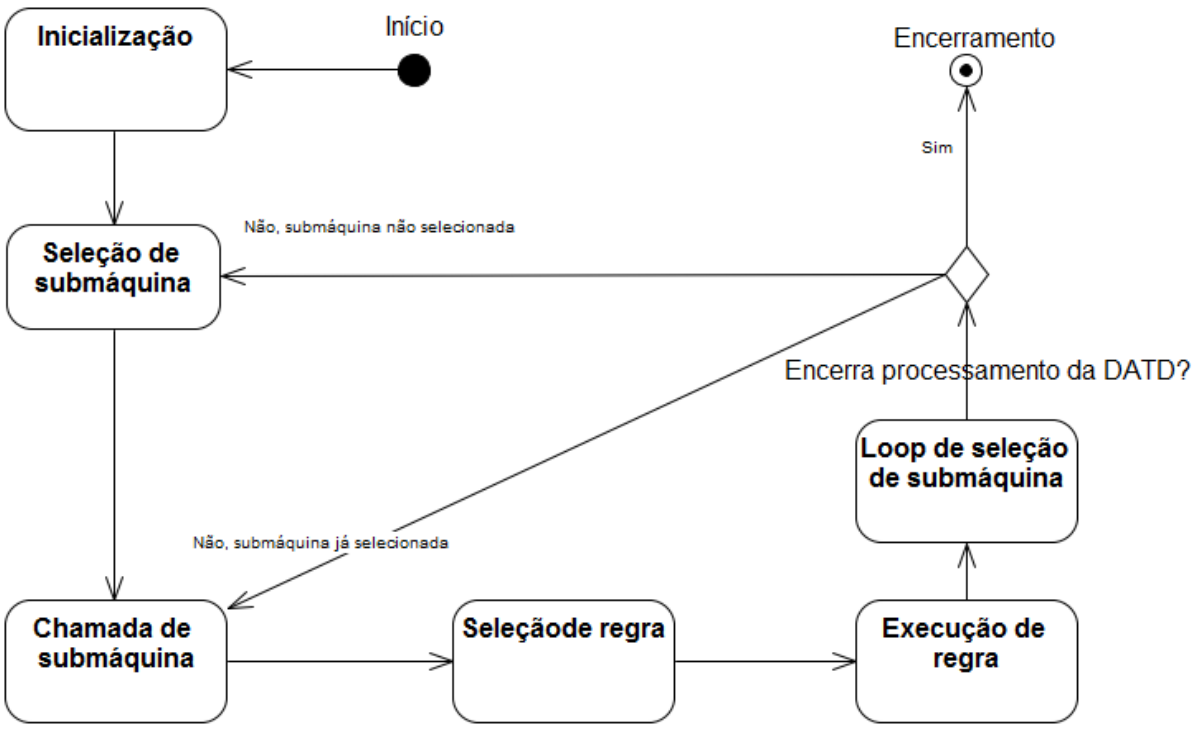
7.4.1 Inicialização

Na inicialização do DATD, são carregadas todas as regras do dispositivo e a memória semântica, incluindo um modelo do usuário inicialmente vazio. Esta atividade é realizada apenas na inicialização do Adaptalker, isto é, no início da conversa. As demais execuções do DATD utilizarão a mesma configuração em que o dispositivo estava ao final de sua execução anterior.

As estruturas de dado de execução abaixo são criadas para o controle do dispositivo:

- M_a: contendo como valor inicial M_0;
- Lista de entrada ζ: inicialmente vazia, será preenchida pelo filtro no evento de ação finalizada do usuário.

Figura 31 – Diagrama de atividades para o fluxo de execução do DATD



Para cada máquina μ em M_0 são criadas as variáveis abaixo:

- q_a: o estado correntemente ativo de μ, cujo valor inicial é q_0;
- Φ: uma lista dos p atos dialogais φ_{sk}^{t}de saída, inicialmente vazia;
- E: uma lista das expectativas $e_{k,l}^{t}$ de μ sobre a próxima ação do usuário, inicialmente vazia;
- K: aponta para a transição pendente de resposta do usuário, se esta existir. Inicialmente vazia. Desta forma, apenas um ato dialogal por submáquina pode

requerer resposta do usuário – o que ajuda a resolver o problema mencionado na situação 6;

- ζ_μ: lista de entradas da submáquina. $\zeta_\mu \in \zeta$;
- *floor*: variável que guarda o valor do foco requisitado pela última regra de μ. Este mesmo nome é utilizado para esta variável no Ravenclaw (BOHUS; RUDNICKY, 2009).

7.4.2 Seleção de submáquina

A seleção de submáquina é a atividade responsável pela escolha da próxima submáquina a realizar o processamento, que se dá através dos seguintes passos:

1. Para cada submáquina μ em M_a, solicita-se que informem quais os atos dialogais $\varphi^t_{i,j,k} \in \zeta$ que podem processar e o valor de um indicador i_ζ da *distância* entre as expectativas atuais da submáquina e a regra que irá processar o ato dialogal. Esses valores são calculados da seguinte forma:
    a. Atribui-se o valor 1 a i_ζ da submáquina.
    b. Atribui-se o estado de *binding* atual da submáquina a uma variável v_q.
    c. Verifica-se em μ se existe uma regra, iniciando no estado de *binding* atual da submáquina, cuja condição π pode ser satisfeita apenas com variáveis e atos dialogais contidos em ζ. Se tal regra existe, retorna-se para o DATD i_ζ e ζ_c com os atos dialogais de π.
    d. Se não existe tal regra, executa-se a regra com condição vazia, se esta existir. Se não existir, retorna-se ζ_c vazio com i_ζ igual a ∞.
    e. Para prevenir a entrada em um loop interminável, verifica-se se o novo estado de *binding*, após o passo *d*, é igual à variável v_q. Se for, retorna-se ζ_c vazio com i_ζ igual a ∞.
    f. Incrementa-se i_ζ e a execução continua no passo c.
2. Seleciona-se a submáquina com menor i_ζ para execução.
    a. Se o menor i_ζ é igual a ∞:
        i. Insere-se em ζ um ato dialogal de função comunicativa reservada *noRuleFound*.
        ii. Repete-se a atividade de seleção de máquina. Agora, uma submáquina chamada de *GerenciamentoDeErro*, que faz parte

do conjunto padrão de regras, irá se candidatar para tratar todos os atos dialogais em ζ, pois possui ao menos uma regra fazendo o *binding* com ato dialogal de função comunicativa *noRuleFound*.

b. Se persistir o empate entre duas submáquinas, aquela que estiver mais próxima do início em M_a será escolhida

A descrição da submáquina *GerenciamentoDeErro* será feita no capítulo 8 - Resultados.

7.4.3 Chamada de submáquina selecionada

A ativação de uma submáquina é realizada após a seleção de submáquina, e é responsável por colocar o $\mu \in M_a$ selecionado para execução de regras, através dos seguintes passos:

1. Se é a primeira vez que μ é selecionada neste turno:
    a. Removem-se todos os atos dialogais em Φ.
    b. Removem-se todas as expectativas em *E.*
2. ζ_μ é esvaziado para então receber os atos dialogais em ζ_c de μ
3. Verifica-se se μ possui transição pendente em *K*. Se existe:
    a. Verifica-se a existência de ação adaptativa posterior α_p em K. Se existe, é executada.
    b. Atribui-se para q_a o estado destino da transição.
    c. Verifica-se se ζ_μ está contido em *E*. Se estiver, removem-se os atos dialogais em Φ e as expectativas em *E* que foram gerados por *K*. Este passo existe para viabilizar a situação 5 sem comprometer a situação 8.
    d. Remove-se a transição pendente de K.
4. Atribui-se à variável *floor* o foco na submáquina atual.

7.4.4 Seleção de regra

A seleção de regra escolhe uma transição $\gamma_{i,\pi}$ que se inicie no estado q_a e cuja condição π possa ser satisfeita utilizando-se apenas as variáveis x ∈ X e ζ_μ. É possível, embora não desejado, que exista mais de uma transição $\gamma_{i,\pi}$ cujas condições sejam satisfeitas simultaneamente. Neste caso, a transição com a condição com mais

restrições deve ser selecionada. Se existir uma regra cuja condição é a cadeia vazia {ε}, esta tem a menor prioridade, pois é a menos restritiva.

7.4.5 Execução de regra

A regra selecionada é executada através dos seguintes passos:

1. Executa-se a ação adaptativa anterior α_a de $\gamma_{i,\pi}$, se existir.
2. Se o foco da transição é no usuário:
    a. Atribui-se $\gamma_{i,\pi}$ a *K*.
    b. Executa-se a tarefa β de $\gamma_{i,\pi}$.
3. Se o foco da transição não é no usuário:
    a. Executa-se a tarefa β de $\gamma_{i,\pi}$.
    b. Executa-se a ação adaptativa posterior α_p de $\gamma_{i,\pi}$, se existir.
    c. Muda-se o estado de μ, isto é, atribui-se q_j de $\gamma_{i,\pi}$ a q_a.
4. Atribui-se à variável *floor* de μ o foco definido na transição.
    a. Se o foco é na submáquina, a execução continua com a seleção de uma nova regra
    b. Se não, o processamento de ζ_μ por μ é encerrado.

A execução da tarefa β depende de seu tipo. As tarefas de requisição e de informação ao usuário requerem os seguintes passos:

- Insere-se em Φ os atos dialogais φ_{sk}^t definidos em β.
- Se q_j de $\gamma_{i,\pi}$ é um estado de *binding*, obtém-se todas as transições que iniciem neste estado, cujas condições π sejam diferentes de cadeia vazia. Para cada transição obtida:
    a. Inserem-se os atos dialogais de π em *E*.
    b. Associa-se a esses atos dialogais o grau de confiança c_μ da transição, se definido.

A tarefa de sistema é mais simples, pois consiste em executar as instruções em β de forma direta. As instruções podem fazer consultas na memória semântica e armazenar o resultado em variáveis da submáquina, atualizar a memória semântica com valores de variáveis e enviar comandos de integração com sistemas e módulos externos diretamente ou através do quadro negro.

A tarefa de chamada de submáquina é feita da seguinte forma:

1. Localiza-se se a submáquina μ' sendo chamada
    - Se já está em M_a, nada é feito.
    - Se ainda não foi inicializada, é feita sua inicialização, da mesma forma já descrita para as submáquinas em M_0 (subtópico 7.4.1).
    - Se está inativa, ela é reativada, sendo inserida de volta em M_a.
2. Atribui-se ao ζ_c da submáquina μ' exatamente os mesmos atos dialogais de ζ_c da máquina μ.
3. A execução de μ' é agendada para iniciar imediatamente após o término da execução da submáquina corrente μ.

7.4.6 Loop de seleção de submáquina

Esta atividade verifica se o turno do usuário já foi completamente processado e se todas as requisições anteriores do sistema foram atendidas (situação 10). Se as condições foram atingidas, então termina-se a execução do DATD, e as saídas e expectativas da submáquina são retornadas como resposta à camada de filtro.
O processamento é feito através dos seguintes passos:

1. Se a submáquina μ não gerou nenhuma saída (isto é, Φ está vazio) e seu estado atual q_a é um estado de aceitação, então μ é inativada, isto é, removida de M_a.
2. Verifica-se se existe uma submáquina μ' que foi chamada pela última submáquina μ.
    a. Se existir, então ela é colocada como a submáquina selecionada, o processo de loop de seleção de submáquina é encerrado e o processo de chamada de submáquina é executado.
3. Remove-se de ζ todos os atos dialogais de ζ_μ última submáquina executada.
4. Verifica-se se ζ está vazia.
    a. Se não estiver, a atividade de loop de seleção de submáquina é encerrada e a atividade de seleção de submáquina é executada.
5. Verifica-se se existe alguma submáquina com transição pendente em K referente ao turno anterior (a_s^{-1}). Se existir:
    a. Se neste turno foi processado ao menos um ato dialogal com função comunicativa reservada *noRuleFound*, *noUnderstanding*,

doubtfulUnderstanding ou *pendingSystemRequisitions*, o fluxo continua normalmente no passo 6.

b. Se não, adiciona-se a ζ um ato dialogal de função comunicativa reservada *pendingSystemRequisitions* com item dialogal contendo as transições pendentes, encerra-se a atividade de loop de seleção de máquina e executa-se a atividade de seleção de submáquina, permitindo que uma submáquina se candidate a resolver a situação 10.

6. Verifica-se se existe ao menos uma submáquina que gera expectativas sobre a resposta do usuário, isto é, com *E* contendo ao menos um ato dialogal. Se não temos a situação 20, os seguintes passos devem ser executados:
   a. Verifica-se se existe ao menos uma submáquina com *floor* igual a foco neutro
      i. Se existir uma única, executa-se a atividade de chamada de tal submáquina, para que ela mantenha a iniciativa do sistema.
      ii. Se não existir, continua o fluxo normalmente em 7 – o que pode levar a uma situação em que nem o usuário nem o sistema tentam manter a iniciativa.
      iii. Se existirem duas ou mais, então executa-se a chamada da submáquina mais próxima do início de M_a
7. Finalmente, todas as verificações necessárias para o processamento do turno do usuário foram atingidas. As saídas de todas as submáquinas (atos dialogais em Φ e expectativas em *E*) processadas neste turno são retornas à camada de filtro e o processamento do DATD é encerrado.

8 RESULTADOS

O Adaptalker foi utilizado para desenvolver um pequeno gerenciador ilustrativo, para um sistema de venda de pizzas por telefone. Durante este desenvolvimento, seguiu-se o princípio de prover máxima reutilização de código-fonte e das regras produzidas, de forma a construir uma primeira versão de um arcabouço (ou *framework*) para desenvolvimento de gerenciadores adaptativos de diálogo, ainda que bastante rudimentar.

8.1 Proposta do gerenciador ilustrativo

O principal objetivo do GD de venda de pizzas foi poder verificar se os critérios listados no capítulo 6 são satisfeitos pelo modelo, em situações de diálogo reais ou muito similares a situações reais. Tal tema foi escolhido por ser um diálogo corriqueiro do dia-a-dia do morador da cidade de São Paulo que é orientado a uma tarefa específica e por existirem disponíveis na Internet algumas ontologias básicas de pizzas em OWL. Não foram desenvolvidos os demais módulos que permitiram ter um sistema de diálogo falado completo, assim como não foram desenvolvidos testes utilizando um mágico-de-oz. As entradas foram definidas manualmente para as situações que se desejava testar e a execução foi acompanhada passo-a-passo, com o resultado sendo comparado aos testes de mesa criados ainda durante a concepção e modelagem do Adaptalker.

8.2 Estrutura básica do GD desenvolvido

A aplicação desenvolvida contém as seguintes partes:

- Uma pequena ontologia em OWL para venda de pizzas por telefone, criada e mantida utilizando-se o editor Protégé (TUDORACHE; VENDETTI; NOY, 2008).
- Código-fonte das interfaces de dados trocados entre o GD e os demais módulos, o que inclui interfaces para eventos, hipóteses, segmentos funcionais, atos dialogais e itens semânticos.
- Código-fonte do DATD, incluindo um carregador da ontologia OWL e das regras de tomada de decisão a partir de arquivos e uma interface para poder fazer

pesquisas, inferir conhecimento e atualizar a ontologia através do OWL API e de um *reasoner*. Durante a implementação, foi utilizado o *reasoner* Hermit (SHEARER; MOTIK; HORROCKS, 2008), embora outros pudessem ter sido utilizados.

- Código-fonte do filtro.
- Declaração das regras do DATD, contendo regras específicas de venda de pizzas e submáquinas que possam ser reutilizadas em outras aplicações.

Para desenvolver o DATD, é necessário definir *M*, *O* e Ψ_s conforme a expressão (17). Eles podem ser extraídos de modelos de diálogo, isto é, exemplos de diálogo que ilustram as situações que o sistema deve atender. É importante ressaltar que diálogos entre dois humanos não podem ser a única forma de se obter essas regras, pois o relacionamento é diferente daquele em uma iteração entre homem e sistema.

Foram utilizados dez diálogos curtos para a extração do conjunto inicial de *O* e Ψ_s. As trocas do diálogo em texto em linguagem natural foram substituídas por um texto estruturado, com notação similar àquela descrita no capítulo 4. A Figura 32 contém um trecho que exemplifica o trabalho. Perceba que a diferença reside na não existência de múltiplas hipóteses. A notação "S1.1.1", por exemplo, identifica "*Self-Introduction*" como primeiro ato dialogal do turno 1 e segmento 1 do sistema. A notação "S2.1.1" identifica "*AutoPositive (U1.2.1)*" como primeiro ato dialogal do turno 2 e segmento 1 do sistema, em que parâmetro da função comunicativa indica que o *feedback* se refere ao primeiro ato dialogal do primeiro turno e segundo segmento do usuário.

Figura 32 – Trecho de diálogo que exemplifica a notação utilizada para extração de atos dialogais

Turno	**Linguagem Natural**	**Estruturado**
a_s^1	Pizzaria do Fulano, Boa noite, com quem falo?	S1.1.1 *Self-Introduction* + S1.2.1 *Initial-Greeting* + S1.3.1 *Wh-Question* (*nome*, Usuario *hasName* nome)
a_u^1	Boa noite, meu nome é Ciclano, gostaria de fazer um pedido.	U1.1.1 *Return-Greeting* + U1.2.1 *Self-Introduction* + 1.2.2 *Answer* (S1.3.1) + U1.3.1 *Inform-Intention* (*MakeOrder*)
a_s^2	Olá Ciclano, você já é nosso cliente?	S2.1.1 *AutoPositive* (U1.2.1) + *Check-Question* (Usario SubClassOf *Client*)

Nota-se que algumas das funções comunicativas utilizadas no exemplo não fazem parte da norma ISO 24617-2 ou do DIT++. É o caso do *Inform-Intention*, que foi criado para capturar a intenção do usuário em relação ao sistema e que tem como item semântico o ato dialogal (completo com item semântico ou incompleto, como é o caso na Figura 32) representando a intenção. *MakeOrder* é a função comunicativa para fazer o pedido de um produto, também um caso especial da função *Inform*. *Inform-Intention* poderia ser usado também com atos dialogais de função comunicativa *question*, como quando se tem uma dúvida, que é o caso de "Gostaria de fazer uma pergunta".

De posse de *O* e Ψ_s, pode-se iniciar a definição das submáquinas através das trocas, isto é, relacionando os atos dialogais do usuário $\varphi_{i,j,k}^{t}$ aos atos dialogais de resposta do sistema φ_{sk}^{t}, o que irá permitir a modelagem das regras de tomada de decisão. O relacionamento inverso, isto é, de φ_{sk}^{t} para $\varphi_{i,j,k}^{t}$ permitem mapear as expectativas. É importante anotar eventuais relações entre as trocas, tais como pré-requisitos, bloqueadores e preferência por uma determinada ordem.

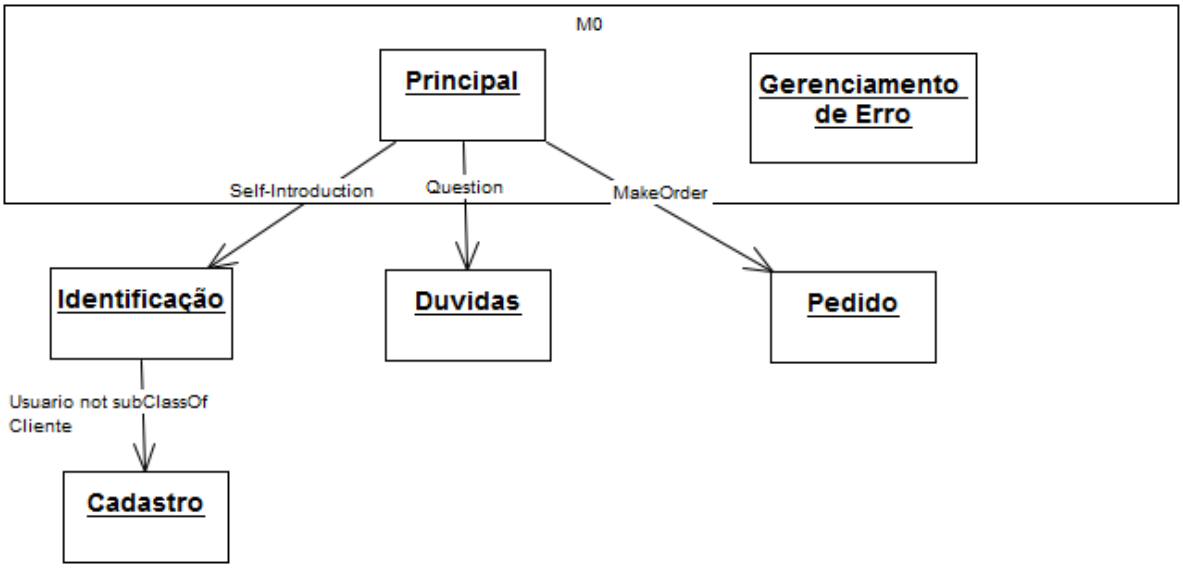
8.3 Projeto de submáquina

Para decidir quais regras devem ser colocadas em uma mesma submáquina, deve-se procurar regras que são requisitos para a execução de uma mesma tarefa, possivelmente com os mesmos pré-requisitos e bloqueadores. Para venda de pizzas, é possível agrupar as trocas em alguns grupos: cumprimento e identificação, cadastro de usuário, dúvidas, pedido e encerramento. As trocas de cumprimento e de encerramento foram colocadas na mesma submáquina, chamada de *principal*, que está em M_0. Outra submáquina em M_0 é a de gerenciamento de erro, que trata atos dialogais com as funções comunicativas *noRuleFound*, *noUnderstanding*, *doubtfulUnderstanding* e *pendingSystemRequisitions*. A Figura 33 mostra essas submáquinas e as relações entre elas. Por exemplo, a identificação é pré-requisito para pedido. A submáquina de cadastro é ativada a partir da identificação. As demais submáquinas são ativadas a partir de chamadas na submáquina principal.

Como exemplo, detalha-se a submáquina de cadastro, cuja configuração inicial que pode ser visualizada em forma gráfica e simplificada na Figura 34. Várias transições

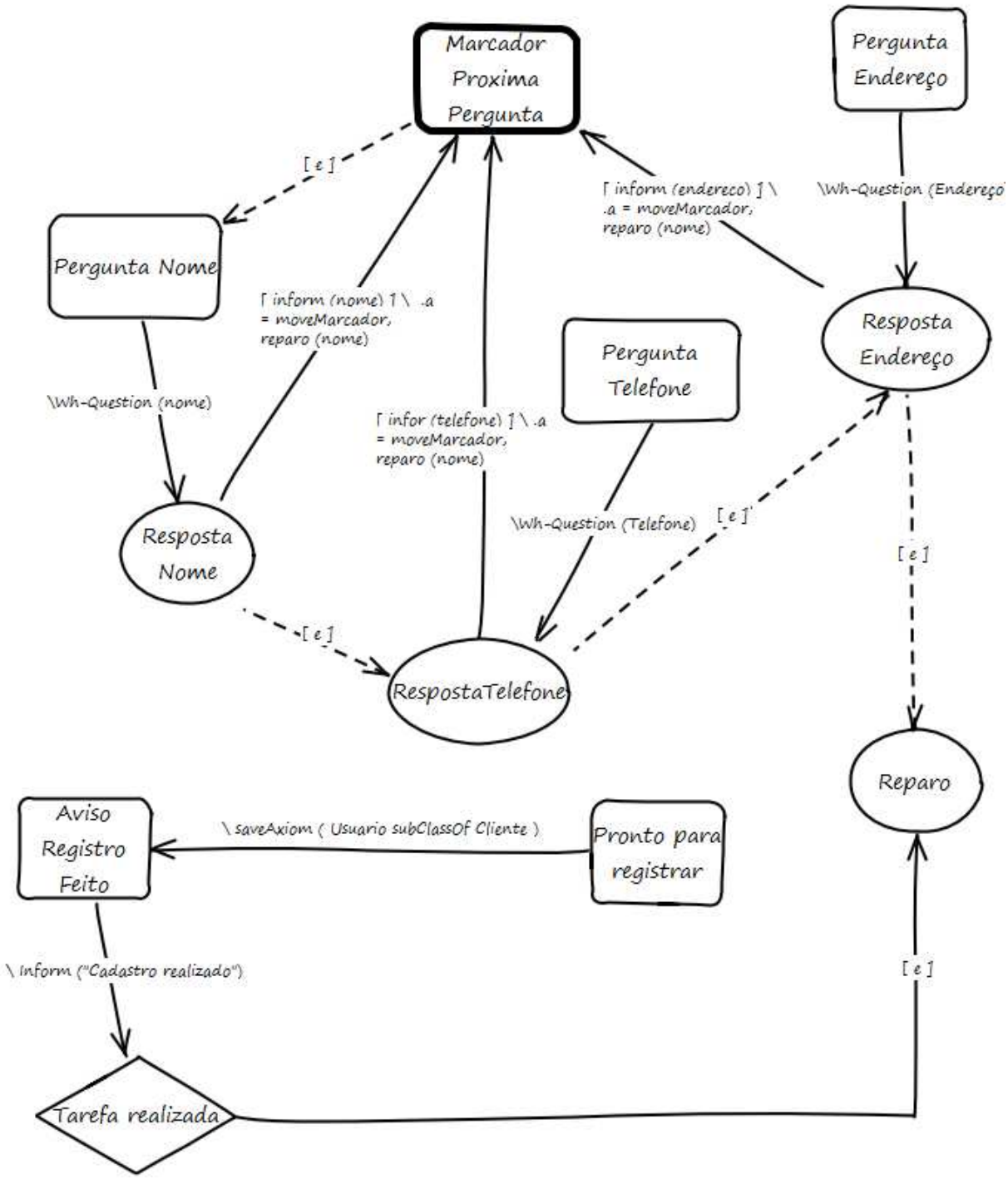
contendo caminhos alternativos de resposta do usuário e regras de validação não foram exibidas para manter a visualização adequada. A lógica utilizada vale para a construção de qualquer outra tarefa que requeira que o usuário forneça um conjunto de informações que não possuem dependência semântica entre si. Esta lógica é descrita mais detalhadamente no Apêndice B deste trabalho.

Figura 33 – Submáquinas do GD de venda de pizza. As setas indicam que a submáquina na origem ativa aquela no destino da seta.



A submáquina de cadastro tem como estado inicial um estado marcador, identificado como “Marcador Próxima Pergunta”. Este estado sempre tem uma única transição incondicional, sem tarefa ou ação adaptativa, apontando para a “próxima pergunta”. Toda vez que esta “próxima pergunta” é respondida, altera-se o estado destino desta transição, que inicia no estado “Marcador Próxima Pergunta”, para uma nova “próxima pergunta”, através de uma chamada à função adaptativa “moveMarcador”. A “próxima pergunta” é uma estrutura formada por uma transição incondicional cuja tarefa é do tipo requisição com um ato dialogal de função comunicativa *Wh-Question*, isto é, causará uma ação do sistema que solicita ao usuário o fornecimento da informação. O estado de origem sempre tem seu nome formado pela *string* “Pergunta” seguido pelo nome da informação que deseja obter. O estado destino sempre tem seu nome formado pela *string* “Resposta” seguido pelo nome da informação que se deseja obter.

Figura 34 – Diagrama com forma gráfica simplificada da configuração inicial da submáquina de Cadastro de Cliente. O Apêndice A contém um esclarecimento sobre a notação gráfica utilizada.



No exemplo, temos três informações que precisam ser obtidas antes do cadastro do cliente: “nome”, “telefone” e “endereço”.

Também existe um estado marcador de reparo. Ele irá acumular transições que permitem corrigir, a qualquer momento, informações fornecidas na própria submáquina. Estas transições são adicionadas de forma adaptativa por chamadas à função “reparo”.

Existem transições incondicionais ligando os estados de *binding* entre si. São estas transições que permitem que o usuário tenha iniciativa, pois a submáquina irá tentar vincular os atos dialogais do usuário em todos os estados de *binding* interligados de forma incondicional (ver “Seleção de submáquina” no algoritmo de execução para mais detalhes), mesmo para aqueles atos dialogais que não estavam entre as expectativas.

Finalmente, existe a tarefa que só pode ser executada após todas as informações serem fornecidas e que está, na configuração inicial, em uma posição inalcançável. A função adaptativa “moveMarcador” altera o estado destino da transição partindo de “Marcador Próxima Pergunta” para o estado “Pronto para registrar” quando todas as perguntas tiverem sido respondidas.

Para o entendimento ficar completo falta, ainda, a descrição da função adaptativa “moveMarcador”. Ela tem duas responsabilidades: alterar a “próxima pergunta”, como já mencionado, e remover, da sequência de estados de *binding* interligados por transição incondicionais, os estados que fazem o encaixe das respostas já obtidas. Utilizando (20), define-se a função “moveMarcador” como:

- Λ = (transição λ_1), em que λ_1 deve ter como condição um ato dialogal que forneça uma informação requisitada para a execução da tarefa, por exemplo, *inform* (Usuario *hasName* nome);
- V = {transição proximoEstadoBinding, transição proximaQuestao, transição marcadorProximaQuestao, transição perguntaAtual};
- G é um conjunto vazio, isto é, não são utilizados geradores;
- A_f contém as seguintes ações elementares:
    1. Consulta (proximoEstadoBinding, μ, $\lambda_1.q_i$, ?, ε, ε);
    2. Consulta (proximaQuestao, μ, ?, proximoEstadoBinding.q_j, ε, *Wh-question*);
    3. Consulta (marcadorProximaQuestao, μ, “Marcador Proxima Pergunta”, ?, ε, ε);
    4. Consulta (perguntaAtual, μ, marcadorProximaQuestao.q_j, $\lambda_1.q_i$, ε, *Wh-question*);
    5. Compara (perguntaAtual, *null*, removePerguntaSequencia (λ_1, proximoEstadoBinding), moveMarcadorProximaPergunta (proximoEstadoBinding, marcadorProximaQuestao, proximaQuestao)).

A última ação elementar verifica se a variável perguntaAtual é nula. Se for, chama a função adaptativa removePerguntaSequencia. Novamente utilizando (20), ela pode ser definida como:

- Λ = (transição encaixaInformaçãoAtual, transição encaixaProximaInformação);
- V = {transição encaixaInformaçãoAnterior};
- G é vazio;
- A_f contém as seguintes ações elementares:
    1. Consulta (encaixaInformaçãoAnterior, μ, ?, encaixaInformaçãoAtual.q_i, ε, ε);
    2. Altera (encaixaInformaçãoAnterior.q_j = encaixaProximaInformação.q_i).

Se a variável perguntaAtual não é nula, utiliza-se a função adaptativa moveMarcadorProximaPergunta, que pode ser definida como:

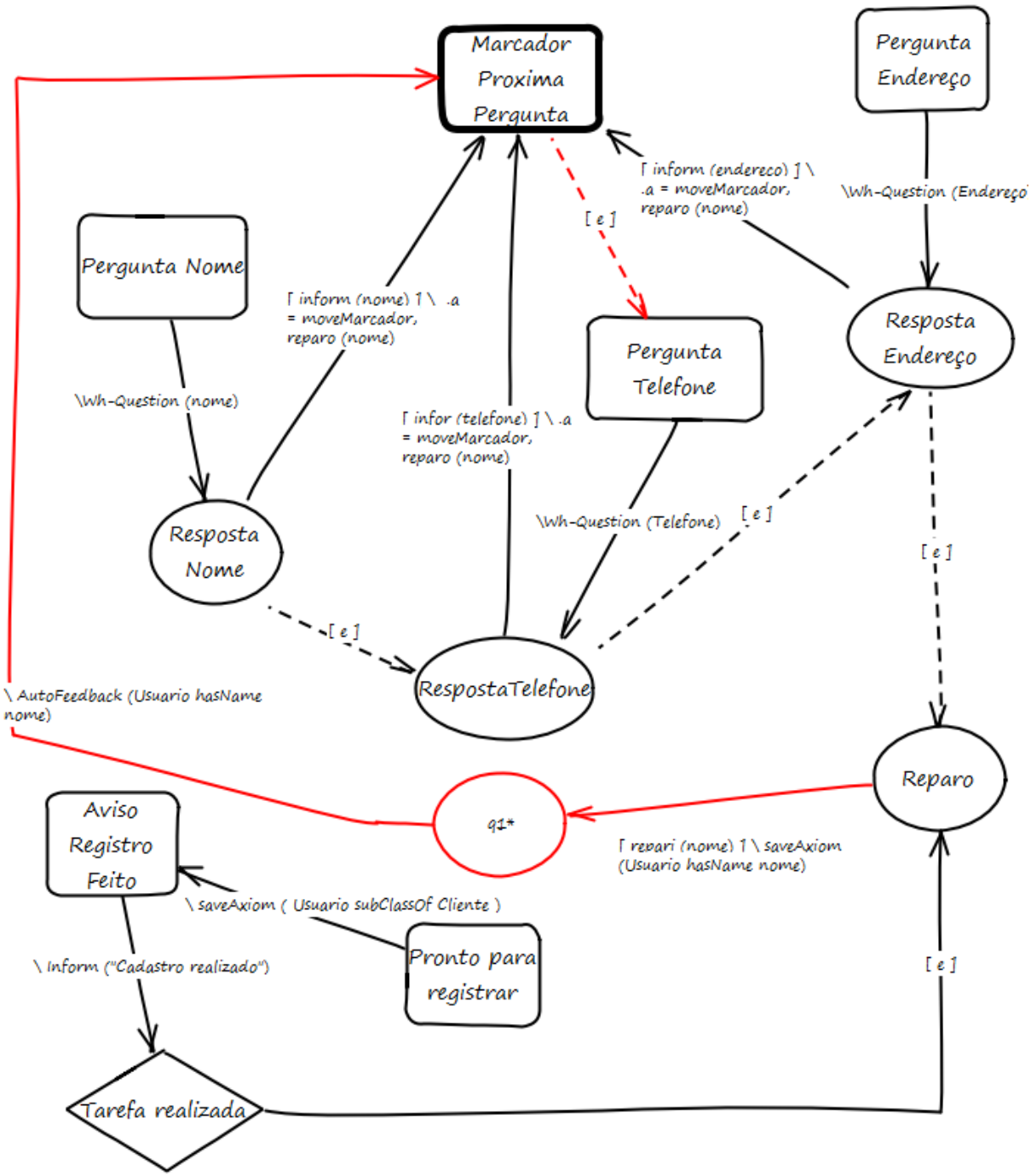
- Λ = (transição encaixaProxima, transição marcadorProximaQuestao, transição proximaQuestão);
- V é vazio;
- G é vazio;
- A_f contém as seguintes ações elementares:
    1. Compara (encaixaProxima.qj, “Reparo”, Altera (marcadorProximaQuestao.q_j = “Pronto para registrar”), Altera (marcadorProximaQuestao.q_j = proximaQuestão.q_j).

Outra função adaptativa importante é a que adiciona uma transição para reparo anterior. Ela é definida como:

- Λ = (ObjectPropertyAxiom propriedadeAatualizar, valor);
- V é vazio;
- G = {estado q*};
- A_f contém as seguintes ações elementares:
    1. Adiciona (μ, ”reparo”, q*, (*repair* || *inform* (nome, Usuario *hasName* nome)), ε, ε, ε, propriedadeAatualizar (valor), foco na submáquina);
    2. Adiciona (μ, q*, “Marcador Proxima Pergunta”, ε, ε, ε, ε, *AutoFeedback* (Usuario *hasName* nome), foco neutro).

8.3.1 Exemplos de execução da submáquina de cadastro

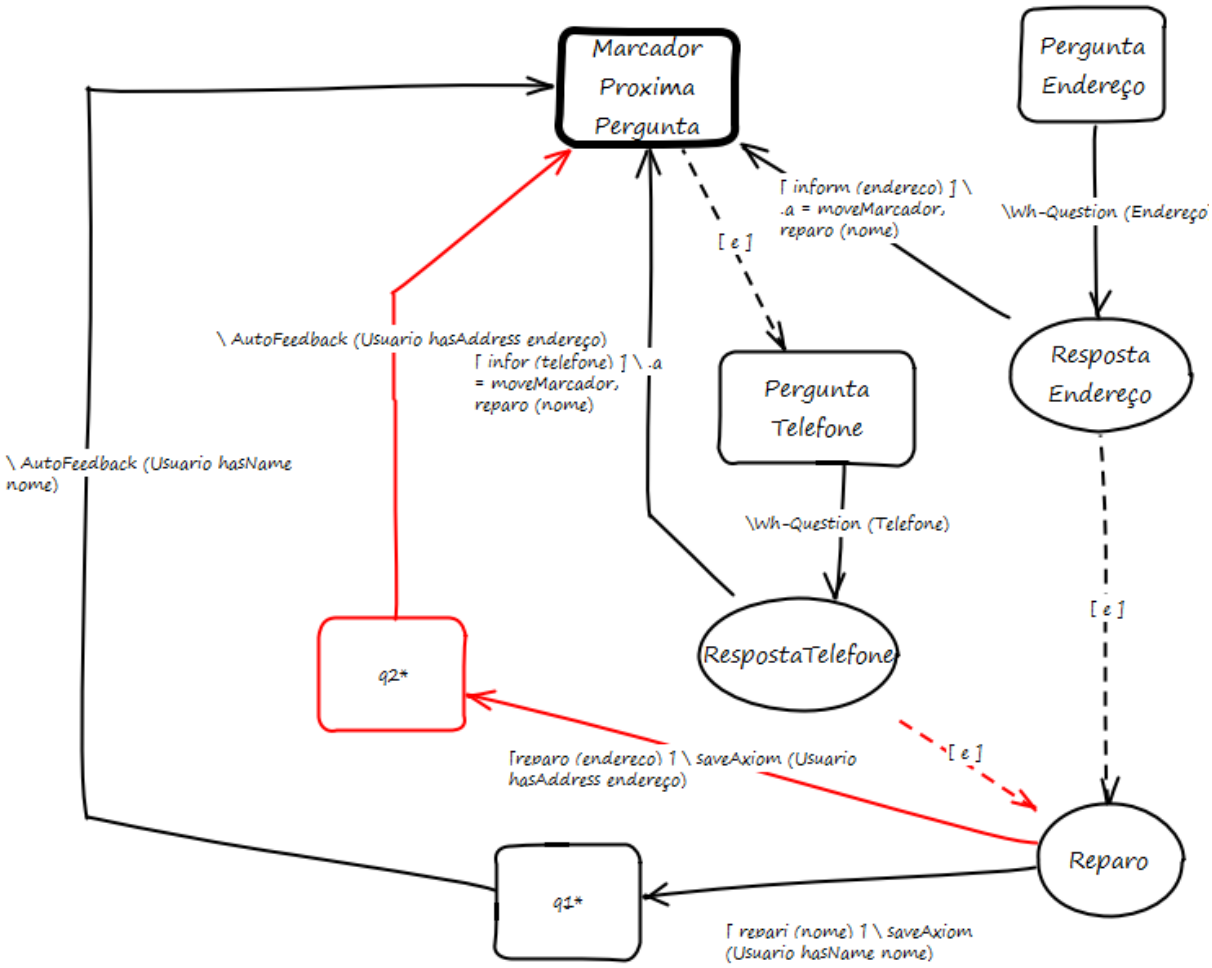
Figura 35 - Diagrama com forma gráfica simplificada da submáquina de Cadastro de Cliente após o usuário fornecer seu nome.



A submáquina de cadastro, a partir de seu estado inicial “Marcador Proxima Pergunta”, inicia sua execução com a regra de tarefa de requisição ao usuário *Wh-Question* (nome, Usuario *hasName* nome). Em um primeiro exemplo, o usuário

responde com “Meu nome é Ciclano”, que é um segmento funcional com dois atos dialogais, *Answer* (relacionando o segmento funcionalmente a pergunta anterior) e *Inform* (Usuario *hasName* Ciclano). A transição selecionada é a que tem q_i = “Resposta nome” e q_j = “Marcador Proxima Pergunta”. Ambas as funções adaptativas “moveMarcador” e “reparo” serão executadas. A configuração da submáquina após a execução desta submáquina pode ser vista na Figura 35. É fácil observar que após cada pergunta respondida pelo usuário, que nunca toma a iniciativa, acontece uma mudança semelhante na configuração da submáquina, até que a tarefa possa ser concluída.

Figura 36 - Diagrama com forma gráfica simplificada da submáquina de Cadastro de Cliente após o usuário fornecer seu nome e endereço. Alguns estados foram suprimidos para reduzir o tamanho da figura.



Em um exemplo da situação 5, o usuário não responderia apenas com seu nome, mas com seu nome e telefone no mesmo turno. Devido à verificação feita pelo algoritmo

de execução do DATD na chamada de submáquina, é detectado que o fornecimento do telefone era uma das expectativas causadas pela pergunta *Wh-Question (Telefone)*, que é então removida de *Φ*, com todas as suas expectativas. Ao final do processamento da submáquina, *Φ* conterá a pergunta *Wh-Question* (endereço).

Em um exemplo da situação 8, o usuário responde “Meu nome é Ciclano e meu endereço é Rua dos Ciclaninhos, número 23”, isto é, responde com dois segmentos funcionais, o primeiro com dois atos dialogais *Answer* e *Inform* (Usuario *hasName* Ciclano) e o segundo com um único ato dialogal *Inform* (Usuario *hasAddress* “Rua dos Ciclaninhos, número 23”). Após processar o primeiro segmento funcional, a configuração da submáquina será a mesma da Figura 35. Ao processar o segundo segmento, a transição incondicional ligando os estados “Resposta Telefone” e “Resposta Endereço” será selecionada para execução, pois *Inform* (endereço) não está entre as expectativas causadas pela pergunta *Wh-Question* (Telefone). Em seguida, a regra ligando os estados “Resposta Endereço” a “Marcador Proxima Pergunta” será selecionada, causando o acionamento das funções “moveMarcador” e “reparo”. A configuração após este processamento pode ser vista na Figura 36. Repare que desta vez o marcador de próxima pergunta não é movido, pois o telefone ainda não foi fornecido; entretanto, o estado “Resposta endereço” é removido da sequência de estados de *binding* interligados por transições incondicionais, pois a informação já foi fornecida.

8.3.2 Comentários sobre outras construções possíveis para a submáquina de cadastro

Uma característica importante do Adaptalker é que ele permite que o desenvolvedor possa utilizar diferentes conjuntos de regras para resolver o mesmo problema, resultando em comportamentos levemente distintos do sistema. No caso da submáquina de cadastro, existem ao menos duas outras maneiras de construí-la:

- Utilizando variáveis de controle de preenchimento de informação ao invés da função adaptativa “moveMarcador”. Estas variáveis seriam utilizadas como condições das transições que requisitam informação ao usuário. Esta técnica de construção é mais parecida com as técnicas utilizadas em outros gerenciadores de diálogo não adaptativos, como o Ravenclaw. O Adaptalker não obriga o desenvolvedor a utilizar a adaptatividade;

- Alterando a função adaptativa “reparo” para colocar as transições de reparo em uma submáquina exclusiva de reparo, de tal forma que o reparo seja possível mesmo após a submáquina de cadastro ser desativada. Se necessário, deve-se utilizar a memória semântica para passar informações de uma submáquina para outra, pois o escopo das variáveis é limitado à sua submáquina de origem. Esta solução é ainda mais flexível para o usuário, que pode corrigir suas informações cadastrais a qualquer momento do diálogo, mas, em diálogos que requisitam muitas informações ao usuário, a submáquina de reparo pode crescer em demasia.

8.4 Utilização da adaptatividade

A adaptatividade é utilizada para adicionar algumas capacidades ao modelo baseado em máquinas de estado:

- Permitir ao usuário ter a iniciativa: o usuário pode antecipar e informar o sistema sobre coisas que ele ainda não pediu (situações 5, 8 e 9). Ambas as situações 5 e 8 já foram exemplificadas.
- Permitir a realização do reparo tanto pelo usuário quanto pelo sistema nas terceiras e quartas posições (situações 15, 16, 18 e 19) sem necessidade de retrocesso para estados anteriores. A utilização de adaptatividade para permitir o reparo das informações já foi exemplificada, embora o reparo ele próprio ainda não tenha sido. A execução do reparo não requer adaptatividade.
- Permitir ao sistema retomar um assunto previamente postergado (situação 12).
- Permitir lidar com trocas de diálogo sensíveis ao contexto.

Um exemplo deste último caso é o acionamento da submáquina de pedido, que só pode ser permitido após se obter a identificação do usuário. Uma solução apropriada é, na configuração inicial da submáquina principal, ter uma transição cuja condição π contenha o ato dialogal “*inform-intention* (*make-order*)“ que leva à tarefa de requisição que informa ao usuário “Para fazer um pedido, por favor identifique-se primeiro”. Após a obtenção da identificação, uma ação adaptativa troca a tarefa mencionada por uma outra, de chamada à submáquina *pedido*.

É interessante ressaltar que as funções adaptativas se tornam consideravelmente mais complexas em situações sensíveis ao contexto. Suponha que após o sistema ter

identificado o usuário, este perceba que há um erro no entendimento do sistema relativo ao seu nome e queira repará-lo no meio do registro do pedido. Como proceder nesta situação? A situação mais adequada parece ser a remoção da submáquina de pedido de M_a, mas mantendo sua configuração, até que a identificação do usuário tenha sido claramente obtida novamente. Mas perceba que a função de reparo fornecida previamente já não resolveria este problema, pois agora ela precisa, além de realizar o reparo da informação, bloquear o acesso a uma outra submáquina inteira. Para ser capaz de retomar um assunto, pode-se alterar ou adicionar uma transição do DATD que permita relembrar o assunto que esteja sendo postergado. Por exemplo, a submáquina de pedido possui uma regra com origem em q_0 solicitando que o usuário faça o pedido através do ato comunicativo *request* (*makeOrder*). Se, em algum turno, antes de a identificação ser obtida, o usuário manifestar interesse em alguma pizza específica através do ato dialogal *makeOrder* (x, x *subClassOf* Pizza), pode-se utilizar uma ação adaptativa para substituir o ato dialogal *request* (*makeOrder*) por um *check-question* (Usuario *makeOrder* x, x *subClassOf* Pizza).

8.5 Avaliação do Adaptalker conforme critérios

Dos critérios extraídos da literatura, já foi evidenciado que o Adaptalker é capaz de lidar com múltiplas hipóteses paralelas, iniciativa mista, sequenciamento do diálogo e suporte ao reparo. Modificadores de emoções e de prosódia podem ter seus valores consultados diretamente no quadro negro, se tais indicadores estiverem presentes, e comparados com valores específicos das condições π das transições. O último critério, facilidade de desenvolvimento, é discutida detalhadamente no capítulo 9 - Discussão.

Das várias situações listadas que devem ser suportadas, já foram evidenciadas as situações 4, 5, 8, 12 e situações de reparo pelo usuário (13, 15 e 18) quando a informação não tem dependência semântica com outras informações.

A situação 1 é detectada pelo algoritmo da camada de filtro, que gera um ato dialogal de função reservada *noUnderstanding* que é tratado pela submáquina de gerenciamento de erro. As situações 2 e 3 também são detectadas pelo algoritmo da camada de filtro. A situação 3 gera um ato dialogal de função *doubtfulUnderstanding* se a dúvida persistir, que também é tratado pela submáquina de gerenciamento de erro.

A situação 6 não é evitada pelo GD, embora não seja possível ter duas transições pendentes na mesma submáquina. Da forma que foi construído, ter pendências em máquinas diferentes é uma situação que funciona corretamente no DATD. Entretanto, se houver dependência semântica ou adaptativa entre duas transições pendentes, podem acontecer situações problemáticas não tratadas pelo algoritmo. É o que pode acontecer se uma transição pendente, ao ser executada, tentar remover a outra ainda em *K*. Uma boa prática ao desenvolver um GD com o Adaptalker é, portanto, não colocar ações adaptativas em ações com tarefas que requisitam informações ao usuário nem remover transições que requisitam informações.

A situação 7 é detectada no loop de seleção de submáquina, o que irá gerar um ato de função *noRuleFound*, que será tratada, mais uma vez, pela submáquina de gerenciamento de erro. A situação 10 é detectada no mesmo processo de loop, o que irá gerar um ato de função *pendingSystemRequisitions* a ser tratado também pela submáquina de gerenciamento de erro. Da forma que foi construída, a submáquina de gerenciamento de erro pode ser reutilizada em outras aplicações, não tendo nada específico de venda de pizzas. Ela também faz o tratamento de atos dialogais com função *partiaUnderstanding* e *timeout*. Esta última é recebida pelo GD quando o usuário ultrapassa a um tempo máximo permitido sem responder a uma solicitação.

As situações 11 e 12 são suportadas pelo Adaptalker, mas não há detecção automática de assunto que deve ser postergado. Essa situação deve ser manipulada pelo desenvolvedor, seguindo as observações já feitas neste mesmo capítulo. Os mesmos comentários podem ser feitos sobre os reparos feitos pelo sistema (situações 14, 16, 17 e 19).

A situação 20 é detectada pelo processo de loop de seleção de submáquina do DATD, que então procura submáquinas com foco neutro para continuar o processamento e mantiver a iniciativa. Se o desenvolvedor, entretanto, não projetar as submáquinas com cuidado, é possível que a conversa estagne, se o usuário não manter a iniciativa (por exemplo, quando não está certo do que fazer) e nenhuma submáquina solicitar a iniciativa.

A situação 21 só pode ser detectada pelo filtro, comparando as probabilidades a posteriori das múltiplas hipóteses paralelas. Entretanto, não houve desenvolvimento para o tratamento desta característica, que deve ser feita em trabalhos futuros.

9 DISCUSSÃO

9.1 Comparação com gerenciadores não adaptativos baseados em máquinas de estados

É possível utilizar o Adaptalker da mesma maneira que um gerenciador não adaptativo baseado em máquinas de estado. Para isto, basta não utilizar as funções adaptativas e manter a iniciativa sempre com o sistema, utilizando, se necessário, retrocessos solicitados pelo usuário para corrigir informações dadas previamente. Ainda assim, outras melhorias introduzidas poderiam ser utilizadas, como múltiplas hipóteses paralelas e as submáquinas. Tratamentos de erros tais como os da situação 1 e 3 poderiam continuar sendo feitos através de uma submáquina de gerenciamento de erro. Desta forma, as mesmas facilidades de modelagem e de teste convencionais continuariam válidas.

Ao utilizar a adaptatividade como proposto, no entanto, cuidados adicionais devem ser tomados no desenvolvimento do aplicativo. Testar um dispositivo com chamadas adaptativas requer um trabalho de análise maior que em um dispositivo sem chamadas adaptativas. Para atingir uma cobertura de testes de 100%, por exemplo, será necessário testar as entradas em todas as configurações atingíveis pela execução de funções adaptativas, combinando as várias submáquinas. Além disso, deve-se testar o comportamento do sistema quando o usuário tem a iniciativa e adianta o fornecimento de informações. Entretanto, se forem tomados alguns cuidados, como minimizar a quantidade de funções adaptativas de uma submáquina que alterem outras submáquinas, pode-se testar as submáquinas separadamente, e somente depois integrá-las para testar especificamente as configurações que somente são atingíveis após chamadas de ações adaptativas de outras submáquinas.

Outra facilidade da adaptatividade é que ela pode ser utilizada para manter estável a quantidade de estados alcançáveis em cada configuração, mesmo com a adição de novos estados, ao se fazer a *poda* (isto é, a remoção) de transições que não devem mais ser utilizadas. Por exemplo, na Figura 34 existem seis estados alcançáveis e na Figura 35 também existem seis alcançáveis, dois dos quais diferentes da configuração anterior. Utilizando-se todas as transições que podem ser disparadas em cada configuração, é possível agrupar as entradas de testes em partições, e aplicar os

testes considerando a combinação das partições ao invés de todos os valores possíveis.

No exemplo da submáquina de cadastro, existem apenas duas funções adaptativas, acionadas em três pontos distintos. A cada acionamento, os parâmetros utilizados são distintos. Estes três pontos podem ser atingidos em qualquer ordem, levando, portanto, a seis possíveis combinações de acionamento. Internamente, essas funções adaptativas não dependem do conteúdo semântico nem de outras submáquinas; dependem apenas da configuração da própria submáquina no momento em que são chamadas. Desta forma, têm-se seis configurações possíveis em que esta submáquina pode ser testada. Quanto à iniciativa, temos as seguintes opções: (i) o sistema mantém a iniciativa até o fim; (ii) o usuário antecipa todas as informações em um único turno; (iii) o usuário fornece duas informações em um primeiro turno; e (iv) o usuário fornece uma informação no primeiro turno e duas no segundo. Algumas configurações só podem acontecer combinadas com algumas situações de iniciativas, por exemplo, (i) e (ii) sempre acontecerão na mesma sequência de configurações, em que as informações são processadas na ordem esperada pelo sistema – apenas um teste é necessário para cada um. Mesmo (iii) e (iv) não podem acontecer em qualquer ordenação, pois a informação “nome” será sempre processada antes das outras se fornecida no mesmo turno que elas. Pode-se demonstrar que, tanto para (iii) quanto para (iv) existem apenas três testes necessários. Sendo assim, existem oito testes básicos que devem ser feitos na submáquina de cadastro. Se houvesse uma quarta informação solicitada nela como, por exemplo, número do documento de registro, seriam necessários dezesseis testes, e se houvessem cinco informações, seriam necessários trinta e dois testes. Aparentemente, a quantidade de testes básicos necessários é exponencial à quantidade de informações que devem ser fornecidas pelo usuário, isto é, igual a 2^n, para submáquinas construídas seguindo a lógica descrita no Apêndice B. Fica também para trabalhos futuros esta demonstração. Devem ser feitos testes também para os reparos de informação pelo usuário, embora as transições de reparo não tenham funções adaptativas associadas. A quantidade de testes de reparo é proporcional à quantidade de informações que podem ser fornecidas, já que a transição de reparo testada é a mesma independente do turno em que é feita.

Se o cadastro fosse feito através de uma submáquina convencional, que possui uma única configuração constante, os testes se resumiriam a uma única sequência, pois o

sistema não permite que o usuário tenha a iniciativa. Entretanto, os testes de reparo seriam mais trabalhosos do que no caso adaptativo, pois requerem que o usuário solicite o retrocesso até o passo em que a informação possa ser corrigida, isto se as transições de reparo não estiverem replicadas a cada novo estado, o que aumentaria consideravelmente a quantidade de transições que precisam ser testadas. No primeiro caso é necessário testar o retrocesso em cada estado. Como o primeiro estado não tem retrocesso (nenhuma informação foi fornecida), apenas os demais têm a opção de retrocesso, sendo necessário, portanto, cinco testes para três informações, sete para quatro e assim por diante – isto é, a quantidade de testes é igual ao dobro do número de informações menos um. Entretanto, os testes seriam mais longos, pois mesmo algumas informações não reparadas precisariam ser fornecidas duas vezes pelo testador dependendo da situação, devido ao retrocesso. No segundo caso, a quantidade de transições iniciando em um mesmo estado é proporcional à distância do mesmo até o estado inicial, isto é, no primeiro estado temos uma transição, no segundo duas e assim por diante. É fácil notar que a soma das transições é a soma de uma progressão aritmética, cujo resultado é *(n+1)*n/2*. Por exemplo, para três informações, teremos seis transições para testar. Se for necessário fazer testes combinatórios, o crescimento é fatorial, isto é, *n!*: um teste para uma informação, dois para duas, seis para três, vinte e quatro para quatro e assim por diante.

Deve-se notar que, se forem consideradas as transições adicionais, suprimidas no diagrama, que chamam tarefas de sistema para validação dos dados fornecidos, a quantidade de testes aumentaria mais ainda. Entretanto, a quantidade destes testes não tem relação com as funções adaptativas e é a mesma para dispositivos adaptativos e não adaptativos.

Existem em outras aplicações práticas várias situações diferentes daquelas consideradas no exemplo ilustrativo de venda de pizzas. Eventualmente, serão necessárias submáquinas com dependências mais complexas entre as informações, que ainda não foram testadas e que vão necessitar de funções adaptativas de maior complexidade, exigindo uma quantidade ainda maior de testes.

A decisão entre usar um modelo adaptativo e um não adaptativo reside na relação entre o custo de desenvolvimento e o benefício do usuário; um sistema em que o usuário pode ter iniciativa e corrigir a informação a qualquer momento é mais amigável do que aquele que não suporta essas situações, mas requer mais tempo de desenvolvimento. Se houver maneiras de automatizar a construção dos dispositivos,

essa diferença no tempo de desenvolvimento pode ser reduzida consideravelmente, como se pode verificar no Apêndice B.

9.2 Comparação com outras técnicas de gerenciamento

A comparação com outras técnicas de gerenciamento é afetada pela escassez de trabalhos criteriosos em relação às situações do diálogo suportadas. Não foi encontrada publicação com trabalho equivalente àquele feito no capítulo seis. Portanto a comparação será feita apenas com os modelos mais estudados e, ainda assim, não será possível a avaliação comparativa da maioria dos critérios.

A organização hierárquica de chamada das submáquinas no Adaptalker pode ser considerada uma solução equivalente àquela do Ravenclaw, que possui uma organização hierárquica das suas tarefas. Esta solução permite reduzir o espaço de busca de regras. Outras similaridades são a iniciativa mista do usuário e suporte ao reparo. O suporte ao reparo do Ravenclaw é feito por um agente da plataforma (isto é, que não precisa ser definido pelo desenvolvedor da aplicação) que é colocado com prioridade máxima no topo da agenda de execução. Existem outros agentes do Ravenclaw, similares a este de reparo, para manipular as mesmas situações tratadas pela submáquina de gerenciamento de erro do Adaptalker.

Existem diferenças importantes entre as duas soluções. O Ravenclaw não é capaz de manipular múltiplas hipóteses simultaneamente, o que é suportado pelo Adaptalker, embora isso não seja devido à adaptatividade. O Ravenclaw permite postergar e retomar assuntos apenas através de personalização de seu comportamento através de codificação em C++, isto é, o dispositivo não é capaz de naturalmente suportar esta condição. Neste ponto, a adaptatividade é vantajosa para o Adaptalker, concedendo ao modelo uma capacidade que lhe dá vantagem relativa a seus concorrentes. Por exemplo, para compartilhar tais regras de tomada de decisão, basta copiar as regras da submáquina, enquanto o outro requer cópia tanto de código-fonte quanto de regras, criando uma dependência entre ambas as partes. O Adaptalker permite resolver os problemas decorrentes de sensibilidade ao contexto através de funções adaptativas, enquanto o Ravenclaw, novamente, necessitará de personalização de seu código-fonte. Existem alguns casos comuns de dependência entre os agentes, tais como pré-condições, em que o desenvolvedor pode utilizar

macros disponibilizadas pela plataforma e que certamente aceleram o tempo de desenvolvimento.

É importante ressaltar, no entanto, que se forem necessárias novas funções comunicativas não existentes no conjunto básico de funções do Adaptalker, será necessário alterar a implementação existente, pois a interpretação dos itens semânticos relacionados é especificada por código-fonte. A interpretação inclui, por exemplo, métodos para comparação com as condições das transições.

Uma característica interessante introduzida pela adaptatividade no Adaptalker, não só em relação ao Ravenclaw, mas mesmo quando comparado a outras técnicas, é que ele permite o uso de *inversão de controle*. Em todos os gerenciadores revisados, sempre são as próprias regras que contêm toda a informação sobre quando ser executadas, quer seja pelo uso de pré-condições ou de gatilhos. Por exemplo, para habilitar a oferta de um produto específico em situações específicas, esta condição precisa ser descrita através da verificação dos valores de variáveis na própria regra que oferece o produto. Mas com a adaptatividade, pode-se substituir tal condição por uma chamada adaptativa que remova ou adicione a transição de oferta, eliminando a necessidade de variáveis de contexto.

Algo importante a ser observado é que não é possível alterar o Ravenclaw para lhe adicionar adaptatividade, pois não há uma descrição formal das regras que ele utiliza – há apenas a descrição da interface de programação e detalhamento de seu algoritmo (BOHUS; RUDNICKY, 2009), além de alguns tutoriais na internet. Entretanto, se houvesse uma formalização destas regras, a adaptatividade poderia ser incorporada para adicionar as características ausentes já mencionadas. O mesmo pode ser dito sobre a utilização da adaptatividade em técnicas baseadas em *frames*.

Durante este trabalho de pesquisa, foram considerados os modelos baseados em MDP e POMDP para o uso de adaptatividade. Entretanto, foram encontradas algumas dificuldades que não favoreceram o prosseguimento desta frente particular da pesquisa. A principal delas foi relativa ao uso de iniciativa mista. Dado que as regras desses dispositivos têm a ação do sistema como um de seus componentes para a escolha do estado de destino (ao invés de ser uma saída), como decidir o estado de destino nas situações em que o usuário antecipa uma informação? Isto é, a tomada de decisão do usuário foi feita antes da tomada de decisão do sistema, que deveria ser irrelevante neste caso. Em uma solução não adaptativa, isso não seria um problema, pois as condições são constantes e pode-se definir uma política de

escolhas que será mantida de forma constante durante toda a execução – a tomada de decisão do sistema em tal estado sempre seria a mesma. Entretanto, uma vez que uma ação adaptativa tenha sido executada, os parâmetros que levaram a decidir por uma política ao invés de outra já não são mais os mesmos, e a escolha anterior deveria ser reavaliada, o que parece ser, inicialmente, uma desvantagem do modelo adaptativo, pois a escolha de uma política é uma atividade que consome muita computação e não deveria ser repetida frequentemente.

10 CONCLUSÃO

Neste trabalho foi apresentado uma revisão literária ampla do gerenciamento do diálogo em sistemas computacionais falados, de forma a refinar os critérios de avaliação dos gerenciadores de diálogo de forma geral, resultando na primeira contribuição importante deste trabalho. Uma técnica para desenvolvimento de gerenciadores adaptativos foi então apresentada e utilizada para criar uma aplicação simples. Esta técnica é a referência para um arcabouço de desenvolvimento de gerenciadores adaptativos de diálogo, que é a segunda e mais importante contribuição deste trabalho, e que abre várias oportunidades de trabalhos futuros.

10.1 Contribuições

O Adaptalker e a revisão literária realizada contribuem tanto para a área de pesquisa de gerenciamento de diálogo quanto para a área de pesquisas em adaptatividade. As contribuições deste trabalho de mestrado estão enumeradas abaixo.

10.1.1 Aperfeiçoamento dos gerenciadores baseados em máquinas de estado

O uso de adaptatividade permitiu incorporar novas características em um método comercialmente já bastante utilizado, o gerenciamento de diálogo baseado em máquinas de estado. As características viabilizadas pela adaptatividade foram a iniciativa mista, a sensibilidade ao contexto e o reparo pelo usuário em qualquer turno, sem necessidade de retrocesso ou de replicação das regras de reparo em vários estados. Além disso, outras características importantes, embora não devidas a adaptatividade, foram incorporadas, como a capacidade de manipular múltiplas hipóteses paralelas sobre as ações do usuário e a divisão das regras em diversas submáquinas. A facilidade de desenvolvimento e de testes característica do método não adaptativo original foi herdada, embora a utilização de ações adaptativas aumente um pouco sua complexidade, como demonstrado para que a iniciativa mista e o reparo se tornem viáveis. Entretanto, essa complexidade de desenvolvimento pode ser diminuída através do uso de técnicas padronizadas de construção do dispositivo adaptativo, tal como a descrita no Apêndice B e utilizada para criar a submáquina de cadastro de usuário. Essas técnicas podem viabilizar, em trabalhos futuros, a

automação da construção de tais dispositivos a partir de uma descrição estruturada das informações e das tarefas que compõe o sistema de diálogo.

10.1.2 Contribuições ao projeto da Minerva

O Adaptalker pode ser utilizado na Minerva. Todas as condições necessárias são atendidas pela proposta. Para ser utilizado nela de fato, entretanto, é necessário construir os demais módulos da arquitetura, como por exemplo os responsáveis pela análise semântica e geração de texto em linguagem natural, e integrá-los ao Adaptalker através da definição das funções comunicativas específicas requeridas pela Minerva, tanto de entrada quanto de saída, tal como seria necessário para qualquer outra aplicação desenvolvida com o método proposto. Os indicadores de emoções do usuário, embora não utilizados no gerenciador ilustrativo de venda de pizzas, podem ficar disponibilizados no próprio quadro negro e serem indicados através de variáveis de contexto ou a proposta atual poderia ser estendida para incluir nos atos dialogais de entrada esses indicadores, de forma similar aos qualificadores de sentimentos descritos na norma ISO 24617-2 (BUNT et al., 2012).

10.1.3 Dispositivo adaptativo organizado em submáquinas

Do ponto de vista da adaptatividade, o modelo de submáquinas utilizado é uma variação dos dispositivos descritos em outros trabalhos, tais como aquele das máquinas de Markov (BASSETO, 2000) e dos autômatos adaptativos (SHIBATA, 2008). As ações adaptativas que permitem que uma submáquina consulte ou altere transições de outra continuam presentes nesta proposta. Entretanto, o DATD permite que cada submáquina se comporte como um agente que pode se candidatar a resolver um problema, representado pela cadeia de entrada contendo os atos dialogais do usuário. Uma submáquina pode se candidatar a resolver apenas parte dos atos dialogais presentes, de tal forma que várias submáquinas possam colaborar entre si para processar ações do usuário com combinações mais complexas de atos dialogais. Para os casos em que mais de uma submáquina se candidata a resolver o problema, um método simples de comparação das soluções é utilizado. Para evitar que muitas máquinas fiquem ativas simultaneamente, aumentado o espaço de busca de regras, propõe-se também uma técnica para desativar as submáquinas.

Esta forma de utilização das submáquinas poderia ser aproveitada em outros dispositivos, não se limitando a aplicações no gerenciamento de diálogo falado ou ao particular dispositivo subjacente estudado. Ela pode ser utilizada sempre que o espaço de busca de regras for muito grande ou se este espaço crescer muito rapidamente, e se as regras puderem ser agrupadas em partições, cada uma com grande coesão semântica entre suas regras. No gerenciamento de diálogo, esses problemas já tinham sido notados em outros trabalhos, levando ao aperfeiçoamento de modelos já existentes, como é o caso do modelo do estado da informação oculta para gerenciadores baseados em POMDP (YOUNG et al., 2010).

10.1.4 Classificação formal dos cenários do diálogo do ponto de vista da problemática do gerenciamento

O trabalho de refinamento dos critérios de avaliação dos gerenciadores de diálogo existentes na literatura resultou na classificação formal de uma série de situações do diálogo que devem ser suportadas por um GD. Não foi encontrado trabalho equivalente na literatura, sendo, desta forma, uma contribuição importante para a área de sistemas de diálogos falado. Durante o projeto de um gerenciador específico, podem-se utilizar os vinte e um cenários listados, na forma proposta ou estendidas, para verificar se ele atende a todas as condições necessárias para a utilização no dia-a-dia com o usuário final. É importante relembrar que o Adaptalker por si só não garante a satisfação de todas as condições, mas permite que o desenvolvedor, ao utilizá-lo como a plataforma de desenvolvimento de seu GD, decida quais situações devem ser suportadas e modele as regras de forma a tratá-las adequadamente.

10.2 Trabalhos futuros

Existem vários trabalhos futuros que podem ser derivados da proposta descrita neste texto.

10.2.1 Desenvolvimento e testes do sistema completo

Um trabalho futuro importante é a utilização do Adaptalker em um sistema de diálogo falado completo, desde o reconhecimento até a síntese de voz. Desta forma será

possível avaliar melhor as contribuições feitas. A aplicação de testes desenvolvida é pequena, com poucas transições, embora satisfaça a quase totalidade dos cenários listados no capítulo 6, exceto pela situação 21 e pelos reparos feitos pelo sistema (situações 14, 16, 17 e 19). Apenas seis submáquinas foram criadas, nenhuma delas com mais de cinquenta regras. Uma aplicação comercial, no entanto, poderá requerer muito mais submáquinas, algumas delas possivelmente com muito mais regras, o que poderá exigir possíveis aperfeiçoamentos no modelo visando o desempenho. Estas novas regras podem levar a combinações dos cenários do capítulo 6 em que o Adaptalker ainda não foi testado.

10.2.2 Refinamento dos critérios e adaptação do Adaptalker para a multimodalidade

A multimodalidade não foi utilizada no modelo final do Adaptalker, por se acreditar que não haveria tempo suficiente para tratar o assunto de forma adequada, sem prejudicar as demais propostas da pesquisa. Além disso, existem poucos trabalhos sobre gerenciadores de diálogos multimodais, se comparados aos trabalhos sobre gerenciadores não multimodais, e pouca padronização, ferramentas e técnicas para classificação de gestos em sistemas computacionais. Acredita-se, no entanto, que a proposta de interfaces para os atos dialogais e funções comunicativas feitas no capítulo 5 são adequadas para fazer o gerenciamento multimodal. Trabalhos futuros podem aperfeiçoar o modelo proposto para gestos e tirar proveito das técnicas propostas e da adaptatividade para viabilizar a multimodalidade no Adaptalker. Tal trabalho deveria iniciar com a expansão da lista de cenários de diálogo que devem ser suportados por um GD, pois os gestos adicionam variáveis que não podem ser contempladas pela lista atual, como eventos relativos aos gestos sendo recebidos em paralelo aos eventos de voz.

Ressalta-se que um modelo multimodal se beneficia muito de um modelo incremental, que permite tratar de forma mais adequada eventos paralelos, pois as unidades de processamentos são menores que o tamanho de um turno, o que permite rever as decisões tomadas em intervalores menores do que em um modelo não incremental.

10.2.3 Extensão do Adaptalker para situações referentes a diálogos envolvendo várias pessoas

Existem alguns tipos de aplicações propícios a situações que envolvam mais de dois indivíduos, tais como robôs guias de museu. Existem problemas adicionais que devem ser considerados neste tipo de conversação, como a iniciativa e controle do turno (por exemplo, nem sempre é o sistema quem deve responder a uma pergunta) e o cisma do diálogo, isto é, a divisão da conversa em duas conversas, em uma das quais possivelmente não se deseja que o sistema participe.

10.2.4 Adaptação do Adaptalker para o processamento incremental

A utilização de um modelo incremental de processamento pode não ser viável para o Adaptalker – um novo modelo completamente novo pode ser necessário. Isto não quer dizer, no entanto, que a adaptatividade não seja uma boa solução para modelos incrementais. A dificuldade maior em utilizar o Adaptalker é devida ao modelo baseado em máquina de estados cujas unidades básicas de construção de diálogo são conjuntos de atos dialogais completos e finais, isto é, hipóteses sobre a fala do usuário contendo um turno completo e que não podem mais ser revogadas.
O processamento incremental pode ser utilizado para tornar o comportamento do sistema mais natural conforme a percepção do usuário, principalmente quanto ao *feedback* e ao reparo em um sistema multimodal. O sistema pode, por exemplo, dar *feedbacks* positivos ao usuário antes do turno deste último terminar utilizando outro meio de comunicação que não a voz, ou antecipar a detecção de *feedbacks* negativos do usuário quanto à ação do sistema pela análise de expressões faciais e movimentos de cabeça e de pescoço.

10.2.5 Incorporação de adaptatividade a outras técnicas de gerenciamento

A adaptatividade pode ser incorporada a outras técnicas de gerenciamento de diálogo, como, por exemplo, nos modelos baseados em planejamento e agenda de execução e nos modelos baseados em MDP e POMDP. Muitos destes modelos não contemplam sensibilidade ao contexto como parte de seu modelo formal, resolvendo esta

necessidade através da utilização de procedimentos computacionais desenvolvidos via código-fonte, como é o caso do Ravenclaw.

10.2.6 Equivalência formal do DATD com os autômatos adaptativos

Seria importante tentar demonstrar, se possível, a equivalência formal do dispositivo adaptativo de tomada de decisão proposto a um autômato adaptativo, algo que ficou ausente neste trabalho. Isto demonstraria que o formalismo proposto é Turing-completo, pois o autômato adaptativo também é (ROCHA; NETO, 2000). Como há vários detalhes da execução do dispositivo específicos do gerenciamento do diálogo, poderá ser necessário fazer a abstração de alguns passos de execução e mostrar a equivalência entre a abstração e o modelo concreto. Se a equivalência não for completa, pode-se tentar demonstrar que o DATD resolve um subconjunto dos problemas resolvidos por um autômato adaptativo.

10.3 Considerações finais

Este trabalho apresenta uma evolução no estudo do gerenciadores de diálogo ao descrever uma proposta adaptativa de gerenciador, especialmente se considerado que o dispositivo subjacente, baseado em máquinas de estados, é popular comercialmente mas é considerado um modelo com deficiências importantes quando comparado a outras técnicas (MCTEAR, 2002). A adaptatividade é utilizada para remover estas deficiências, revitalizando o modelo. Os resultados obtidos demonstram que o modelo proposto, denominado de Adaptalker, proporciona o desenvolvimento de gerenciadores de diálogo que suportem várias situações complexas de diálogo, incluindo muitas que não estão evidentes em qualquer um dos trabalhos revisados, embora possam acontecer normalmente no uso diário de sistemas de diálogo.

O dispositivo adaptativo para tomada de decisão proposto faz uso de várias características já presente em outros dispositivos adaptativos, como estados e transições entre estados (utilizados em autômatos adaptativos), cadeia de saída e funções adaptativas com chamada de ações elementares em uma sequência definida (utilizada em sistemas de Markov adaptativos), regras com condições de execução (tabelas de decisão adaptativas) e organização em submáquinas (sistemas de Markov

adaptativos e autômatos adaptativos). Essas características são utilizadas em conjunto com outras introduzidas neste trabalho, como:

- A ativação e desativação de submáquinas de forma dinâmica;
- Controle do foco de execução das submáquinas;
- Concorrência entre as submáquinas para resolver o problema, representado pela cadeia de entrada, de forma total ou parcial;
- Especialização dos estados (estados de *binding* e estados convencionais) e das saídas das transições (quatro tipos de tarefas) para resolver problemas distintos. No caso do gerenciamento do diálogo, os estados de *binding* são utilizados para relacionar as ações do usuário a ações prévias do sistema; os estados convencionais são utilizados para decidir a próxima ação do sistema a partir da última ação do usuário.

Pode-se concluir que essas características do dispositivo adaptativo, associadas às técnicas de gerenciamento de diálogo, incluindo a utilização de normais internacionais e padrões obtidos da literatura, tornam o framework proposto competitivo, inclusive superando outras técnicas de gerenciamento de diálogo em alguns dos critérios considerados – ao menos quanto aos critérios que puderam ser comparados.

REFERÊNCIAS

ALFENAS, D. A. et al. **Sistemas de Markov Adaptativos: Formulação e Plataforma de Desenvolvimento.** Memórias do WTA 2012 Sexto Workshop de Tecnologia Adaptativa. São Paulo, Brasil: 2012.

ALFENAS, D. A.; PEREIRA-BARRETTO, M. R. **Adaptatividade em Robôs Sociáveis: uma Proposta de um Gerenciador de Diálogos.** Memórias do WTA 2012 Sexto Workshop de Tecnologia Adaptativa. São Paulo, Brasil: 2012.

ALFENAS, D. A.; PEREIRA-BARRETTO, M. R.; PAIXÃO, R. DOS S. **Sobre o uso de formalismos adaptativos no gerenciamento de diálogos em robôs sociáveis.** Memórias do WTA 2013 Sétimo Workshop de Tecnologia Adaptativa. São Paulo, Brasil: 2013.

AUBANEL, V.; NGUYEN, N. Automatic recognition of regional phonological variation in conversational interaction. **Speech Communication**, v. 52, n. 6, p. 577–586, 2010.

BAKER, M. et al. The role of grounding in collaborative learning tasks. In: **Collaborative learning: Cognitive and computational approaches**. [s.l.] Elsevier Science, 1999. p. 31–63.

BASSETO, B. A. **Um sistema de composição musical automatizada, baseado em gramáticas sensíveis ao contexto, implementado com formalismos adaptativos**. Tese (Doutorado) - Escola Politécnica, Universidade de São Paulo, São Paulo, Brasil, 2000.

BOCK, C. et al. **OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (Second Edition)**. Disponível em: <http://www.w3.org/TR/owl2-syntax/>. Acesso em: 20 jul. 2014.

BOHUS, D.; RUDNICKY, A. I. The RavenClaw dialog management framework: Architecture and systems. **Computer Speech & Language**, v. 23, n. 3, p. 332–361, jul. 2009.

BRAVO, C. A. **Gramáticas Livres de Contexto Adaptativas com Verificação de Aparência**. Tese (Doutorado) - Escola Politécnica, Universidade de São Paulo, São Paulo, Brasil, 2004.

BUI, T. H. **Multimodal Dialogue Management - State of the artCTIT Technical Report series**. Enschede, Holanda: Centre for Telematics and Information

Technology, University of Twente, 2006. Disponível em: <http://www.narcis.nl/publication/RecordID/oai:doc.utwente.nl:63050>. Acesso em: 29 jul. 2013.

BUNT, H. **Dimensions in Dialogue Act Annotation.** Proceedings of the Fifth International Conference on Language Resources and Evaluation. Genova, Itália: 2006.

BUNT, H. **The DIT++ taxonomy for functional dialogue markup** (D. Heylen et al., Eds.) Proceedings of the AAMAS 2009 Workshop “Towards a Standard Markup Language for Embodied Dialogue Acts” (EDAML 2009). Budapeste, Hungria: 2009.

BUNT, H. et al. **ISO 24617-2 : A semantically-based standard for dialogue annotation.** Proceedings of LREC 2012. Istambul, Turquia: 2012.

CALBRIS, G. **Elements of meaning in gesture**. [s.l.] John Benjamins Publishing Company, 2011.

CASTRO JÚNIOR, A. A; JOSÉ NETO, J.; PISTORI, H. Determinismo em Autômatos de Estados Finitos Adaptativos. **Revista IEEE América Latina**, v.5, n.7.

CHEONGJAE LEE et al. Recent Approaches to Dialog Management for Spoken Dialog Systems. **Journal of Computing Science and Engineering**, v. 4, n. 1, p. 1–22, 2010.

CLARK, H. H.; BRENNAN, S. E. Grounding in communication. In: RESNICK, L. B.; LEVINE, J. M.; TEASLEY, S. D. (Eds.). **Perspectives on socially shared cognition**. Washington DC: American Psychological Association, 1991. p. 127–149.

CUEVA, D. R. **Fusão Computacional de Observações Afetivas**. Dissertação (Mestrado) - Escola Politécnica, Universidade de São Paulo, São Paulo, Brasil, 2012.

DENTLER, K. et al. Comparison of reasoners for large ontologies in the OWL 2 EL profile. **Semantic Web**, v. 2, p. 71–87, 2011.

DEUGO, D.; WEISS, M.; KENDALL, E. Reusable patterns for agent coordination. In: OMICINI, A. et al. (Eds.). **Coordination of Internet agents**. Berlim: Springer, 2001. p. 347–368.

DÍAZ, M. et al. **Building up child-robot relationship from initial attraction towards social engagement** (V. Hafner et al., Eds.) HRI 2011 Workshop on

Expectations in intuitive human-robot interaction. Lausana, Suíça: 2011Disponível em: <http://introbotics.eu/HRI2011/>.

DUMAS, B.; LALANNE, D.; OVIATT, S. Multimodal Interfaces : A Survey of Principles , Models and Frameworks. In: LALANNE, D.; KOHLAS, J. (Eds.). **Human Machine Interaction**. [s.l.] Springer Berlin Heidelberg, 2009. p. 3–26.

EDLUND, J. et al. Towards human-like spoken dialogue systems. **Speech Communication**, v. 50, n. 8, p. 630–645, 2008.

GAMA, E. et al. **Design patterns: elements of reusable object-oriented software**. [s.l.] Addison-Wesley, 1994.

GARCEZ, P. M.; LODER, L. L. Reparo iniciado e levado a cabo pelo outro na conversa cotidiana em português do Brasil. **Delta**, v. 21, n. 2, p. 279–312, 2005.

GARFINKEL, H. **Studies in Ethnomethodology**. [s.l.] Englewood Cliffs, NJ: Prentice-Hall, 1967.

GONÇALVES, R. **Um Modelo Matemático para Inferência Computacional de Estado Emocional a Partir de Detectores de Expressões Faciais**. Dissertação (Mestrado) - Escola Politécnica, Universidade de São Paulo, São Paulo, Brasil, 2013.

GOODWIN, C. Action and embodiment within situated human interaction. **Journal of Pragmatics**, v. 32, n. 10, p. 1489–1522, set. 2000.

HIRAKAWA, A. R.; SARAIVA, A. M.; CUGNASCA, C. E. **Autômatos Adaptativos Aplicados em Automação e Robótica (A4R).** Memórias do WTA 2012 Sexto Workshop d Tecnologia Adaptativa. São Paulo, Brasil: 2012

HORRIDGE, M.; BECHHOFER, S. The owl api: A java api for owl ontologies. **Semantic Web**, v. 2, p. 11–21, 2011.

HOWARD, R. A. **Dynamic Probabilistic Systems**. 1st. ed. Nova York, EUA: John Wiley & Sons, 1971.

JOSÉ NETO, J. **Introduçao à compilaçao**. Rio de Janeiro, Brasil: LTC, 1987. p. 222.

JOSÉ NETO, J. **Contribuiçoes à metodologia de construçao de compiladores**. Tese (Livre Docência) - Escola Politécnica, Universidade de São Paulo, São Paulo, Brasil, 1993.

JOSÉ NETO, J. Adaptive Automata for Context-Sensitive Languages. **SIGPLAN NOTICES**, v. 29, n. 9, p. 115–124, 1994.

JOSÉ NETO, J. **Adaptive rule-driven devices - general formulation and case study** (B. W. Watson, D. Wood, Eds.) Implementation and Application of Automata 6th International Conference, CIAA. Pretória, África do Sul: Springer-Verlag, 2001.

JOSÉ NETO, J. Um Levantamento da Evolução da Adaptatividade e da Tecnologia Adaptativa. **IEEE Latin America Transactions**, v. 5, n. 7, 2007.

JOSÉ NETO, J.; ALMEIDA JUNIOR, J. R. DE; SANTOS, J. M. M. DOS. **Synchronized statecharts for reactive systems.** Proceedings of the IASTED International Conference on Applied Modelling and Simulation. Honolulu, Havaí: 1998.

KANG, S.; KO, Y.; SEO, J. Generating Effective Responses for a Plan-based Dialogue System in Human-robot Interaction. **INFORMATION-AN INTERNATIONAL INTERDISCIPLINARY JOURNAL**, v. 15, n. 2, p. 949–960, 2012.

KERBRAT-ORECCHIONI, C. **A MULTILEVEL APPROACH IN THE STUDY OF TALK-IN-INTERACTION.** In: 5th Internation Pragmatics Conference. ***Proceedgins***...Cidade do México, México: 1996.

KERBRAT-ORECCHIONI, C. **Análise da Conversação. Princípios e Métodos.** São Paulo, Brasil: Parábola Editorial, 2006.

KNAPP, M. L.; JUDITH A. **Comunicação não-verbal na interação humana**. [s.l.] JSN Editora, 2006.

KOPP, S. Social Resonance and Embodied Coordination in Face-to-Face Conversation with Artificial Interlocutors. **Speech Communication**, v. 52, n. 6, p. 587–597, 2010.

KRÖTZSCH, M.; SIMANCÍK, F.; HORROCKS, I. **A Description Logic Primer**. [s.l.], 2013.

LALANNE, D. et al. **Fusion Engines for Multimodal Input : A Survey** ICMI-MLMI'09 Proceedings. Cambridge, EUA: ACM, 2009.

LEITE, T. **A segmentação da língua de sinais brasileira (libras): Um estudo lingüístico descritivo a partir da conversação espontânea entre surdos**. Tese (Doutorado em Estudos Linguísticos e Literários em Inglês) – Faculdade de Filosofia, Letras e Ciências Humanas, Universidade de São Paulo, São Paulo 2008.

LEMON, O. Learning what to say and how to say it: Joint optimisation of spoken dialogue management and natural language generation. **Computer Speech & Language**, v. 25, n. 2, p. 210–221, abr. 2011.

LIBERMAN, K. Semantic Drift in Conversations. **Human Studies**, v. 35, n. 2, p. 263–277, 2012.

MADEO, R. C. B. **Máquinas de Vetores Suporte e a Análise de Gestos: incorporando aspectos temporais**. Dissertação (Mestrado em Sistemas de Informação) - Escola de Artes, Ciências e Humanidades, Universidade de São Paulo, São Paulo, 2013.

MCCLEARY, L.; LEITE, T. D. A. Turn-taking in Brazilian Sign Language : Evidence from overlap. **Journal of Interactional Research in Communication Disorders**, 2012.

MCTEAR, M. F. Spoken Dialogue Technology : Enabling the Conversational User Interface. **ACM Computing Surveys**, v. 34, n. 1, p. 90–169, 2002.

MILLER, G. A. WordNet : A Lexical Database for English. **Communications of the ACM**, v. 38, n. 11, p. 39–41, 1995.

PETUKHOVA, V. V. **Multidimensional Dialogue Modelling**. Tese (Ph.D.) -Tilburg University, 2011.

PISTORI, H.; JOSÉ NETO, J.; PEREIRA, M. . Adaptive Non-Deterministic Decision Trees : General Formulation and Case Study. **INFOCOMP Journal of Computer Science**, 2006.

ROCHA, R. DA; NETO, J. **Autômato adaptativo, limites e complexidade em comparaçao com a Máquinha de Turing.** Proceedings of the second Congress of Logic Applied to Technology - LAPTEC 2000. São Paulo: Faculdade SENAC de Ciências Exatas e Tecnologia: 2000. Disponível em: <http://www.pcs.usp.br/~lta/artigos/rocha_laptec2000.pdf>. Acesso em: 25 ago. 2014

ROMERO, R. A. F.; SANTOS, V. DE C. Imitação de expressões faciais para aprendizado de emoções em robótica social. 2012.

SACKS, H.; SCHEGLOFF, E. A.; JEFFERSON, G. A Simplest Systematics for the Organization of Turn-Taking for Conversation. **Language**, v. 50, n. 4, p. 696–735, 1974.

SAINI, P.; KAUR, P. Automatic Speech Recognition-A Review. **International journal of Engineering Trends & …**, v. 4, n. iii, p. 132–136, 2013.

SCHEGLOFF, E. A. On some gestures ' relation to talk. In: ATKINSON, J. M.; HERITAGE, J. (Eds.). **Structures of Social Action: Studies in Conversation Analysis**. Cambridge: Cambridge University Press, 1984. p. 266–296.

SCHEGLOFF, E. A. Body Torque *. **Social Research**, v. 65, n. 3, p. 535–596, 1991.

SCHEGLOFF, E. A. Repair After Next Turn: The Last Structurally Provided Defense of Intersubjectivity in Conversation. **American Journal of Sociology**, v. 97, n. 5, p. 1295–1345, 1992.

SCHEGLOFF, E. A.; JEFFERSON, G.; SACKS, H. The preference for self-correction in the organization of repair in conversation. **Language**, v. 53, n. 2, p. 1977, 1977.

SCHEGLOFF, E. A.; SACKS, H. Opening up Closings *. **Semiotica**, v. 8, p. 289–327, ago. 1973.

SCHLANGEN, D.; SKANTZE, G. **A General , Abstract Model of Incremental Dialogue Processing.** Proceedings of the 12th Conference of the European Chapter of the ACL. Atenas, Grécia: Association for Computational Linguistics, 2009.

SCHULLER, B. et al. Paralinguistics in speech and language—State-of-the-art and the challenge. **Computer Speech & Language**, v. 27, n. 1, p. 4–39, jan. 2013.

SHANI, G.; PINEAU, J.; KAPLOW, R. A survey of point-based POMDP solvers. **Autonomous Agents and Multi-Agent Systems**, v. 27, n. 1, p. 1–51, 8 jun. 2012.

SHEARER, R.; MOTIK, B.; HORROCKS, I. HermiT: A Highly-Efficient OWL Reasoner. **OWLED**, v. 432, 2008.

SHIBATA, D. P. **Traduçao grafema-fonema para a língua portuguesa baseada em autômatos adaptativos**. Dissertação (Mestrado em Sistemas Digitais) - Escola Politécnica, Universidade de São Paulo, São Paulo, 2008.

SKANTZE, G. GALATEA: A Discourse Modeller Supporting Concept-level Error Handling in Spoken Dialogue Systems. **Recent Trends in Discourse and Dialogue**, 2008.

SKANTZE, G.; HJALMARSSON, A. **Towards Incremental Speech Generation in Dialogue Systems.** Proceedings of SIGDIAL 2010: the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue. Tóquio, Japão: Association for Computational Linguistics, 2010.

SPEER, R.; HAVASI, C. **Representing General Relational Knowledge in ConceptNet 5.** Proceedings of Eighth International Conference on Language Resources and Evaluation, LREC, 2012.

STANGE, R. L.; NETO, J. J. **Aprendizagem Incremental usando Tabelas Decisão Adaptativas.** Memórias do WTA 2011 Quinto Workshop de Tecnologia Adaptativa. São Paulo: 2011

TCHEMRA, A. **Tabela de decisão adaptativa na tomada de decisão multicritério**. Tese (Doutorado em Sistemas Digitais) - Escola Politécnica, Universidade de São Paulo, São Paulo, 2009.

THOMSON, B. **Statistical methods for spoken dialogue management**. Tese (Ph.D.) - University of Cambridge, EUA: 2009.

THRUN, S. et al. **MINERVA : A Second-Generation Museum Tour-Guide Robot.** Proceedings of 1999 IEEE international conference on robotics and automation. IEEE, 1999.

TOMASELLO. Human Cooperative Comunication. In: **Origins of Human Communication**. Cambridge, EUA: MIT Press, 2008. p. 57–108.

TRAUM, D. et al. Incremental Dialogue Understanding and Feedback for Multiparty , Multimodal Conversation. In: **Intelligent Virtual Agents**. [s.l.] Springer Berlin Heidelberg, 2012. p. 275–288.

TRAUM, D. R.; LARSSON, S. The Information State Approach to Dialogue Management. In: KUPPEVELT, J. VAN; SMITH, R. W. (Eds.). **Current and New Directions in Discourse and Dialogue**. [s.l.] Springer Netherlands, 2003. p. 325–353.

TUDORACHE, T.; VENDETTI, J.; NOY, N. F. **Web-Protege: A Lightweight OWL Ontology Editor for the Web.** (C. Dolbear, A. Ruttenberg, U. Sattler, Eds.) Proceedings of the Fifth OWLED Workshop on OWL: Experiences and Directions.

CEUR-WS, 2008. Disponível em: <https://owl1-1.googlecode.com/svn-history/r642/trunk/www.webont.org/owled/2008/papers/owled2008eu_submission_40.pdf>. Acesso em: 20 jul. 2014.

UCHIDA, H.; ZHU, M. **The universal networking language beyond machine translation.** In: International Symposium on Language in Cyberspace. **Proceedings**...Seul, Coreia: 2001. Disponível em: <http://www.undl.org/publications/UNL-beyond MT.html>.

WALLACE, R. S. **The Elements of AIML Style**. [s.l.], 2003.

WILLIAMS, J. D.; AVE, P.; PARK, F. **Incremental Partition Recombination for Efficient Tracking of Multiple Dialog States.** In: Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on. **Proceedings**...IEEE, 2010.

WILLIAMS, J. D.; YOUNG, S. Partially observable Markov decision processes for spoken dialog systems. **Computer Speech & Language**, v. 21, n. 2, p. 393–422, abr. 2007.

WOOD, J. T. **Interpersonal Communication: Everyday Encounters**. 6th. ed. Boston: Wadsworth, Cengage Learning, 2010. p. 368.

YOUNG, B. S. et al. POMDP-Based Statistical Spoken Dialog Systems : A Review. **Proceedings of the IEEE**, v. 101, n. 5, 2013.

YOUNG, S. et al. The Hidden Information State model: A practical framework for POMDP-based spoken dialogue management. **Computer Speech & Language**, v. 24, n. 2, p. 150–174, abr. 2010.

YOUNG, S.; GASIC, M. Effective Handling of Dialogue State in the Hidden Information State POMDP-based Dialogue Manager. **ACM Trans. Speech Lang. Process**, v. 7, n. 3, 2011.

ZHENG, K. et al. **Supervisory control of multiple social robots for navigation**2013 8th ACM/IEEE International Conference on Human-Robot Interaction (HRI) Proceedings. IEEE, mar. 2013. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6483497> .

APÊNDICE A – Sobre a notação gráfica simplificada do DATD

Algumas figuras deste trabalho contêm uma visualização gráfica e simplificada do dispositivo adaptativo de tomada de decisão (DATD), que é parte do modelo proposto para o gerenciamento de diálogo Adaptalker. Este apêndice clarifica a notação utilizada.

Estados de *binding* – Representados por círculos, alguns mais ovais que outros.

Estados convencionais – Representados por retângulos com cantos arredondados.

Estado inicial – Ressaltado por uma borda mais grossa que a dos demais estados.

Estado de aceitação – Representado por losangos.

Transições – Representadas por linhas com setas ligando os estados. O estado de destino sempre fica junto à seta. Setas com linha contínua indicam transições que não serão, no exemplo mencionado, alteradas por ações adaptativas. Setas com linhas tracejadas podem ser alteradas ou removidas por funções adaptativas. A cor vermelha é utilizada para ressaltar alterações feitas de forma adaptativa na submáquina, se possível – remoções simplesmente desaparecem.

Condições – As condições para execução de transição estão entre colchetes (“[“ e “]”), anotadas junto à transição. A condição não é descrita de forma completa, mas apenas com um texto resumido que permita referenciá-la no texto, se necessário. A ausência de colchetes ou a utilização do símbolo “ε” indicam cadeia vazia, isto é, a execução da transição é incondicional.

Tarefa – As tarefas executadas estão anotadas junto à transição a que pertencem, e iniciam após uma barra “\”. A tarefa não é descrita de forma completa, mas apenas com um texto resumido que permita referenciá-la no texto, se necessário.

Ação adaptativa – As ações adaptativas estão anotadas junto à transição à qual pertencem, e iniciam após uma barra “\” da mesma maneira que a tarefa. O símbolo “α” seguido por um sinal de igual indica o nome da ação. Se existe um ponto (“.”) antes de “α”, então a ação é adaptativa posterior, Se o ponto fica depois do “α”, então a ação é adaptativa anterior. A ação adaptativa não é descrita de forma completa, mas apenas com um texto resumido que permita referenciá-la no texto, se necessário.

APÊNDICE B - Algoritmo para geração de regras para o DATD

O algoritmo abaixo, com passos enumerados de 1 a 8, pode ser utilizado para gerar regras para uma submáquina do DATD para solicitar um conjunto de informações semanticamente independentes entre si[10]. Tal algoritmo foi utilizado para criar a submáquina de cadastro, utilizada como exemplo ao longo do texto. Neste apêndice, utiliza-se um novo exemplo, uma submáquina para cadastro de visitantes, que segue o mesmo princípio.

1. Define-se a ordem padrão em que se deseja solicitar as informações requeridas pela tarefa. No nosso exemplo, define-se que se deve solicitar o nome, telefone e empresa do visitante, nesta ordem.

2. Criam-se dois estados para cada informação, o primeiro um estado convencional com nome começando com “Pergunta” e o segundo um estado de *binding* com nome começando com “Resposta”. Para a informação “nome”, por exemplo, teremos os estados “Pergunta nome” e “Resposta nome”. Para cada informação, cria-se uma transição ligando esses dois estados, sem condição e com tarefa de requisição ao usuário de função comunicativa “*wh-question*” e item semântico igual à informação solicitada. Essas transições devem ser colocadas em um vetor de transições v_s, com a transição solicitando a primeira informação em $v_s[1]$ e os demais colocados nas posições seguintes, conforme a ordem definida no passo 1.

3. Cria-se um inteiro de iteração i, com valor inicial igual a 1.

4. Cria-se um estado chamado “Marcador próxima pergunta”, que é também o estado inicial da máquina. Cria-se uma transição entre “Marcador próxima pergunta” e o estado de origem da transição $v_s[1]$.

5. Cria-se um estado chamado “Reparo”.

6. Para cada transição $v_s[i]$:

    6.1 Cria-se uma transição entre o estado de destino de $v_s[i]$ e “Marcador próxima pergunta”, tendo como condição de execução um ato dialogal de função comunicativa *inform* e item semântico igual à informação solicitada ou dois atos dialogais, um de função *inform* e o outro de função *answer*. Não há

[10] É possível criar um algoritmo equivalente, mas para informações que possuem alguma dependência semântica, como seria o caso de uma tarefa de oferecer bebida utilizando as seguintes informações: “*bebida”, “sabor do chá”*. A descrição de tal algoritmo será deixada para trabalhos futuros.

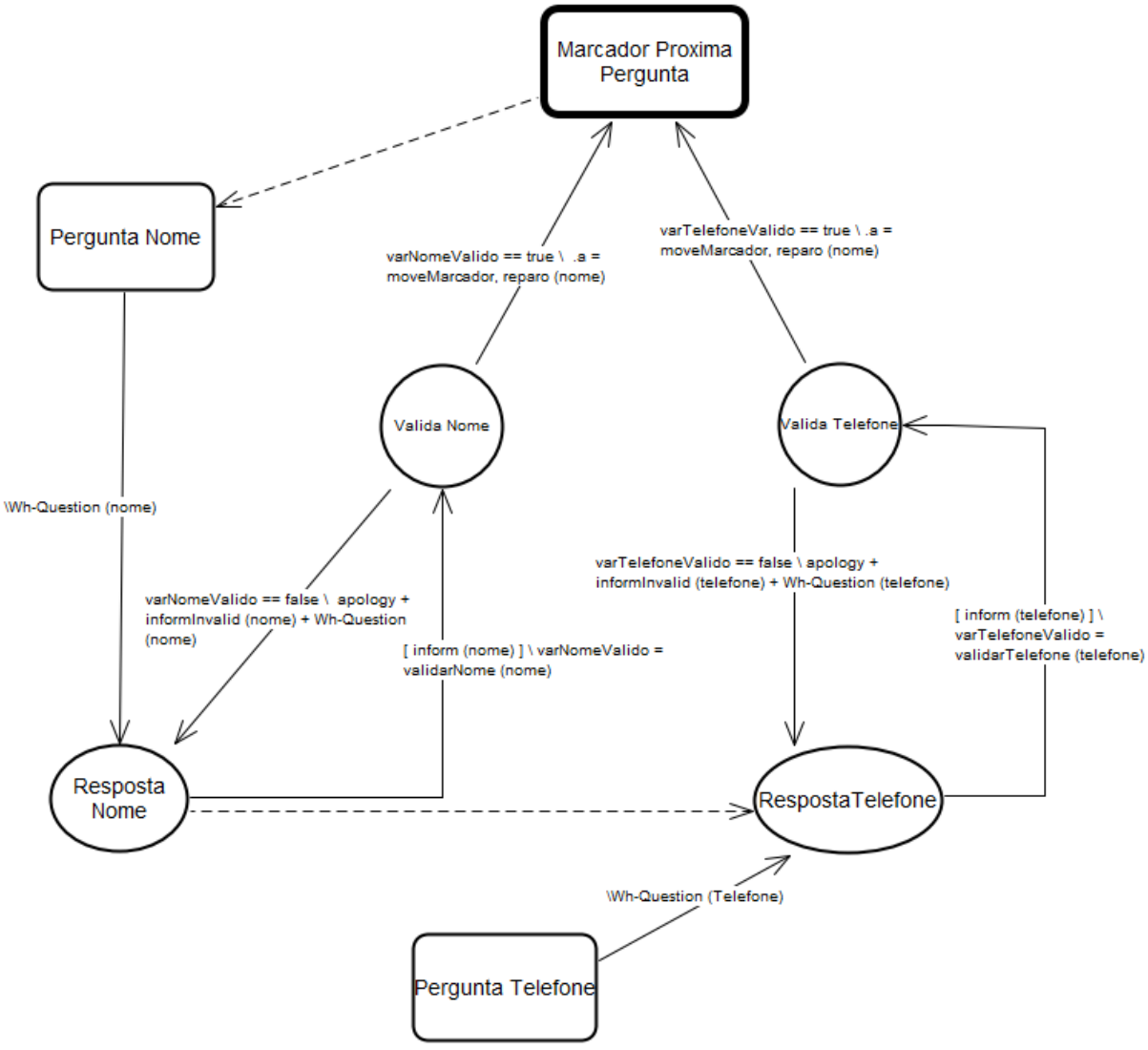
tarefa, mas há uma ação adaptativa posterior para chamada das funções “reparo” (passando como parâmetro o nome da informação) e “moveMarcador”, as duas como declaradas no capítulo 9 - Resultados.

6.2 Se $v_S[i+1]$ existe, cria-se uma transição entre o estado destino de $v_S[i]$ e o estado destino de $v_S[i+1]$ que possui como condição a cadeia vazia.

6.3 Incrementa-se *i*.

6. Cria-se uma transição ligando o estado de destino de $v_S[i-1]$ (é a última transição em v_S) e o estado “Reparo”, tendo como condição a cadeia vazia.

Figura 37 – Parte da submáquina de cadastro, agora com regras com tarefas de sistema com instruções de validação das informações fornecidas. O Apêndice A explica a notação utilizada.



7. Criam-se as transições para execução da tarefa de sistema que utiliza as informações obtidas. O primeiro estado da primeira transição deve-se chamar “Pronto para Registrar”. A última transição deve ser uma que dá *feedback* sobre a execução com sucesso da tarefa e cujo estado destino é um estado de *binding* de aceitação, que indica que a submáquina foi executada com sucesso.
8. Cria-se uma transição ligando o estado de aceitação e o estado “Reparo”, tendo como condição a cadeia vazia. Esta transição existe para que seja possível fazer reparos em informações mesmo depois da desativação da submáquina.

O desenvolvedor pode alterar a máquina de estados gerada, por exemplo, para validar cada informação fornecida. Todas estas transições de validação devem ser inseridas entre a transição de *binding* e o estado “Marcador próxima pergunta”, antes da chamada da função adaptativa “moveMarcador”. A Figura 37 exibe uma parte da submáquina de cadastro, agora expandida com as regras responsáveis pela validação das informações recebidas do usuário.