Emerson Luís dos Santos

Um Estudo sobre Memória Dinâmica em um Ambiente de Software

Dissertação apresentada como créditos parciais do curso de Mestrado ao Programa de Pós-Graduação em Informática Aplicada, Centro de Ciências Exatas e de Tecnologia, da Pontifícia Universidade Católica do Paraná.

Curitiba
Ago 2005

Emerson Luís dos Santos

Um Estudo sobre Memória Dinâmica em um Ambiente de Software

Dissertação apresentada como créditos parciais do curso de Mestrado ao Programa de Pós-Graduação em Informática Aplicada, Centro de Ciências Exatas e de Tecnologia, da Pontifícia Universidade Católica do Paraná.
Área de Concentração: Agentes de Software

Orientador: Prof. Dr. Bráulio Coelho Ávila
Co-orientador: Prof. Dr. Edson Emílio Scalabrin

Curitiba
Ago 2005

Santos, Emerson Luís dos
S237e Um estudo sobre memória dinâmica em um ambiente de software / Emerson
2005 Luís dos Santos; orientador, Bráulio Coelho Ávila; co-orientador, Edson Emílio Scalabrin. — 2005.
xiv, 118f.: il.; 30cm

Dissertação (mestrado) — Pontifícia Universidade Católica do Paraná, Curitiba, 2005
Inclui bibliografia

1. Ciência cognitiva. 2. Inteligência artificial. 3. Software. I. Ávila, Bráulio Coelho. II. Scalabrin, Edson Emílio. III. Pontifícia Universidade Católica do Paraná. Programa de Pós-Graduação em Informática Aplicada. IV. Título.

CDD 20. ed. 006.31
005.3

Every hour of every day I'm learning more[a]
The more I learn, the less I know about before[b]
The less I know, the more I want to look around[c]
Digging deep for clues on higher ground[d] [e]

[a] A cada hora de cada dia eu estou aprendendo mais

[b] Quanto mais eu aprendo, menos eu sei sobre antes

[c] Quanto menos eu sei, mais eu quero olhar à volta

[d] Procurando por pistas sobre um lugar de sabedoria e bons tratamentos

[e] O extrato é o refrão da canção *Higher Ground*, cujas letra, música e interpretação pertencem à banda britânica *UB40*. Foi lançada em 1993, integrando o álbum *Promises and Lies*, da mesma banda.

À minha avó Ernesta, in memoriam, pelos 7 mais belos anos da minha vida.

Agradecimentos

Agradeço primeiramente a Deus porque desfruto de uma vida em que todas as grandes mudanças acontecem no momento mais adequado e, por mais que nesse momento eu não compreenda isso plenamente, acabo sempre reconhecendo com o passar do tempo o quanto cada experiência foi importante.

Em segundo lugar, embora uma singela menção não possa quantificar a gratidão devida, estarão sempre meu pai João Adair, minha mãe Vanda, minha irmã Ana Paula e meu irmão Robson, pelo sacrifício a mim dedicado e pelas conquistas que juntos obtivemos em meio a situações adversas e ambientes desfavoráveis. Tenham a certeza de que é pensando em nós que eu encaro as dificuldades que escolhi, convivo com as limitações a que me submeto e sou feliz mesmo sem algumas tentações do mundo das quais decidi abdicar e nunca me arrependi.

Sem dúvida alguma, muitos dos caminhos que segui e que me possibilitaram alcançar a conclusão desta dissertação são devidos às orientações que a mim o Prof. Bráulio dirigiu, aos auxílios dos quais precisei e que ele sempre se antecedeu a me conceder e à confiança em mim depositada. Certamente, uma das pessoas que mais influenciaram minha vida, talvez a mais educada que já conheci, e que, à sua maneira, deseja muito o meu êxito profissional e pessoal. Este trabalho não teria nem se aproximado de sua conclusão sem os recursos por você empregados em nosso empreendimento.

Ao Prof. Scalabrin, eu exponho mais uma vez a minha profunda admiração pelos bons olhos com que sempre viu os trabalhos realizados pelo grupo formado por Márcio, Mitsuo e eu, pelo conhecimento a nós transmitido e pelos valiosos conselhos que dele ganhei que abrangiram desde o desenvolvimento deste trabalho até os valores culturais e as expectativas às quais uma boa pessoa deve atender. Juntamente com o Prof. Shmeil, não é exagero dizer que ele nos conduziu a no mínimo próximo do ápice a que o ensino talvez possa chegar, ultrapassando o que se costuma denominar educação.

Agradeço também ao Prof. Shmeil pelas encantadoras sessões de transmissão de conhecimento, porque seria pejorativo a elas referir-se como aulas ou espaços de esclarecimento de dúvidas, tão nobres eram os momentos em que nos sentía-mos na presença de um catedrático proclamado. Sou muito feliz por ter tido a oportunidade de presenciar tudo isso.

Embora ele não tenha ministrado nenhuma cadeira, todas as vezes em que interpelei o Prof. Orlando para o esclarecimento de dúvidas pude apreciar o imenso conhecimento por ele detido e que ele demonstrou sempre disposto a compartilhar. Suas idéias tiveram contribuição particular para minhas reflexões ao longo deste trabalho.

As dificuldades superadas em conjunto por Mitsuo, Márcio e eu serão por mim sempre lembradas. A paciência que tiveram para conviver comigo, o respeito que demonstraram por minhas opiniões e o esforço que sempre fizeram para me ajudar quando precisei nunca poderão por mim ser igualmente retribuídos.

À Fernanda, por todos os momentos em que crescemos juntos na compreensão dos assuntos dos nossos trabalhos e pelo caprichoso auxílio prático na confecção da apresentação do meu PDM.

A Carolina e Vera, pelo respeito que sempre demonstraram por minhas opiniões e pela amizade que tivemos ao longo do mestrado.

O tratamento a mim dispensado por meu padrinho Sebastião e minha madrinha Leoni só são comparáveis ao recebido por um filho. Serei eternamente grato a vocês pela acolhida tão fraterna em sua casa e nunca esquecerei os almoços que com tanto carinho vocês prepararam.

Em grande parte desta jornada, eu pude conviver com meus primos Roberto, Rosângela e Emely, que sempre tiveram paciência comigo e tanto me ajudaram. Vocês adicionaram uma parcela de contribuição muito especial nesse tempo em que me dediquei a este trabalho.

Definitivamente, muitos dos bons momentos que tive ao longo desse tempo foram ao lado do meu primo Rodrigo e dos meus amigos Michael e Rafael. Muito obrigado por todo o esforço que sempre fizeram para que eu pudesse estar ao lado de vocês nesses momentos e pela amizade plena e sólida da qual me permitiram fazer parte.

A este ilustre companheiro de nome Genival, com o qual tive o prazer de construir uma amizade importantíssima. Você foi um dos que mais me ajudou a crescer como pessoa, transmitindo a mim conhecimentos que têm me ajudado a adquirir uma visão sistêmica adequada. O trabalho que juntos realizamos é uma das fontes de recursos que permitiram a mim concluir este trabalho.

Agradeço ainda ao Prof. Fabrício pelas importantes correções, seus valiosos comentários e ricas sugestões de melhoria a este documento em versões anteriores.

Um agradecimento particular à PUCPR pela oportunidade educacional e pelo suporte financeiro a mim concedidos.

Sou muito grato ainda a todos aqueles que contribuíram para que este trabalho tenha sido realizado e àqueles que me concederam qualquer tipo de auxílio de cunho pessoal ao longo do tempo em que permaneci envolvido neste projeto, mas que infelizmente, por razões de espaço, não pude privilegiar com uma menção explícita; nem por isso suas contribuições são vistas por mim com apreço inferior.

Resumo

O domínio de aplicação deste trabalho é um ambiente de *software* primitivo, consistindo de agentes virtuais que executam um conjunto mínimo de ações reflexivas e sem deslocamento espacial, no intuito de satisfazer objetivos instanciados através de mensagens estruturadas segundo uma linguagem restrita. Nesse domínio, mediante observações representadas em um formalismo elaborado e coletadas em sincronia com os ciclos de uma simulação, dotar um agente observador externo ao ambiente virtual de uma memória dinâmica para internalizar as alterações no estado do mundo, visando dispor de previsões que evidenciem algum grau de compreensão das relações entre os eventos reconhecidos, é o propósito deste estudo. Ao longo da história da Inteligência Artificial, teorias sobre estruturas de memória que alguns autores acreditam integrarem os fundamentos do raciocínio humano restringiram-se a aplicações onde o conhecimento é estático e em domínios orientados à compreensão de linguagem natural. Sistemas baseados em agentes de *software* amplamente conhecidos na literatura têm utilizado métodos cujas fundações apóiam-se em abordagens que ignoram as teorias cujo objetivo é compreender a memória humana. O estudo de como essas teorias poderiam ser utilizadas para fornecer um módulo de raciocínio para um agente de *software* em um ambiente simples poderia incentivar o desenvolvimento de pesquisas que levassem a um novo paradigma de aprendizado em agentes. Dispondo de uma arquitetura que permite a um projetista especificar todo o curso de uma simulação, a fim de construir um ambiente virtual apropriado ao aprendizado, demonstram-se estruturas e processos de uma abordagem preliminar que poderia evoluir a um agente dotado de uma memória dinâmica com algum grau de robustez. Neste trabalho, a organização da memória é toda baseada em um formalismo desenvolvido em conjunto com os processos pelos quais suas estruturas são alteradas, partindo sempre de observações que causam a geração de instâncias que, por sua vez, são combinadas para a evolução de abstrações. Baseando-se em estruturas que representam estados, eventos e seqüências de eventos, a memória executa um ciclo de previsão, reconhecimento, falha e explicação que rege a evolução de estruturas abstratas, as quais conferem à memória potencial para prever novos estados do mundo. As expectativas sobre o estudo são de que ele possa contribuir para o desenvolvimento de pesquisas futuras sobre o tema.

Palavras-chave: *Memória Dinâmica, Aprendizado, Ambiente de Software.*

Abstract

The application domain of this work is a primitive software environment, consisting of virtual agents which perform a minimal set of reflexive actions without spatial motion, aiming to satisfy goals instantiated with messages structured according to a restricted language. Within such domain and by means of observations represented in an elaborate formalism which are collected at every simulation cycle, providing an observer agent outside the virtual environment with a dynamic memory to internalise changes in world state, in order to obtain predictions which reveal some degree of understanding about the relations among the events recognised, is the purpose of this study. Along the history of Artificial Intelligence, theories on memory structures which some authors believe to integrate the foundations of human reasoning are limited to applications where knowledge is static and to domains oriented to natural language understanding. Ubiquitously widespread agent-based systems have used methods whose foundations lie on approaches which ignore theories whose aim is desmistify human memory. Research on how such theories could be used to provide a reasoning module in a software agent within a simple environment might stimulate the development of works which conducted to a brand new paradigm of learning in agents. With an architecture which allows a designer to specify all the course of a simulation, in order to build a virtual environment appropriate to learning, structures and processes within a preliminary approach are presented; such approach might evolve into an agent with a dynamic memory displaying some degree of robustness. In this work, the organisation of memory is fully based on a formalism developed along with the processes whereby its structures are modified, starting always from observations which cause the generation of instances which, in turn, are combined and evolve into abstractions. With structures which represent states, events and sequences of events, the memory performs a cycle of prediction, reminding, failure and explanation which governs the evolution of abstractions; such structures enable the memory to predict new world states. The expectations about this work lie on a possible contribution to the development of future research on the subject.

Key words: *Dynamic Memory, Learning, Software Environment.*

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Contextualização

O desejo de que o computador pudesse resolver problemas que requerem capacidades de raciocínio humano tem alimentado as pesquisas em Inteligência Artificial (IA) desde os primórdios da computação. Suas sub-áreas têm apresentado notórias contribuições no tratamento de diversas classes de problemas.

Em particular, uma das grandes lições adquiridas ao longo de aproximadamente meio século de estudo foi a de que seres humanos dependem de uma grande quantidade de conhecimento para a execução de suas atitivades mesmo dentro de um domínio particular [Schank and Abelson, 1977, Lenat and Guha, 1989, Guha et al., 1990]. Por um lado, tentativas de se inserir manualmente o conhecimento em programas baseados em motores de inferência genéricos revelaram-se frustrantes; por outro, o desenvolvimento de algoritmos de aprendizado para a aquisição automática de conhecimento tem demonstrado severas limitações. Além disso, a maioria das técnicas de aprendizado é baseada em fundamentos estatísticos e matemáticos projetados para operar sobre registros de bases de dados, tornando-as desconexas com os esquemas de representação de conhecimento desenvolvidos ao longo da história da IA.

Uma tendência que culminou em um paradigma de *design* de programas e que se aproxima mais de um estágio de amadurecimento é a orientação a agentes. A concepção de programas como entes ativos com identidade em um ambiente dinâmico favorece a descrição e a interpretação de suas competências, no entanto a implementação dos módulos que integram a arquitetura de um agente pode sofrer dos mesmos problemas relativos à

aquisição de conhecimento. Embora agentes visem buscar mais naturalidade no processo de descrição de programas, os métodos de aprendizado existentes fogem a esse objetivo.

O aprendizado em seres humanos sucede-se por meio de experiências orientadas a objetivos [Schank, 1999]. Aprender é mais fácil para eles quando estão inseridos no contexto de um ambiente que lhes proporciona um cenário em condições ideais para que o aprendizado ocorra. Situações desse tipo, com cenários artificialmente construídos, têm sido usadas com sucesso na área de educação [Schank, 1992, Rickel and Johnson, 2000].

Com efeito, não tem sido raro também na Inteligência Artificial o uso de ambientes de *software* especialmente construídos para propiciar oportunidades de aprendizado a agentes de *software* e até mesmo a agentes que não dispõem de módulos de aprendizado [Tambe et al., 1995, Grand and Cliff, 1998, Rickel and Johnson, 2000, Isaac and Sammut, 2003]. O trabalho com simulações controladas facilita a tarefa de treinamento de um agente, pois propicia meios para a especificação de um roteiro de alterações no estado do mundo e, assim, orientá-lo a fornecer as sensações desejadas a um agente que nele habita e cujo objetivo é aprender.

Embora essa abordagem afaste-se das proposições defendidas por uma importante corrente da área, a EAI (*Embodied Artificial Intelligence*)[1], facilita-se a tarefa de estudar exclusivamente o desenvolvimento de uma ou algumas poucas competências de um agente sem se preocupar com problemas que não fazem parte do objetivo de estudo, principalmente quando os objetivos concentram-se sobre competências sofisticadas de um agente de *software* e que não envolvem o acompanhamento de percepções baseadas em sinais puros. Além disso, os custos, as exigências prototipais e os pré-requisitos em módulos poderiam tornar a abordagem defendida pela EAI inviável para o estudo de algum tipo de competência que se acredita encontrar-se em um nível superior da cadeia de circuitos cognitivos de um ser vivo.

Trabalhos relativamente recentes têm suportado a visão de que ambientes simples podem ser de grande auxílio a alguns ramos da pesquisa atual: Dzeroski et al. [1998] e Kaelbling et al. [2001] têm inclusive se interessado em desenvolver métodos de aprendizado utilizando-se do ambiente clássico conhecido como mundo dos blocos[2]. Uma abordagem cooperativa para a evolução de agentes ao longo de diversas gerações também foi testada

[1] A EAI surgiu em meados da década de 80 e prega que a pesquisa em IA utilize robôs e foque sempre um ambiente real a fim de, entre outros fins, inibir-se a construção de expectativas positivas sobre resultados promissores de testes em situações não reais que pudessem mais tarde vir a se revelarem frustradas em situações reais. Não é objetivo deste trabalho criticar ou apoiar tal posição.

[2] Esses ambientes remontam aos primórdios da IA, sendo amplamente conhecidos como *micro-worlds*.

no domínio de aplicação mundo dos blocos [Baum and Durdanovic, 2000].

Importantes pesquisadores [Minsky, 1975, Schank, 1982, Zadeh, 1999] argumentam ainda que é preciso espelhar-se em entes do mundo real no desenvolvimento de artefatos de *software* dos quais se espera a exibição de comportamento inteligente. Partindo-se desse princípio, o estudo de formas de se atribuir capacidades de aprendizado a agentes que levem em consideração objetivos, previsões e memória, na qual processos de aprendizado e estruturas de representação sejam projetados conjunta e harmoniosamente, revela-se um importante tema para a pesquisa atual em IA [Schank, 1999].

1.2 Problema

Ao longo de suas pesquisas, Schank tem proposto um grande número de estruturas de memória, juntamente com processos de operação dessas estruturas, para a representação de conhecimento em Inteligência Artificial [Schank, 1972, Schank and Abelson, 1977, Schank, 1982, 1986]. Entretanto, seus trabalhos anteriores a 1980 limitavam-se a demonstrar a validade da operação dessas estruturas mediante a utilização de uma memória previamente produzida de forma manual, com freqüência em aplicações de compreensão de linguagem natural; após essa data, Schank passou a expor ostensivamente a sua preocupação com a alteração automática das estruturas em memória, propondo assim a sua Teoria de Memória Dinâmica [Schank, 1982]. Para Schank, era evidente que se devia buscar desenvolver programas cujas memórias pudessem usar entradas para promover alterações em sua própria estrutura, a fim de manifestarem evidências de aprendizado.

No entanto, o desenvolvimento de programas que exibissem comportamentos desejados por Schank em sua teoria original não supriu as expectativas por ele mesmo alimentadas. Seja em função da alta complexidade das estruturas e seus respectivos comportamentos tais como ele propôs, seja em razão de que possivelmente essas estruturas pertençam a um estágio bastante sofisticado de uma memória[3], o fato é que a implementação de comportamentos previstos na teoria por ele proposta revelou-se difícil.

A ausência de trabalhos fundamentados nessas idéias e situados em domínios de aplicação mais simples, tais como o mundo dos blocos, pode ter prejudicado o interesse por essas pesquisas. Conceitos importantes mencionados por Schank, como estruturas do

[3] O que poderia ser uma influência do fato de que muitos de seus trabalhos anteriores terem sido orientados à compreensão de linguagem natural, a qual parece ser uma tarefa que requer uma memória bastante sofisticada.

tipo *estado*, foram muito pouco contempladas em seus trabalhos, sendo por ele mesmo referidas praticamente apenas em estágios iniciais [Schank, 1972]. Um estudo concentrado em um domínio de aplicação simples, que abordasse sua interpretação e fornecesse idéias para o desenvolvimento de uma memória dinâmica adequada poderia conferir importantes contribuições adicionais aos trabalhos originais e ajudar a resgatar um pouco do interesse por essas pesquisas.

1.3 Objetivos

Primeiramente, cabe realçar que este trabalho não objetiva sugerir métodos de aprendizado, formas de representação, metodologias de aquisição de conhecimento ou arquiteturas para aprendizado em agentes. Contudo, o trabalho visa oferecer idéias e exemplos de abordagens que podem ser úteis para trabalhos posteriores sobre o assunto.

1.3.1 Objetivo Geral

Realizar um estudo de caso preliminar de como poderiam ser empregados os conceitos de memória dinâmica segundo a definição de Schank [1982] para o desenvolvimento de um agente observador que possa registrar suas observações de um ambiente de *software* simplificado, consistindo em uma simulação com tempo limitado, a fim de obter conhecimento suficiente para efetuar previsões sobre estados futuros do mundo.

1.3.2 Objetivos Específicos

1. Constituir uma arquitetura de sistema que permita a um projetista deter o controle da simulação, permitindo a ele determinar o estado inicial do mundo, os ciclos da simulação e as alterações de estado do ambiente mediante arquivos cuja sintaxe propicie legibilidade para um usuário;

2. Simular um ambiente de *software* constituído por agentes virtuais que executam ações como resposta à instanciação de objetivos, a qual ocorre pela sucessão de mensagens textuais apropriadas. As representações do mundo devem respeitar algum tipo de formalismo de alto nível, agregando mais significado em contraste com o uso de sinais puros;

3. Desenvolver um agente observador que (i) receba uma percepção por ciclo de simulação contendo o retrato do ambiente no referido instante, e (ii) cujo componente cognitivo deve resumir-se a uma memória que busca reconhecer cada estado do mundo; na falha desse processo de reconhecimento, a memória deve alterar automaticamente sua estrutura, a fim de inibir a ocorrência da mesma perturbação num momento futuro, e abstrair padrões que a habilitem a dispor de previsões que evidenciem algum grau de compreensão das atividades transcorridas no ambiente;

4. Projetar a estruturação e organização da memória do agente observador, apresentando desde os tipos de estruturas necessárias e suas noções semânticas até as formas como essas estruturas participam no emaranhado global e os processos pelos quais elas são evoluídas.

1.4 Motivação

O uso de uma abordagem baseada na Teoria de Memória Dinâmica [Schank, 1982] para o aprendizado em agentes pode ser uma alternativa aos modelos matemáticos e estatísticos que predominam na pesquisa atual. Esses modelos não objetivam moldar o aprendizado da forma como ele ocorre em seres humanos.

Ao se almejar o desenvolvimento de técnicas que levem em consideração abordagens mais semelhantes às utilizadas pelos seres humanos no aprendizado em agentes de *software*, contribui-se para desmistificar como a memória humana funciona e quais poderiam ser os seus componentes. Esse conhecimento seria de fundamental importância para o desenvolvimento de programas com competências altamente sofisticadas.

A utilização de ambientes de *software* com simulações controladas que representam mundos simplificados, desprovidos de complexidades desnecessárias como entradas sensoriais de baixo nível, pode auxiliar a direcionar os passos iniciais mais facilmente. Abordar domínios de aplicação mais simples ajuda a definir problemas específicos menores, diminuindo a complexidade da tarefa de se propor estruturas e processos.

1.5 Organização da Dissertação

O restante do trabalho está organizado em quatro capítulos, sendo dois deles de suporte, um descrevendo as idéias propostas e outro apresentando as conclusões inferidas. Em

anexo, estão inclusos dois apêndices que trazem informações extras que podem ser convenientes ao leitor.

No **Capítulo 2**, é apresentada uma visão geral das teorias enunciadas por Schank, bem como os problemas relacionados à representação e a aquisição de conhecimento. Uma atenção especial é dada à noção de memória dinâmica.

Ainda em termos de suporte, o terceiro capítulo contém um apanhado geral sobre agentes de *software*, trazendo desde definições e taxonomias até casos de agentes dotados de algum tipo de aprendizado. Ambientes de *software* e suas características também são contemplados ao final do capítulo.

Em seguida, a descrição da arquitetura do sistema que define o ambiente, bem como a organização da memória do agente observador e em quê consistem seus processos, é exposta no **Capítulo 4**. Ao passo em que compreende o estudo prático que é foco da dissertação, esse capítulo exemplifica a evolução das estruturas de memória com o acompanhamento de uma simulação que permite ao leitor ter uma visão gradual do processo de aprendizado.

Contribuições, discussões finais e limitações deste trabalho estão dispostos no quinto capítulo. São sugeridos também temas para trabalhos futuros.

Ao final do documento, o **Apêndice A** transcreve a origem da palavra agente segundo o filósofo espanhol Mosterín. Por sua vez, o **Apêndice B** reproduz breves pensamentos dos pesquisadores Piaget e Vygotsky a respeito do aprendizado.

Capítulo 2

Memória Dinâmica

2.1 Considerações Iniciais

Um dos maiores avanços que ocorreram na Inteligência Artificial (IA), desde as pesquisas iniciais da área, foi a concepção de que seus problemas dependem de conhecimento de domínio e conhecimento de senso comum — sistemas que não se utilizavam de conhecimento e buscavam tratar problemas com métodos de resolução genérica não obtiveram sucesso [Newell and Simon, 1963]. Tornara-se possível a concepção de motores de inferência genéricos que, atuando sobre bases de conhecimento (BC), seriam ferramentas potenciais para a resolução de problemas em IA.

Para que conhecimento pudesse ser armazenado em bases, eram necessários esquemas de representação que proporcionassem a semântica requerida, a fim de que motores de inferência específicos estivessem habilitados a manipular o conhecimento representado. Diversos formalismos para representação de conhecimento têm sido propostos [Quillian, 1968, Minsky, 1975, Schank, 1972, Schank and Abelson, 1977, Sowa, 1984, Hayes-Roth, 1985] e utilizados em grande escala, embora os motores de inferência para tais representações sejam, por vezes, dependentes de domínio.

A experiência tem demonstrado que não é suficiente a existência de esquemas padronizados para se representar conhecimento se não for possível a aquisição de tal conhecimento de forma praticável. A aquisição manual, feita por engenheiros de conhecimento junto a especialistas de domínio, possui deficiências [McGraw and Harbison-Briggs, 1989].

Diversos paradigmas de aprendizado têm sido desenvolvidos e com efeito utilizados ao longo da história da IA [McCulloch and Pitts, 1943, Hunt et al., 1966, Cover and Hart,

1967, Holland, 1975, Sutton and Barto, 1981, Cheeseman et al., 1988, Watkins, 1989]. As técnicas clássicas oriundas desses paradigmas baseiam-se em métodos estatísticos e matemáticos na descrição de algoritmos. Entretanto, ainda não é consenso que seres humanos de fato aprendam por meio de métodos neuronais semelhantes. Sendo assim, essas técnicas desviam-se do modelo padrão de aprendizado em agente cognitivo que é sabido ser bem sucedido: o humano. Uma desvantagem disso é que uma vez que não se toma um modelo natural como base, é difícil argumentar que tais técnicas poderão perseverar e um dia ser parte de um modelo artificial de inteligência comparável ao humano. Além disso, o conhecimento[1] obtido com tais técnicas é por vezes difícil de ser atualizado — devendo-se, em alguns casos, iniciar-se um novo processo de aquisição e substituir os modelos de interpretação obtidos — e sofre com ruído, amostras pouco representativas e até com a quantidade de dados manipulados [Holsheimer and Siebes, 1994][2].

Dada a dinamicidade do mundo real, a necessidade de atualização do conhecimento e da sua utilização de forma natural direcionou a pesquisa à obtenção de conhecimento por meio de experiências. O armazenamento de experiências constitui o que se denomina memória em seres humanos. Entretanto, a memória é muito mais do que um repositório de experiências: é a fonte de grande parte da atividade cerebral humana, desde o reconhecimento de sensações até a execução de tarefas complexas [Minsky, 1985]. Além disso, a memória humana é organizada de tal forma que pode ser alterada dinâmica e ininterruptamente durante toda a vida de uma pessoa [Schank, 1999]. Assim, esse conceito passou a ser visto como um potencial modelo alternativo para a representação e utilização de conhecimento, de forma que experiências já armazenadas pudessem guiar a busca por soluções para problemas novos.

Dessa forma, compreender a organização e as atividades inerentes à memória humana é útil para o desenvolvimento de ferramentas computacionais que objetivem o tratamento de problemas complexos em ambientes dinâmicos. Para isso, é fundamental desenvolver-se modelos que busquem de fato emular o comportamento interno da memória humana. Neste ponto, a complexidade transfere-se para o uso de experiências anteriores como guia na busca por soluções para problemas novos.

[1] É importante esclarecer que o conteúdo de uma BC não é conhecimento puro, uma vez que o conhecimento não pode ser separado dos processos que evidenciam sua utilização [Rich and Knight, 1991, Schank, 1999]. O conhecimento está mais presente no uso que dele se faz do que em sua descrição.

[2] Apesar dessas deficiências, esses paradigmas de forma alguma podem ser considerados pouco úteis. Suas aplicações em diversas áreas do conhecimento têm comprovado sua utilidade para uma gama de domínios de aplicação como, por exemplo, *Data Mining*.

2.2 Experiências Originando Conhecimento

2.2.1 O Conhecimento nas Organizações

O ciclo de vida de um ser vivo consiste de uma interação contínua com o ambiente, na qual as experiências, embora únicas, tendem a apresentar padrões ora semelhantes, ora distintos entre si. Grande parte das ações — especialmente as conscientes — praticadas por qualquer ser vivo foi por ele aprendida em algum momento no passado. Muito pouco do que o cérebro — em particular o do ser humano — é capaz de realizar[3] parece ser inato [Minsky, 1985]. Como é possível tamanho aprendizado?

Embora pouco compreendam sobre suas próprias mentes, as pessoas conseguem interagir muito bem entre si. Preservando seu conhecimento, podem transmiti-lo de diversas formas ao longo de gerações. São ainda extremamente hábeis em cooperar e, assim, formam organizações que, por sua vez, também podem interagir entre si e cooperar, formando outras organizações e toda a sociedade.

Organizações, assim como seres humanos, dependem de uma grande quantidade de conhecimento para sobreviverem [Wiig, 1995]. Parte de tal conhecimento também é adquirido com experiências, embora provavelmente em menor escala que na proporção adquirido/inato em um ser humano[4].

Seja por vantagem competitiva, expansão, amadurecimento ou melhor desempenho nas suas atividades, aumentar o conhecimento sobre o mercado de atuação é interessante para uma organização. Como o sucesso de cada organização depende do estabelecimento de objetivos promissores e, obviamente, do alcance de tais objetivos [Wiig, 1993], é de suma importância que cada organização possa identificar quais planos podem alcançar quais objetivos e quais os planos mais adequados em cada situação. Organizações preocupam-se em atingir seus objetivos de uma forma que minimize os esforços com recursos limitados [March and Simon, 1958]. A observação dos comportamentos de organizações seguramente estabelecidas poderia ser um bom método para se aprender a interagir com o mundo. Uma vez que o mundo real é um ambiente dinâmico, adaptações em conseqüência de aprendizado poderiam ocorrer ao longo de toda a vida de uma organização.

[3] Logicamente, o cérebro não atua diretamente no ambiente, mas controla mecanismos de atuação.

[4] Isso é justificável porque boa parte do conhecimento de uma organização pode já estar com ela no momento de sua criação, pois é um conhecimento que pertence aos indivíduos que a constituem; além disso, quando uma organização incorpora novos indivíduos, ela anexa os conhecimentos dessas pessoas, o que é diferente da aquisição em interações com o ambiente.

Conseqüentemente, é possível a uma organização refletir ora mais competências, ora mais precisão no uso das competências que possui.

Toda a atividade de uma organização, mesmo em caráter passivo, faz parte do conhecimento dessa organização. Quanto maior seu conhecimento, maior sua capacidade de aprendizado [Wiig, 1993]. O aprendizado e o conhecimento estão relacionados aos padrões de interação de uma organização com o ambiente, sendo nessas interações que eles se manifestam. Assim, o conhecimento de uma organização é refletido pelas ações por ela tomadas, evidenciando o acúmulo de conhecimento por meio de experiências. Dessa forma, parece inadequado dizer que uma organização *usa* conhecimento — como um recurso ou habilidade; ele está presente na organização como um todo e manifesta-se nas ações efetuadas por ela, sendo, portanto, determinante para seu valor [Wiig, 1995].

A probabilidade de que um problema percebido encontrará uma solução organizacional aumenta conforme a organização tem recursos adicionais para recuperar soluções previamente criadas ou criar novas soluções [Cohen et al., 1972]. Assim, a compreensão da importância do conhecimento em uma organização gerou o desejo de se tentar localizá-lo, preservá-lo, valorizá-lo e mantê-lo. Uma organização de pessoas deveria reter algum conhecimento de seus esforços passados e condições do ambiente; se uma organização aprende, então esse conhecimento deveria estar disponível posteriormente [Duncan and Weiss, 1979]. Tal necessidade consolidou-se no conceito de construção de uma Memória Organizacional [Walsh and Ungson, 1991]; apesar do conceito de aprendizado organizacional [Argyris and Schon, 1978, Senge, 1990] ser diferente do conceito de memória dinâmica que será exposto na **Subseção 2.3.1**, o acúmulo de experiências também é desejável em uma memória organizacional.

No meio computacional, o conhecimento começou a ser explorado com a construção de ferramentas que utilizavam BCs para solucionar problemas. Essas ferramentas passaram a ser chamadas de sistemas baseados em conhecimento. Além de constituírem uma era representativa da história da IA, esses sistemas revelaram deficiências e dificuldades que fomentaram a diversificação das pesquisas e uma melhor compreensão dos desafios futuros.

2.2.2 Sistemas Baseados em Conhecimento

Uma parcela significativa das pesquisas em IA relaciona-se a sistemas baseados em conhecimento. Desses, os simbólicos orientados a regras — o que dificulta a representação de exceções, uma vez que regras tendem a ser rígidas e gerais [Riesbeck and Schank,

1989a] —, têm deparado-se com os problemas da aquisição, codificação e manutenção do conhecimento quando adquirido de forma manual [Shortliffe, 1976, Duda et al., 1979, McDermott, 1982]. Na verdade, o método de inclusão de conhecimento em ferramentas de forma manual é inadequado quando os problemas a serem tratados possuem complexidades semelhantes à do mundo real [Schank, 1999], uma vez que alterações significativas no ambiente podem degradar a eficiência da ferramenta.

Experiências têm mostrado que a quantidade de conhecimento específico e de senso comum utilizado pelos seres humanos, mesmo dentro de um domínio particular, é grande [Schank and Abelson, 1977, Lenat and Guha, 1989, Guha et al., 1990]. Conseqüentemente, a alimentação de ferramentas computacionais com *pseudo*-conhecimento[5] previamente pronto torna-se pragmaticamente difícil. Em virtude disso, essa abordagem foi aos poucos sendo descartada para dar lugar a métodos automáticos de extração de conhecimento baseados em padrões descobertos em conjuntos de exemplos.

A sub-área da IA denominada Aprendizado de Máquina (AM) utiliza algoritmos para aquisição automática de conhecimento a partir de exemplos com métodos indutivos [Holsheimer and Siebes, 1994, Mitchell, 1997] para alimentar sistemas baseados em conhecimento. Algoritmos clássicos de Aprendizado de Máquina que seguem tanto paradigmas simbólicos — como evolução de árvores de decisão — quanto subsimbólicos — tais como evolucionistas e conexionistas — utilizam abordagens de aprendizado não-incremental[6] com técnicas matemáticas ou estatísticas, que eliminam a necessidade da aquisição manual [Holland, 1975, Quinlan, 1986, 1993, Chauvin and Rumelhart, 1995]. No entanto, as formas como o conhecimento é adquirido e representado dificultam a sua interpretação posterior e, por serem métodos de aprendizado indutivo estático, apresentam desvantagens em ambientes dinâmicos [Valiant, 1984, Haussler, 1988, Kearns and Vazirani, 1994, Holsheimer and Siebes, 1994, Mitchell, 1997]. Sistemas não-dinâmicos — ou com aprendizado não-incremental — são mais adequados a domínios estáticos, o que exclui grande

[5]Embora o conteúdo representado segundo formas de *representação de conhecimento* clássicas seja denominado conhecimento, os processos de inferência necessários para interpretá-los e manipulá-los também pertencem ao "conhecimento" como um todo. Assim, o conteúdo das formas de representação não é conhecimento puro em sua forma elementar [Rich and Knight, 1991].

[6]O aprendizado estático ou não-incremental diz respeito à utilização de uma base de exemplos para treinamento, sobre a qual um algoritmo de aprendizado é aplicado e uma representação é gerada. Essa representação é, posteriormente, utilizada para a interpretação de novos exemplos. O desempenho do método é proporcional à representatividade da base de treinamento. Caso o ambiente sofra mudanças significantes que influenciem a forma como exemplos coletados são interpretados, uma nova etapa de treinamento é necessária, com exemplos representativos do novo comportamento do mundo.

parte dos domínios do mundo real [Riesbeck and Schank, 1989a]. De qualquer forma, a manipulação simbólica parece guardar certa coerência em relação aos processos naturais de aprendizado em seres vivos [Zadeh, 1999]; Minsky argumenta que métodos numéricos para representação de conhecimento são inerentemente limitados [Minsky, 1975].

O paradigma simbólico de Raciocínio Baseado em Casos (*Case-based Reasoning*) [Schank, 1982], mesmo quando utilizado de forma estática, tende a ser mais flexível que conjuntos de regras, uma vez que casos guardam características singulares naturalmente. O paradigma de orientação a casos estabelece que a interpretação de um novo evento é realizada buscando-se a estrutura mais semelhante já existente em memória e adaptando-se sua interpretação para reconhecer o novo evento que se deseja interpretar [Kolodner, 1993]. Outra vantagem é que casos tendem a ser facilmente compreendidos por seres humanos pela sua naturalidade [Schank, 1999], aspecto no mínimo desejável.

O Aprendizado por Reforço é um método de AM que possibilita adaptação rápida a variações do ambiente mesmo com poucos exemplos de treinamento. Segundo esse paradigma, uma função de prêmio retardado atribui valores maiores às melhores soluções. Entretanto, não é orientado a compreensão, nem adquire novas soluções que possam ser observadas no ambiente: a adaptação refere-se a qual das soluções existentes é melhor em um determinado caso, mas nenhuma "nova" solução é aprendida [Mitchell, 1997].

Em sua teoria denominada Memória Dinâmica (*Dynamic Memory*) [Schank, 1982], Schank defende a hipótese de que seres humanos resolvem grande parte de seus problemas inconscientemente ao serem recordados de situações passadas. Para ele, a memória humana é um sistema dinâmico por excelência e orientado a falhas — que tendem a perturbá-la, provocando nela alterações que a possibilitem comportar o novo caso —, e é dessa forma que sistemas computacionais deveriam tentar resolver problemas. Como é um método baseado em casos, cada problema é interpretado com o conhecimento do caso mais semelhante existente em memória. Assim, soluções para novos problemas passam a ser vistas como recuperação de soluções para problemas anteriores semelhantes, ao invés de produtos de planejamentos construtivos tal como em sistemas de planejamento [Fikes and Nilsson, 1971, Sacerdoti, 1975, Chapman, 1987]. Quanto mais distinto for o novo problema, maior pode ser seu potencial de provocar alterações nas estruturas de memória [Schank, 1999]. Na verdade, o paradigma de Raciocínio Baseado em Casos desenvolveu-se a partir da Teoria de Memória Dinâmica; entretanto, pela dificuldade de implementação das idéias nela defendidas — uma vez que a teoria foi definida apenas em alto nível —, a abordagem estática passou a ser utilizada. É importante ressaltar que

o desenvolvimento de um sistema baseado em memória dinâmica demanda uma grande complexidade na descrição dos comportamentos adaptativos da memória, bem como na organização das suas estruturas e no modo de acesso a elas. Uma memória dinâmica ainda necessita de uma adaptação lenta e gradual. O aprendizado em uma memória dinâmica é mais eficiente à medida em que se prepara cenários adequados às situações em que se deseja que o conhecimento detido seja de fato utilizado [Schank, 1999].

Assim, o formalismo de representação tem implicações fortes no sucesso de um sistema baseado em conhecimento; isso é devido, principalmente, ao fato de que representações de conhecimento atuam como substitutos para interações reais [Davis et al., 1993].

2.2.3 Conhecimento Factual x Conhecimento Pragmático

Com freqüência, depara-se com informações que possibilitam a aquisição de conhecimento factual (descritivo) [Wiig, 1995]. Esse tipo de conhecimento tem sua importância, mas é menos provável que a memória o recupere para ser usado em outras situações em que ele poderia ser mais adequado que o conhecimento pragmático [Schank and Cleary, 1995].

Conhecimento pragmático é aquele que é absorvido dentro do contexto das situações em que ele é, de fato, utilizado. As características dessas situações estabelecem relações minuciosas entre si, o que facilita que uma pessoa seja lembrada de qual ação tomar em uma nova situação a partir das experiências anteriores semelhantes [Schank, 1999].

Para melhor evidenciar as diferenças entre o conhecimento descritivo e o de utilização, é natural exemplificar-se: se uma pessoa que ainda não sabe andar de bicicleta solicita uma explicação detalhada a um exímio praticante, por melhor que seja o detalhamento, a pessoa leiga ainda não estará apta a de fato andar de bicicleta e, mesmo que empenhe-se a aplicar a maioria dos conceitos relacionados pelo praticante — dificilmente o leigo lembraria de todos —, certamente ele irá cair. No entanto, se o leigo insiste e começa a praticar, mesmo que com deficiências, a tendência é que ele melhore seu desempenho gradativamente e, em alguma tentativa futura, obtenha um desempenho satisfatório.

Este processo pode ser melhorado se o leigo observa com atenção algumas evoluções do praticante. Uma vez que memórias são constituídas de informações provindas de todos os órgãos dos sentidos, o apanhado visual pode ser muito mais fiel ao conhecimento prático necessário sobre andar de bicicleta do que a informação descritiva. A principal causa para isso é que pessoas descrevem seus conhecimentos pragmáticos com dificuldade, pois esses conhecimentos estão concentrados no uso e não na descrição [Schank, 1999]. Portanto,

verifica-se que observar as ações de um indivíduo em uma situação pode contribuir para se aprender a atuar de acordo com essa situação, pois uma observação é um caso que pode ser reconhecido por um estereótipo de uma cena.

2.2.4 De Casos a Conhecimento — A Abordagem Natural

Tem sido defendido, no contexto deste trabalho, o tratamento de problemas segundo abordagens que procurem manter alguma coerência com a forma com que os seres humanos efetivamente parecem tratá-los. No entanto, a maneira natural não está formalmente definida e, assim, torna-se difícil até estabelecer um ponto de referência para se buscar a naturalidade no tratamento de problemas. Ainda assim, uma classe de métodos têm sido apontada como alternativa: Raciocínio Baseado em Casos (RBC).

> "RBC pode significar adaptar soluções antigas ao encontro de novas exigências, usar casos antigos para explicar novas situações, usar casos antigos para criticar novas soluções, ou raciocinar a partir de precedentes para interpretar uma nova situação (semelhante a como advogados fazem) ou criar uma solução equitativa para um novo problema (semelhante a como mediadores de trabalho fazem)".
>
> *Kolodner* [Kolodner, 1993]

Diversas implementações têm sido desenvolvidas sob esse paradigma e algumas amplamente reconhecidas [Riesbeck and Schank, 1989b]. Uma vez que casos são resultantes de experiências, a utilização de casos é, portanto, uma abordagem em acordo com o quesito naturalidade [Schank, 1999]. A **Figura 2.1** ilustra o ciclo de um sistema de RBC. Quando um novo caso é apresentado à memória, inicia-se o processo de reconhecimento, que culmina com a ativação do caso em memória mais similar ao novo caso. Executa-se, então, uma adaptação do novo caso para sua posterior inclusão na memória. Uma vez adaptado, o caso é absorvido pela memória. Nesse momento, o caso está pronto para ser utilizado e a memória melhor preparada para reconhecer novos casos.

> "Casos, que representam conhecimento específico amarrado a situações específicas, representam conhecimento em nível *operacional*; isto é, eles tornam explícito como uma tarefa foi executada ou como um conhecimento foi aplicado ou quais estratégias particulares para a realização de um objetivo foram utilizadas. Além disso, eles capturam conhecimento que poderia ser difícil demais para ser obtido com um modelo genérico, permitindo raciocinar a partir de conhecimento específico quando o geral não está disponível. Outra vantagem de casos é que eles mantêm unido conhecimento que é utilizado unido. Um raciocinador que usa casos é salvo de ter de

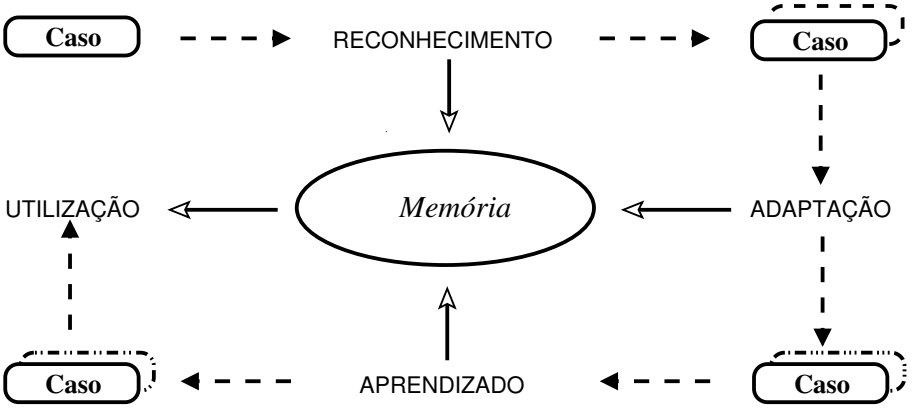

Figura 2.1: O Ciclo de RBC.

> compor várias peças descontextualizadas de conhecimento para resolver um problema. O caso guarda composições de conhecimento que já foram feitas. (...) Um caso é uma peça contextualizada de conhecimento representando uma experiência que ensina uma lição fundamental para se alcançar os objetivos do raciocinador".
>
> *Kolodner* [Kolodner, 1993]

Não está no escopo deste trabalho argumentar que casos são conhecimento[7]. De qualquer forma, o uso de experiências anteriores na interpretação e posterior atuação em um ambiente reflete, como já argumentado na **Subseção 2.2.1**, o conhecimento do respectivo indivíduo. Segundo esse ponto de vista, se a descrição de um caso em si não pode ser considerada como conhecimento, seu uso na compreensão de um novo caso, em conjunto com uma possível tomada de ação ou não, denota, a rigor, conhecimento. Com efeito, a habilidade no uso de uma informação é comumente denotada como evidência de conhecimento pelos próprios seres humanos.

A naturalidade da utilização de experiências possui ainda a vantagem de ser uma abordagem que possui um parâmetro de referência de sucesso: o ser humano. Em seres humanos, a memória cumpre o papel de ajudá-los a compreender situações e atuar adequadamente quando requisitado ou desejado, desde a realização de movimentos físicos até a comunicação em linguagem natural. Aliás, a habilidade exibida pelos seres humanos para comunicar-se possui alta dependência das estruturas e da organização de sua memória.

[7]Não porque isso não seja verdade, mas discussões desse nível transcendem os objetivos fixados para esta dissertação e remetem, possivelmente, a outras disciplinas — por exemplo, Filosofia. Lembra-se apenas que se deve incluir aqui os processos utilizados com os casos.

2.2.5 A Linguagem Natural: um Problema ou uma Vantagem para a Memória?

A comunicação em linguagem natural é uma das atividades que depende exaustivamente da memória, embora de forma não-consciente.

> "Assim como a maioria das atividades corporais — à parte de atividades como pulsação e respiração — a comunicação mediante linguagem natural parece ser, também, responsabilidade da memória — embora não exclusivamente. Além disso, a comunicação em linguagem natural — seja na leitura, escrita ou fala — é uma atividade não-consciente, uma vez que ao menos a maioria das atividades da memória não são conscientes.".
>
> *Schank* [Schank, 1999]

Dessa forma, RBC poderia vir a derivar uma abordagem válida para o tratamento de linguagem natural. Tal abordagem foi denominada *Case-based Parsing* e tem sido implementada como um processo de reconhecimento de conceitos em uma memória pré-modelada [Riesbeck, 1986, Martin, 1989a]. O primeiro *parser* baseado em casos foi denominado *Direct Memory Access Parsing* (DMAP) e movimenta marcadores em índices de conceitos em memória segundo entradas em linguagem natural [Riesbeck and Martin, 1985, Martin, 1993]. A idéia de movimentar marcadores para reconhecimento de conceitos em uma memória semântica foi inicialmente introduzida na implementação do *Teachable Language Comprehender* (TLC) [Quillian, 1969]; no entanto, DMAP possui uma diferença bastante significativa: é a memória quem guia o processo de reconhecimento, enquanto que no TLC são os termos das sentenças que determinam quais conceitos serão compreendidos. DMAP foi o sucessor dos chamados Analisadores Conceituais[8] — programas que mapeavam sentenças em linguagem natural para representações únicas[9]. A principal diferença entre DMAP e um analisador conceitual é que o último, assim como TLC, também era orientado aos termos das sentenças; além disso, analisadores conceituais devolvem uma representação conceitual como resultado da tarefa de análise, enquanto que as únicas saídas de uma análise feita pelo DMAP são as ativações de conceitos que podem ser observadas e a criação de instâncias simples.

Assim, DMAP repassa à memória a tarefa de tratar a ambigüidade e compreender sentenças — o que é uma estratégia que facilita o trabalho do *parser* —, enquanto que

[8] Um exemplo de programa que realiza análise conceitual na interpretação de sentenças em linguagem natural é o trabalho de Birnbaum and Selfridge [1981].

[9] Tais representações ainda serão discutidas à frente neste trabalho.

TLC sofre dos problemas relativos à ambigüidade [Riesbeck, 1986, Martin, 1993]. Uma vez que o sub-conjunto da linguagem natural utilizado está todo representado na memória, é ela quem deve decidir entre significados. Essa abordagem é totalmente diferente da abordagem clássica da área denominada Processamento de Linguagem Natural (PLN), na qual análises tais como léxicas, sintáticas e semânticas são tratadas em etapas separadas [Chomsky, 1965]. É improvável que seres humanos compreendam a linguagem através de uma análise que separa aspectos semânticos de aspectos sintáticos [Schank, 1999]. Na abordagem que utiliza memória, que tem sido chamada Compreensão de Linguagem Natural, tal diferenciação entre etapas não existe: todos os processos são realizados concomitantemente e orientados pela memória [Riesbeck and Schank, 1989b]. Seres humanos não esperam até o final de uma sentença para iniciar o processo de compreensão. Conforme as palavras são lidas ou ouvidas, elas vão sendo interpretadas. Na verdade, mesmo antes de serem lidas ou ouvidas, já existem expectativas na mente de uma pessoa que prevêem o que pode ser lido ou ouvido em seguida [Schank and Riesbeck, 1981b].

Uma dissertação desenvolvida nesta instituição utilizou-se de uma versão restrita de DMAP, denominada *Micro DMAP* [Martin, 1989b], para a compreensão de linguagem natural em um sistema que permite a modelagem de diálogos para situações específicas [Hembecker, 2002]. Esse trabalho demonstrou ser possível que um engenheiro de conhecimento crie modelos de memórias e diálogos — considerando-se apenas um sub-conjunto restrito da linguagem natural — através de uma interface gráfica amigável, a fim de que situações reais de diálogo contenham agentes de *software*[10] como participantes, utilizando DMAP como raciocinador. Embora o sistema não disponha de um módulo de aprendizado, ele facilita a tarefa do engenheiro de conhecimento: sua interface proporciona uma representação visual para a associação de termos a conceitos e para o acompanhamento da análise de um texto.

Apesar de com freqüência encarada como um problema, a linguagem natural também é uma ferramenta importante para o raciocínio: pensamentos podem ser construídos e raciocínio pode ser realizado através de representações internas em linguagem natural [Vygotsky, 1998]; assim, a comunicação em linguagem natural pode proporcionar um aumento significativo nas capacidades de aprendizado. Seres humanos, por vezes, refletem sobre situações ou ações através de construções da mente em linguagem natural. Dada a alta capacidade de representação da linguagem natural, sua flexibilidade e riqueza,

[10] Conceitos sobre agentes serão discutidos mais tarde neste trabalho, no **Capítulo 3**.

sua influência nas capacidades de raciocínio e aprendizado são positivas [Vygotsky et al., 2001]. Portanto, sendo a linguagem natural também responsabilidade da memória, os elementos resultantes dessa atividade tornam-se disponíveis à interpretação de outros tipos de eventos e podem conferir à memória um poder ainda maior de tratamento de problemas em domínios de aplicação distintos.

2.2.6 Dificuldades na Utilização de Experiências

É difícil medir quão útil uma experiência pode ser para um ente que aprende, assim como é difícil até mesmo estabelecer limites a uma seqüência de eventos a fim de denominá-la *experiência*. Neste trabalho, utilizar-se-á, conforme exposto no **Capítulo 3**, seqüências de eventos para representar experiências, porém nenhum controle do tempo que essas seqüências permanecerão em memória será feito — o que seria mais sensato.

Experiências constituem observações do mundo real. Algumas dessas observações podem ser incoerentes em relação a outras. A menos que uma memória dinâmica possuísse conhecimento suficiente para discernir tal anomalia, não há por que esperar que ela assim o faça. Considerar inconsistências como um fenômeno que exige tratamento foge ao escopo deste trabalho. Para o tratamento lógico de inconsistências, sugere-se que o leitor leve em consideração idéias de alguns trabalhos que se utilizaram da Lógica Paraconsistente para o tratamento de inconsistências em sistemas computacionais [da Costa et al., 1999, dos Santos et al., 2002, Ávila, 1996]. De qualquer forma, inconsistências podem existir e o conteúdo de uma memória dinâmica que visa representar o mundo real poderá eventualmente tornar-se inconsistente.

2.3 A Teoria de Memória Dinâmica

2.3.1 O que é uma MD?

A teoria de Memória Dinânica (MD) foi inicialmente proposta por Schank [1982] e desde então diversas pesquisas têm sido baseadas em memória e em RBC [Riesbeck and Schank, 1989b, Kolodner, 1993, Leake, 1996]. Há alguns anos, Schank publicou uma edição revisada de sua teoria ressaltando a relevância de suas idéias à área de educação [Schank, 1999]. Contando com diversos avanços ao longo das duas últimas décadas [Schank, 1986, Schank and Cleary, 1995], essa teoria permanece atual.

Até aqui, não havia sido definido neste trabalho em quê, de fato, consiste uma MD. Essa definição é melhor remetida ao próprio autor da teoria.

> "O que é uma memória dinâmica? É um sistema flexível e sem término[11]. Compare o modo como um especialista armazena conhecimento sobre livros em sua área de pesquisa ao modo como um sistema de catálogo de biblioteca faz o mesmo trabalho. Em uma biblioteca, um conjunto inicial de categorias é escolhido para descrever um domínio de conhecimento. (...) Eventualmente, as categorias terão que ser mudadas; categorias sobrecarregadas requerirão atualização; outras categorias terão que ser criadas para manipular novos assuntos e divisões de assunto. Uma biblioteca não tem uma memória dinâmica. Ela muda com grande dificuldade. Mais importante, para mudar ela requer intervenção externa. Um especialista não tem nenhum desses problemas. (...) Ele pode fazer observações sobre o que ele sabe e assim pode alterar as estruturas de memória que catalogam o que ele sabe. Ele pode fazer isso sem mesmo notar que fez. Ele tem uma memória dinâmica. (...) Nossas memórias mudam dinamicamente no modo como elas armazenam informação abstraindo generalizações significantes de nossas experiências e armazenando as exceções para aquelas generalizações. (...) Nossas memórias estão estruturadas em um modo que nos permite aprender com nossas experiências. Elas podem se reorganizar para refletir novas generalizações — de certo modo, um tipo de esquema de categorização automático — que podem ser usadas para processar novas experiências com base em antigas. Em resumo, nossas memórias ajustam-se dinamicamente para refletir nossas experiências. Uma memória dinâmica pode mudar sua própria organização quando novas experiências exigem. Uma memória dinâmica é por natureza um sistema de aprendizado."
>
> *Schank* [Schank, 1999]

Como já discutido na **Subseção 2.2.2**, uma MD não sofreria de problemas clássicos relativos a outros paradigmas de representação de conhecimento. No entanto, construir uma MD é bastante difícil; essa dificuldade concentra-se mais em conteúdo do que estrutura, embora estrutura também seja um problema difícil [Schank, 1999]. O que seria, então, introduzido em uma MD? Ou, uma vez que uma MD é um sistema que se adapta por si mesmo, como ela seria alimentada?

Entretanto, não basta armazenar: é necessário que o conteúdo armazenado possa ser utilizado. Uma memória dinâmica utiliza experiências anteriores para a compreensão de novas experiências, que podem alterá-la e modificar a forma como experiências posteriores serão por ela reconhecidas.

[11] Por *sem término*, Schank quer dizer que é um sistema que está em contínua modificação e, por isso, nunca alcança algum "estado final".

Alterações ocorrem sempre que, por algum motivo, acontecimentos do mundo não confirmam previsões da memória [Schank, 1982]. Logo, uma memória pode fornecer previsões para acontecimentos do mundo. Esse é um dos aspectos que determinam a compreensão: a capacidade de se poder prever quais eventos podem vir a ocorrer no futuro [Schank, 1999]. Na ocorrência de um evento inesperado, o que denomina-se falha, a memória deve buscar adaptar-se para que uma próxima ocorrência desse evento possa ser prevista e não ocasione, assim, falhas futuras.

Portanto, alterações em memória ocorrem em virtude de falhas de reconhecimento [Schank, 1982]. Falhas dependem da existência de previsões. Previsões são conseguidas quando se identifica no mundo padrões de seqüências de eventos existentes em memória. Esse processo de identificação é denominado lembrança [Schank, 1999].

Para que uma situação cause a lembrança de outra, é necessário que se estabeleça um critério de semelhança entre duas situações quaisquer. Seja qual for o critério, a lembrança deverá ocorrer. Se toda experiência fosse tratada como totalmente nova, pessoas não seriam de nenhuma forma inteligentes [Schank, 1999]. Seria impossível qualquer aprendizado, pois adaptações dependem de falhas, que dependem de previsões que, por sua vez, dependem de lembranças. Pode-se ainda continuar a cadeia de dependência e dizer que lembranças dependem de registros, ou alterações, em memória; logo, lembranças também dependem de falhas [Schank, 1999].

Todo esse ciclo requer tempo e observações adequadas. Observações perturbam e alimentam gradualmente uma MD. Portanto, quanto mais gradual for a complexidade das observações, melhor será o aproveitamento da MD na capitalização das informações. Poder-se-ia conduzir, assim, um indivíduo a um aprendizado mais eficaz se as experiências a que ele fosse submetido apresentassem complexidade gradual, em um ambiente e com seqüências de eventos favoráveis. O aspecto gradual do aprendizado facilita o reconhecimento e a indexação das informações em memória.

2.3.2 A Indexação de Estruturas em Memória

Toda nova observação oferecida a uma MD é comparada às demais experiências por ela já absorvidas. Essa comparação exige uma organização de memória eficiente. A essa organização, dá-se o nome de indexação. Casos (experiências) são indexados de acordo com as diferenças entre si [Schank, 1982].

Se uma observação puder ser considerada um caso específico de uma representação es-

tereotipada conhecida, a memória deverá necessariamente representar a nova observação em algum local sob essa abstração. Quando a similaridade da nova observação com as representações disponíveis em memória for apenas parcial, deve-se registrar uma representação para a nova observação através de um índice que evidencie essa diferença entre ela e as estruturas de memória existentes nesse local. Pode-se notar que uma organização hierárquica é fundamental para a satisfação desses requisitos. Representar diferenças em uma memória para que o reconhecimento proceda-se em um aspecto *top-down*[12] é a função primordial de um índice [Schank, 1999].

Schank não define formalmente como uma memória dinâmica deve ser organizada, porém fornece importantes idéias a respeito dos comportamentos que ela apresenta e sugere estruturas e processos que poderiam manifestar esses comportamentos. Para deixar um pouco mais claro como esse processo poderia ocorrer, pode-se simular o reconhecimento de uma nova entrada de acordo com as características de uma memória dinâmica até aqui discutidas. A princípio, a nova entrada é comparada com estruturas disponíveis ao topo da hierarquia. Se uma dessas estruturas estereotipadas for também uma abstração para o novo caso, o reconhecimento desse caso desce ao longo dessa abstração e o processo é repetido até que em algum momento não seja possível mais descer através de abstrações; nesse momento, o novo caso deve ser registrado em memória e um índice apropriado deve ser construído a fim de que essa falha não venha a ocorrer novamente. Cabe ressaltar que uma falha não é necessariamente depreciativa para uma memória dinâmica: é o modo pelo qual o aprendizado acontece [Schank, 1999].

Como já afirmado, todo índice deve refletir uma diferença fundamental entre uma estrutura e outras similares que torne possível a recuperação da estrutura indexada quando a diferença for reconhecida em uma nova estrutura. Portanto, o índice permitirá que a estrutura seja ativada apenas no momento em que ela for necessária. Isso significa que caso se deseje ser lembrado de uma observação qualquer, é preciso utilizar entradas adequadas para ativar essa observação. Alternativamente, pode-se induzir a memória a indexar observações de acordo com o modo como se espera que ela as reconheça no futuro [Schank and Cleary, 1995]. Isso sugere um aprendizado condicionado, em que dado um conjunto de objetivos, prepara-se um ambiente com situações semelhantes o suficiente àquelas que serão encontradas na cena em que se espera que o conhecimento aprendido seja manifestado. O conjunto de objetivos poderia ser, por exemplo, tarefas a serem

[12] Não se deve assumir que todo o processo de reconhecimento é *top-down* e sim que em alguns passos ele precisa proceder em uma abordagem *top-down*.

executadas; do ponto de vista de uma memória dinâmica, os objetivos seriam estruturas que deveriam ser ativadas a partir de um conjunto de observações. Essa abordagem é conhecida como Aprendizado Direcionado a Objetivo (*Goal-directed Learning*) e ambientes construídos para esse propósito designam-se Cenários Baseados em Objetivo (*Goal-based Scenarios*) [Schank, 1992, 1994, Schank and Kass, 1996]; são voltados à área de Educação.

Quanto menos falhas ocorrem num reconhecimento, mais precisas são as previsões para um dado conjunto de situações. Além disso, quanto mais diretas são as previsões, menos raciocínio inferencial — para buscar explicações para falhas — é necessário [Schank, 1999]. Andar de bicicleta é difícil para uma pessoa que nunca andou, pois várias de suas expectativas não são confirmadas; uma pessoa experiente não se preocupa em tentar imaginar a quantidade de força ideal a ser imprimida sobre um pedal em um determinado momento, pois isso já está coerentemente previsto em suas estruturas de memória. Essa previsão foi aperfeiçoada após muitas experiências, dentre as quais várias sem sucesso. Os insucessos não se repetiram muito e deixaram de acompanhar previsões que indicavam sucesso ao se andar de bicicleta; as mais apropriadas foram sendo reforçadas toda vez que se submete a uma nova experiência. Como apresentado na **Subseção 2.3.1**, a reorganização das estruturas em uma memória dinâmica é um processo contínuo e sem término.

2.4 Estruturas de Organização e Representação

A partir de 1970, diversas teorias para representação de conhecimento cujo ponto primordial de interesse era o significado [Minsky, 1975, Schank, 1972, Schank and Abelson, 1977, Sowa, 1984] foram desenvolvidas. Essas representações visavam englobar conceitos como eventos físicos, eventos mentais, intenções, causas físicas, causas mentais (razões), agrupamentos de intenções e eventos relacionados, previsões de conseqüências ou repercussões de eventos, entre outros. Ao longo desta seção, algumas das estruturas introduzidas nessas teorias serão revistas.

2.4.1 Dependência Conceitual

A Teoria de Dependência Conceitual (DC) foi desenvolvida por Schank [1972] e tinha como principal objetivo fornecer um formalismo para a representação de conhecimento obtido de sentenças em linguagem natural. Programas analisadores conceituais traduziam sentenças expressas em linguagem natural para representações em DC.

Uma DC representa o núcleo de um evento, que é invariante através de descrições; mesmo que a representação em linguagem natural de um evento seja diferente, a representação em DC deverá ser sempre a mesma [Schank and Riesbeck, 1981a]. Ainda de acordo com Schank, todo evento possui:

- um *ator*;
- uma *ação* executada por esse ator;
- um *objeto* sobre o qual a ação é executada;
- uma *direção* na qual a ação é orientada.

Embora alguns elementos não sejam explicitamente expostos em alguma sentença, eles sempre existem. Em alguns casos, informação sintática pode conduzir à determinação de quais objetos possuem quais funções; em outros, já não é possível nomear os encarregados de algumas funções pela ausência de informações na sentença, salvo quando a memória possui antecipadamente a informação necessária para isso; em outros ainda, tais objetos são considerados desconhecidos [Schank and Riesbeck, 1981b].

DCs são geralmente representadas como um termo composto de um argumento, cujo conteúdo é uma lista de *slots*. O nome do termo é denominado *predicado*. Cada *slot* consiste de um termo cujo nome é denominado *role* com um argumento denominado *filler* [Schank and Riesbeck, 1981b]. Esse formato permite que DCs sejam aninhadas, o que possibilita a representação de conceitos complexos. Assim:

- Uma DC pode representar um evento;
- O *predicado* representa o nome do evento;
- A lista de *slots* descreve as particularidades do evento;
- Um *slot* representa uma particularidade;
- Um *role* é o rótulo descrevendo uma particularidade;
- Um *filler* é o valor dessa particularidade, podendo ser outra DC.

Schank [1972] propôs também um conjunto de 11 ações primitivas que, segundo ele, combinadas, estariam hábeis a contemplar a maioria das ações existentes no mundo real.

Ação	Significado
ATRANS	Transferir a posse de um objeto.
PTRANS	Transferir um objeto de posição.
MTRANS	Transferir uma informação a alguém.
PROPEL	Aplicar uma força sobre um objeto.
MBUILD	Construir mentalmente uma idéia.
ATTEND	Prestar atenção em algo.
SPEAK	Pronunciar algo.
GRASP	Pegar um objeto.
MOVE	Mover uma parte do próprio corpo.
INGEST	Ingerir algo.
EXPEL	Expelir algo do próprio corpo.

Tabela 2.1: Ações Primitivas.

Essas ações estão descritas na **Tabela 2.1**. Alguns exemplos de como sentenças em linguagem natural podem ser representadas em DC utilizando as ações descritas na **Tabela 2.1** são apresentadas na **Tabela 2.2**[13].

Algumas ações — de nível mais baixo — são utilizadas apenas como instrumentos de outras — de nível mais alto. A representação de algumas sentenças em linguagem natural pode requerer que tais instrumentos sejam explicitados. Algumas sentenças em linguagem natural, e suas respectivas representações em DC, que exemplificam a noção de instrumentos são apresentadas na **Tabela 2.3**. Esse é um dos modos pelos quais uma DC pode depender de outra, de acordo com a teoria de DC [Schank, 1972] — o outro modo é por causalidade e será discutido na **Subseção 2.4.3**.

Porém, nem todo evento em um ambiente corresponde a uma ação. Situações do mundo são com freqüência reconhecidas como *estados*. Assim, alguns eventos podem corresponder a alterações de estado [Schank and Riesbeck, 1981b]. Infelizmente, a representação de estados não é tão privilegiada quanto a representação de eventos na literatura que trata de DC e nas teorias posteriores a ela relacionadas.

A teoria de DC foi concebida como uma forma de representação de conhecimento orientada a eventos. Ela é constituída de conceitos de baixo nível para diminuir o número de primitivas necessárias à tradução de sentenças de linguagem natural para DC. Além

[13] É importante ressaltar que as representações apresentadas nesta seção como exemplo são extremamente simplificadas, uma vez que foge ao objetivo deste trabalho apresentar uma representação cuidadosa e detalhada, o que demandaria um *background* literário mais adequado.

Exemplo	Representação
Mitsuo deu um celular a Kellen.	`atrans([ator(mitsuo), objeto(celular), de(mitsuo), para(kellen)])`
Mitsuo foi para a casa de Seiji.	`ptrans([ator(mitsuo), objeto(mitsuo), de(_), para(casa_seigy)])`
Mitsuo confessou ao professor sobre as provas.	`mtrans([ator(mitsuo), objeto(provas), de(mitsuo), para(professor)])`
Mitsuo descobriu a solução.	`mbuild([ator(mitsuo), objeto(solucao)])`
Mitsuo levantou as sobrancelhas.	`move([ator(mitsuo), objeto(sobrancelhas_mitsuo)])`
Mitsuo tomou a cerveja.	`ingest([ator(mitsuo), objeto(cerveja)])`
Mitsuo suou.	`expel([ator(mitsuo), objeto(suor), de(corpo_mitsuo)])`

Tabela 2.2: Exemplos de Representações.

disso, ela objetiva que duas sentenças quaisquer com o mesmo significado, não importando sua forma, possuam apenas uma representação em DC. Tais características diminuem consideravelmente o esforço em processos de inferência; em contrapartida, sobrecarregam o trabalho do engenheiro de conhecimento e do programador. Schank não faz referência à aquisição automática dessas estruturas. Cabe citar que a idéia de se preencher *slots* foi posteriormente utilizada por Minsky em sua Teoria de Frames [Minsky, 1975].

2.4.2 Frames

A idéia de recuperação de estruturas em memória, que remonta a Quillian [1968], foi sendo reforçada por diversas teorias principalmente ao longo das décadas de 70 e 80, entre elas a de *Frames* [Minsky, 1975]. Nessa teoria, Minsky adota uma conotação de conhecimento mais ligada a percepções visuais, ao invés de concentrar-se em linguagem natural. Para Minsky, um *frame* é um estereótipo de uma situação, de uma organização dos objetos no mundo. Ainda de acordo com ele, um frame armazenaria também informações a respeito de expectativas sobre o que poderia suceder a situação representada e quais restrições uma situação deve satisfazer para que determine a ativação desse *frame*. Segue-se um extrato da definição por ele oferecida.

Exemplo	Representação
Mitsuo chutou a caminhonete.	instrumentacao([trabalho(propel([ator(mitsuo), objeto(caminhonete)])), instrumento(move([ator(mitsuo), objeto(pe_mitsuo), de(_), para(caminhonete)]))])
Mitsuo avistou o xaxim.	instrumentacao([trabalho(mtrans([ator(mitsuo), objeto(imagem_xaxim)])), instrumento(attend([ator(mitsuo), objeto(olhos_mitsuo), para(xaxim)]))])
Mitsuo apresentou o PDM.	instrumentacao([trabalho(mtrans([ator(mitsuo), objeto(explicacao_pdm), para(professores)])), instrumento(speak([ator(mitsuo), objeto(explicacao_pdm)]))])
Mitsuo comeu o *sushi*.	instrumentacao([trabalho(ingest([ator(mitsuo), objeto(sushi)])), instrumento(grasp([ator(mitsuo), objeto(sushi)]))])

Tabela 2.3: Exemplos de Instrumentos.

"Aqui está a essência da teoria: quando alguém encontra uma nova situação (ou faz uma mudança substancial na visão do presente problema), seleciona da memória uma estrutura denominada *Frame*. Essa é uma estrutura recordada que será adaptada para se ajustar à realidade pela mudança de detalhes conforme necessário.

Um *frame* é uma estrutura de dados para a representação de uma situação estereotipada, como estar em um certo tipo de sala de estar, ou ir para uma festa de aniversário de criança. Anexos a cada *frame* estão diversos tipos de informação. Parte dessa informação é sobre como utilizar o *frame*. Parte é sobre o que alguém pode esperar ocorrer em seguida. Parte é sobre o que fazer se essas expectativas não forem confirmadas.

Nós podemos imaginar um *frame* como uma rede de nós e relações. Os níveis mais altos de um *frame* são fixos e representam coisas que são sempre verdade sobre uma situação suposta. Os mais baixos possuem muitos *slots* terminais que devem ser preenchidos por instâncias específicas ou dados. Cada terminal pode especificar condições que seus preenchimentos devem satisfazer (os próprios preenchimentos geralmente são "*sub-frames*" menores). Condições simples são especificadas por marcadores que poderiam requerer que um preenchimento terminal fosse uma pessoa, um objeto de valor suficiente ou um ponteiro para um *sub-frame* de um certo tipo. Condições mais complexas podem especificar relações entre as coisas associadas a diversos terminais.

...

Para análise de cena visual, os diferentes *frames* de um sistema descrevem a cena de diferentes pontos de vista e as transformações entre um *frame* e outro representam os efeitos de se mover de um lugar a outro. Para tipos de *frames* não-visuais, as diferenças entre os *frames* de um sistema podem representar ações,

> relações causa-efeito, ou mudanças em ponto de vista conceitual. *Frames* diferentes de um sistema compartilham os mesmos terminais; esse é o ponto crítico que torna possível coordenar a informação colhida de diferentes pontos de vista."
>
> *Minsky* [Minsky, 1975]

Apesar de ter contribuído com valiosas idéias a respeito de que tipos de informação deveriam constituir um formalismo de representação de conhecimento e como elas seriam modeladas e utilizadas, Minsky não privilegiou com a mesma generosidade aspectos de como o aprendizado de tais estruturas ocorreria. Aliás, ele próprio reconhece isso e declara que a teoria não estava terminada e constituiria apenas algumas sugestões e trabalhos a serem continuados [Minsky, 1975]. Uma vez que não há discurso suficiente sobre aprendizado, a implementação da teoria torna-se difícil.

2.4.3 Scripts e MOPs

Quando tentaram fazer seus programas compreenderem conexões entre sentenças, Schank e seu grupo perceberam que necessitavam de informações a respeito de relações causais; perceberam também que causalidade era um fenômeno que precisaria ser levado em consideração mesmo dentro de uma só sentença. De acordo com Schank and Abelson [1977], a maioria dessas conexões dependia de determinar quais ações resultam em quais estados e quais estados podem habilitar a ocorrência de quais ações. Schank e seu grupo utilizaram 4 tipos de ligações causais entre eventos [Schank and Riesbeck, 1981b] em seus programas, as quais são apresentadas na **Tabela 2.4**.

Entretanto, suas pesquisas mostraram-lhes que a determinação dessas conexões causais envolvia a necessidade de se ter disponível uma grande quantidade de conhecimento específico de uma situação. Schank e Abelson então propuseram que pessoas teriam estruturas de memória específicas para o armazenamento dessas informações e denominaram essas estruturas *scripts*. Como esse grupo de pesquisadores direcionava seus trabalhos à compreensão de linguagem natural, essas estruturas serviriam, entre outros, para assumir a existência de elementos em uma situação que não foram explícitos ao longo de uma seqüência de sentenças, mas que sem eles as sentenças não teriam sentido entre si. Além de constituírem estereótipos de situações, *scripts* podem fornecer ainda informações necessárias à compreensão do contexto no qual uma situação está envolvida e que não foram completamente mencionadas em uma sentença [Schank and Abelson, 1977].

Tipo	Significado
Resultado	Um evento pode resultar na mudança do estado de algum objeto envolvido no evento. Assim, "A lâmpada quebrou porque Mitsuo bateu nela" é uma relação causal de resultado.
Habilitação	Quando uma mudança de estado ocorre, algumas condições do mundo podem mudar em um modo tal que habilitam a ocorrência de algum evento que não poderia ocorrer antes. Assim, "Mitsuo comeu porque havia comida na mesa" é uma relação causal de habilitação.
Iniciação	Sempre que um evento ocorre, ou sempre que um estado ou um evento potencial existe, é possível que um ator possa estar ciente disso e, assim, vir a pensar sobre isso. Assim, "Mitsuo percebeu que Kellen estava infeliz porque ele a viu chorando" é uma relação causal de iniciação.
Razão	Uma vez que pessoas começam a pensar sobre coisas, elas podem vir a decidir fazer algo. Essa decisão é a razão para fazê-lo. Assim, "Mitsuo comeu peixe porque estava com fome" é um exemplo de relação causal de razão.

Tabela 2.4: Tipos de Relações Causais.

Posteriormente, buscando avanços em direção ao aprendizado de estruturas de memória, Schank revisou sua teoria de *scripts* e enunciou, enfim, sua teoria de Memória Dinâmica [Schank, 1982]. Nessa teoria, Schank introduz uma estrutura que ele denominou *Memory Organization Packet* (MOP) e que, segundo ele, organiza *scripts* — que ele renomeou para *scriptlets* — em memória e que permite a reutilização de partes de um *script* em mais de uma situação.

2.4.4 Planos e Objetivos

Ainda seguindo a linha proposta por Schank e seus colaboradores, mais dois tipos de estruturas se destacam: Planos e Objetivos [Schank and Abelson, 1977]. Em sua análise, objetivos proporcionam informação para se direcionar expectativas sobre eventos futuros. Expectativas estão diretamente relacionadas à compreensão e, assim, objetivos tornam-se elementos fundamentais em um processo de compreensão, pois permitirão obter previsões mais elaboradas em situações para as quais pode ser preciso tomar decisões.

Quando determinam situações específicas, objetivos podem ser alcançados por meio do cumprimento de um *script*. Entretanto, objetivos não vivenciados com freqüência por uma pessoa não são alcançados com um *script*, pois a pessoa não o terá adquirido. Nessas situações, as pessoas utilizam planos, que são estruturas genéricas, a fim de alcançar seus

objetivos [Schank and Abelson, 1977]. É importante ressaltar que, de acordo com Schank e Abelson, planos não são obtidos com planejamentos construtivos [Fikes and Nilsson, 1971, Sacerdoti, 1975, Chapman, 1987], mas sim recuperados pela memória. Dessa forma, planos também participariam do processo dinâmico de reconhecimento e lembrança. Ainda segundo Schank e Abelson, planos são estruturas bem mais sofisticadas que *scripts* e podem operar, por exemplo, organizando conjuntos de *scripts* disponíveis a fim de oferecer possibilidades de satisfação de um conjunto de objetivos que compartilham semelhanças.

Planos e objetivos certamente possuem grande importância no processo de compreensão. Entretanto, Schank restringiu-se a identificar alguns de seus tipos e descrever como eles se comportam; porém, pouco avanço foi realizado na direção do aprendizado de tais estruturas. Mais uma vez, percebe-se que o aprendizado configurou-se como um fator limitante para a compreensão.

2.4.5 Estados e Eventos

Estados são citados por Minsky [1975], Schank and Abelson [1977] e Schank and Riesbeck [1981b] como um conceito a ser representado. Entretanto, estados são pouco privilegiados em suas teorias[14]. Poder-se-ia subentender que ou a representação de estados é trivial, ou ela é de menor importância, ou ainda estados seriam apenas uma noção que ora deve ser representada com uma, ora com outra estrutura por eles introduzida. De qualquer forma, não está explícita em seus trabalhos essa definição.

Eventos são privilegiados por Schank em sua teoria de Dependência Conceitual [Schank, 1972], como observado na **Subseção 2.4.1**. Como ele objetiva a representação de linguagem natural, o conceito de evento por ele utilizado conserva influências desse domínio e a definição de evento é, por conseqüência, ligeiramente afetada.

Uma dissertação produzida nesta instituição introduz uma nova conceitualização para eventos e estados que nega as noções clássicas de Engenharia de Software e que é orientada à Representação de Conhecimento [Soares, 2001]. A semântica defendida nesse trabalho é a de que "um estado é uma condição aplicada a um objeto" e a de que listas de estados são objetos de eventos. Uma noção semelhante será utilizada neste trabalho e apresentada, mais tarde, no **Capítulo 4**.

[14] Embora citados com freqüência [Schank, 1972, Schank and Abelson, 1977], sua conceitualização é pouco discutida.

2.5 Considerações Finais

Conforme argumentado em diversos trabalhos amplamente reconhecidos [Minsky, 1975, Schank, 1982, Zadeh, 1999], é preciso espelhar-se em entes naturais no desenvolvimento de artefatos que se deseja exibirem comportamento inteligente. Uma vez que se objetiva imitar seus comportamentos, é uma boa estratégia tentar imitar também o modo como tal comportamento evolui a partir das estruturas que formam esses entes. Eles constituem um modelo que é sabido funcionar e que se pode estudar.

A utilização de métodos que não buscam reconstruir exemplos naturais possui também grande valor, pois essas técnicas têm obtido sucesso no tratamento de diversos problemas em domínios variados. Entretanto, não parece haver uma instância natural que utilize apenas métodos matemáticos e estatísticos, por exemplo, na solução de problemas; assim, não se possui um exemplo que é sabido funcionar. Isso acarreta incertezas a respeito de se, um dia, evoluções de tais métodos poderão conceder a um agente artificial a habilidade de exibir comportamentos sofisticados comparáveis aos dos seres naturais.

Assim, é perfeitamente aceitável que métodos de Aprendizado de Máquina clássicos sejam utilizados para tarefas pontuais, tal como auxílio à tomada de decisão em Descoberta de Conhecimento e outros domínios específicos. Entretanto, quanto mais próximo de um ser humano se deseja que um artefato computacional se comporte, menos aparente é o suporte que pode ser conseguido com técnicas que não objetivam reproduzir além do comportamento externo também a organização interna de agentes naturais.

De qualquer forma, é possível reconhecer a importância do aprendizado para os seres vivos e, por conseqüência, para agentes artificiais que necessitarão exibir comportamentos semelhantes aos dos seres vivos. O aprendizado é vital para a independência evolutiva de um agente artificial.

Capítulo 3

Agentes como Artefatos de Software

3.1 Considerações Iniciais

Ao longo de sua história, a computação tem observado o surgimento de paradigmas diversos em áreas diversas: os paradigmas de representação de conhecimento e de aprendizado indutivo da IA citados na **Seção 2.1**, os paradigmas de linguagens de programação e os paradigmas — ou metodologias — de desenvolvimento de *software* são alguns exemplos de classes de paradigmas relacionados à computação.

Dentre os paradigmas de desenvolvimento de *software*, destaca-se uma tendência dos últimos anos: Engenharia de *Software* baseada em Agentes (*Agent-based Software Engineering*) [Wooldridge, 1997, Wooldridge et al., 2000, Wooldridge and Ciancarini, 2001, Jennings, 2000, Petrie, 2001]. A evolução para este paradigma é uma tendência a voltar o projeto de *sofware* para aspectos mais humanos e fiéis ao mundo real, e pode ser deduzido da análise das linhas de desenvolvimento ao longo do tempo: *estruturada*, *orientada a objetos*, *orientada a agentes*. Entretanto, o conceito de agentes não se originou no contexto da disciplina de Engenharia de *Software* e remonta aos primórdios da sub-área da IA denominada Inteligência Artificial Distribuída (IAD)[1], em meados da década de 70. É importante ressaltar que os termos *agente* e *agente artificial* serão utilizados daqui em diante para representar apenas o conceito *agente de software*[2].

[1] A IAD é a sub-área da IA comprometida com o desenvolvimento de programas modulares: um problema é decomposto em sub-problemas que são resolvidos de forma separada e as soluções para os sub-problemas são posteriormente unificadas em, ou correspondem a, uma solução global para o problema inicial. Um clássico apanhado sobre a pesquisa em IAD é prestigiado em Bond and Gasser [1988].

[2] Diferentemente, portanto, dos significados mais genéricos de agente — *tudo o que age ou atua*, de acordo com o Dicionário da Língua Portuguesa *On-Line* (`http://www.priberam.pt/dlpo`) — e agente

3.2 Agentes de Software

A primeira referência ao termo *agente* é atribuída a Hewitt, no trabalho em que ele introduziu seu modelo de Ator Concorrente [Hewitt, 1977]. Nesse modelo, ele propôs o conceito de um objeto executando concorrentemente, interativo e auto-contido, o que ele denominou *ator*. Esse objeto possuía um estado interno encapsulado e podia responder a mensagens de outros objetos similares. Segundo Hewitt [1977], um *ator* "é um agente computacional que tem um endereço de *e-mail* e um comportamento. Atores comunicam-se por passagem de mensagens e executam sua ações concorrentemente".

Quando Hewitt introduziu esse conceito, ele obviamente o adaptou à sua aplicação específica. Hoje, esse conceito não é suficiente tampouco adequado para representar a variedade de aplicações e significados referidos pelo termo agente.

3.2.1 Definições de Agente de Software

Embora o conceito de agente não possua um consenso na comunidade acadêmica, outros autores fornecem suas visões:

por Franklin e Graesser:

> "*Um* agente autônomo *é um sistema situado em e uma parte de um ambiente que sente esse ambiente e atua sobre ele, através do tempo, em busca de sua própria agenda e assim afetar o que ele sentir no futuro.*"
>
> *Franklin e Graesser* [Franklin and Graesser, 1996]

Pode-se observar que Franklin e Graesser enfatizam a noção de agente como um ser dotado de autonomia, embora o termo *agente autônomo* soe ao menos um pouco redundante[3]. Essa autonomia é manifestada quando o agente busca alcançar seus objetivos,

artificial — um agente artefato ou, em outras palavras, um agente produzido pelo homem. Uma reflexão sobre o vocábulo agente em relação ao termo primitivo em latim do qual ele deriva é apresentada no **Apêndice A**, onde advoga-se a utilização do termo apenas em situações em que há um sujeito cuja intencionalidade é explícita, segundo tese do filósofo espanhol Mosterín.

[3]Com efeito, conforme será observado nas duas outras definições a seguir, a propriedade autonomia é referenciada como um requisito para que um determinado programa venha a ser reconhecido como um agente, o que parece excluir a hipótese de existência de um agente desprovido de autonomia; daí a redundância do termo *agente autônomo*. Entretanto, poder-se-ia utilizar o termo *programa autônomo* para, por exemplo, parcialmente descrever um agente. Vale ressaltar que essa é uma opinião particular do autor deste trabalho.

pois ele não depende de intervenção externa para isso. Igualmente importante é a noção de que um agente pode apenas ser reconhecido como tal caso lhe seja atribuído um ambiente: "mude o ambiente e nós podemos não mais ter um agente. Um robô apenas com sensores visuais em um ambiente sem luz não é um agente. Sistemas são agentes ou não com respeito a algum ambiente." [Franklin and Graesser, 1996].

por Nwana:

> "*Quando nós realmente precisamos, nós definimos um agente como referindo-se a um componente de* software *e/ou* hardware *que é capaz de atuar exatamente a fim de realizar tarefas no lugar de seu usuário.*"
>
> *Nwana* [Nwana, 1996]

Já Nwana reforça que será difícil chegar a um acordo sobre uma definição para o termo agente — assim como outros pesquisadores também já declararam. Entretanto, quando "ele realmente precisou", Nwana preferiu enfatizar que um agente está a serviço de um usuário final. De fato, a idéia de um agente como um assistente pessoal é uma das mais difundidas entre as aplicações que utilizam agentes [Maes, 1994a].

por Lange e Oshima:

- "... na perspectiva do usuário final,
  Um agente é um programa que assiste pessoas e atua por elas. Agentes funcionam permitindo que pessoas deleguem-lhes trabalho.
- ... na perspectiva do sistema,
  Um agente é um objeto de software *que:*
  - *está situado em um ambiente de execução;*
  - *possui as seguintes propriedades obrigatórias:*
    - *reativo: percebe mudanças em seu ambiente e atua de acordo com essas mudanças;*
    - *autônomo: tem controle sobre suas próprias ações;*
    - *direcionado a objetivo: é pró-ativo;*
    - *temporalmente contínuo: está continuamente executando.*
  - *e pode possuir qualquer das seguintes propriedades ortogonais:*
    - *comunicativo: hábil a comunicar-se com outros agentes;*

> * *móvel: pode viajar de um* host *a outro;*
> * *aprendizado: adapta-se de acordo com experiências prévias;*
> * *crível: parece crível para o usuário final."*
>
> *Lange e Oshima* [Lange and Oshima, 1998]

A definição fornecida por Lange e Oshima é um pouco mais detalhada. Novamente, o conceito de agente como um artefato a serviço de um usuário é realçado. Além disso, são apontadas propriedades que um agente deve possuir a fim de poder ser denominado tal, bem como propriedades adicionais que ele pode apresentar.

Pode-se ainda distinguir entre duas noções alternativas para agentes: uma *fraca*, que não apresenta nenhuma novidade em relação às definições já apresentadas — realça propriedades comportamentais —, e outra *forte*, ou metáfora de agente, na qual um agente é conceituado em termos de características como crenças e intenções, mais comumente utilizadas em relação a seres animados [Wooldridge and Jennings, 1995, Erickson, 1997].

A noção de autonomia compartilha um pouco da obscuridade encontrada na noção de agente. Para Wooldridge [1999], autonomia é a capacidade que um agente possui de atuar sem a intervenção de humanos ou outros sistemas.

Seja qual for a definição preferida, é importante estar claro que um agente de *software* sempre compõe um ambiente, sem o qual não parece ser relevante defini-lo. Além disso, um agente deve possuir objetivos — possivelmente vantajosos para si — que ele buscará continuamente e com algum grau de autonomia satisfazer — e para isso suas ações precisam de alguma forma afetar o que ele perceberá no futuro —, ainda que de forma reativa. Vale lembrar que implícita nessa compilação está a noção de tempo e que é através de janelas de tempo que se observa o ciclo de vida do agente.

O sutil desacordo das definições sobreditas corrobora a seguinte advertência de Russel and Norvig [1995]: "A noção de um agente deve ser a de uma ferramenta para análise de sistemas, não uma caracterização absoluta que divide o mundo entre agentes e não-agentes". Com efeito, tem sido mais fácil o simples uso do termo agente do que buscar um acordo sobre sua robusta definição, ou uma possível robusta definição. Enquanto as divergências contribuem por serem algumas das evidências da necessidade de mais pesquisas e fóruns — configurando-se, portanto, como um atrativo —, o uso claramente incorreto do termo, ao lado de demasiada pompa a ele atribuída, deprecia deveras a imagem da área de pesquisa.

Ainda segundo a mesma obra de Russel e Norvig, pode-se encontrar sua definição de agente como sendo "qualquer coisa que possa ser vista como percebendo seu ambiente

por intermédio de *sensores* e atuando sobre esse ambiente por intermédio de *atuadores*". Claramente, essa definição não se restringe apenas a agentes de *software*; mesmo assim, ela é particularmente interessante porque, ao contrário das definições apresentadas ao longo desta subseção, explicita-se a noção de sensores e atuadores — os componentes de um agente diretamente responsáveis por sua interação com o ambiente —, embora essa definição seja criticada por Franklin and Graesser [1996].

3.2.2 Agente x Programa

Pôde-se, na **Subseção 3.2.1**, corroborar a declaração na **Seção 3.1** de que há uma tendência de se trazer os artefatos de *software* para mais próximo da noção que os seres humanos possuem de mundo e de seres animados. A tendência é notável nas definições de agentes; as propriedades de um agente extrapolam a noção clássica de um programa computacional e transcendem a níveis metafóricos.

Singh sugere razões para o apelo de se ver agentes como sistemas intencionais.

> "Eles (i) são naturais para nós, como projetistas e analistas; (ii) proporcionam descrições sucintas de, e ajudam a entender e explicar, o comportamento de sistemas complexos; (iii) tornam disponíveis certas regularidades e padrões que são independentes da exata implementação física do agente no sistema; e (iv) podem ser usados pelos próprios agentes para raciocinarem uns sobre os outros."
>
> *Singh* [Singh, 1994]

Parece plausível considerar esses argumentos como vantagens de um agente em relação a um programa convencional.

Embora a meta original da pesquisa sobre agentes concentrara-se no estudo de modelos computacionais de inteligência, ou solução de problemas de forma distribuída, ao menos dois novos focos têm se fixado como aspirações nessa pesquisa: (i) simplificar as complexidades da computação distribuída e (ii) superar as limitações das atuais abordagens de interface com o usuário [Bradshaw, 1997].

Quando se versa sobre interfaces, algumas das limitações das abordagens convencionais são melhor evidenciadas: (i) ações apenas em respostas imediatas a interações do usuário, (ii) ausência de composição — ações e objetos básicos não podem ser compostos em outros de alto nível —, (iii) orientação a função, em detrimento do contexto da tarefa e da situação do usuário, e (iv) nenhuma melhoria no comportamento [Bradshaw, 1997].

Agentes podem compartilhar objetivos do usuário [Bradshaw, 1997]. Assim, torna-se mais claro por que agentes passaram a possuir um foco muito mais voltado ao de um assistente; tarefas cada vez mais abstratas e complexas lhes são delegadas pelos usuários, ao contrário do modo tradicional segundo o qual o usuário é requerido para executar seqüências de manipulações simples de objetos. Mais do que fazer de um agente um assistente, o que geralmente se espera dele é que ele por si comporte-se como tal.

Apesar de todo o aparato de um agente, não se deve assumir que qualquer aplicação necessita ser projetada segundo esse paradigma; é importante avaliar se é de fato vantajoso fazer uso dessa tecnologia. Dependendo dos seus requisitos, um sistema arbitrário poderia ser compreendido, desenvolvido e utilizado de forma mais simples e eficaz se ele fosse um programa convencional; em outras situações, pode-se inclusive adotar abordagens híbridas, em que agentes e programas convencionais coexistem. Portanto, a condução de um projeto orientado a agentes necessita de justificativas e da devida compreensão das suas vantagens e desvantagens. Uma compilação de razões de por que projetos baseados em agentes encontram problemas é exposta por Wooldridge and Jennings [1998, 1999]. Como forma de simplificar a discussão, pode-se dizer que um programa convencional pode ser uma melhor alternativa em comparação a um agente se os requisitos do problema não envolvem vantagens suficientes da tecnologia de agentes.

3.2.3 Arquitetura de um Agente

A implementação de um agente requer uma especificação dos componentes que o integrarão. Componentes, comportamentos, conhecimento sobre o ambiente e recursos administrados são elucidados em suas arquiteturas, onde são explícitos de forma abstrata. Kaelbling sugeriu uma definição para arquitetura de agente.

> "... uma coleção específica de módulos de *software* (ou *hardware*), tipicamente projetada por caixas com setas indicando os fluxos de dados e de controle entre os módulos. Uma visão mais abstrata de uma arquitetura é como uma metodologia geral para projetar decomposições modulares particulares para tarefas particulares".
>
> *Kaelbling* [Kaelbling, 1991]

Bradshaw et al. [1997] propõem uma arquitetura de agente que possibilitaria dispor-se de agentes executando tarefas úteis e ainda assim sendo de simples implementação. A arquitetura *KAoS* (*Knowledgeable Agent-oriented System*) é ilustrada na **Figura 3.1**. É possível observar tanto a estrutura interna de um agente quanto seu ciclo de vida.

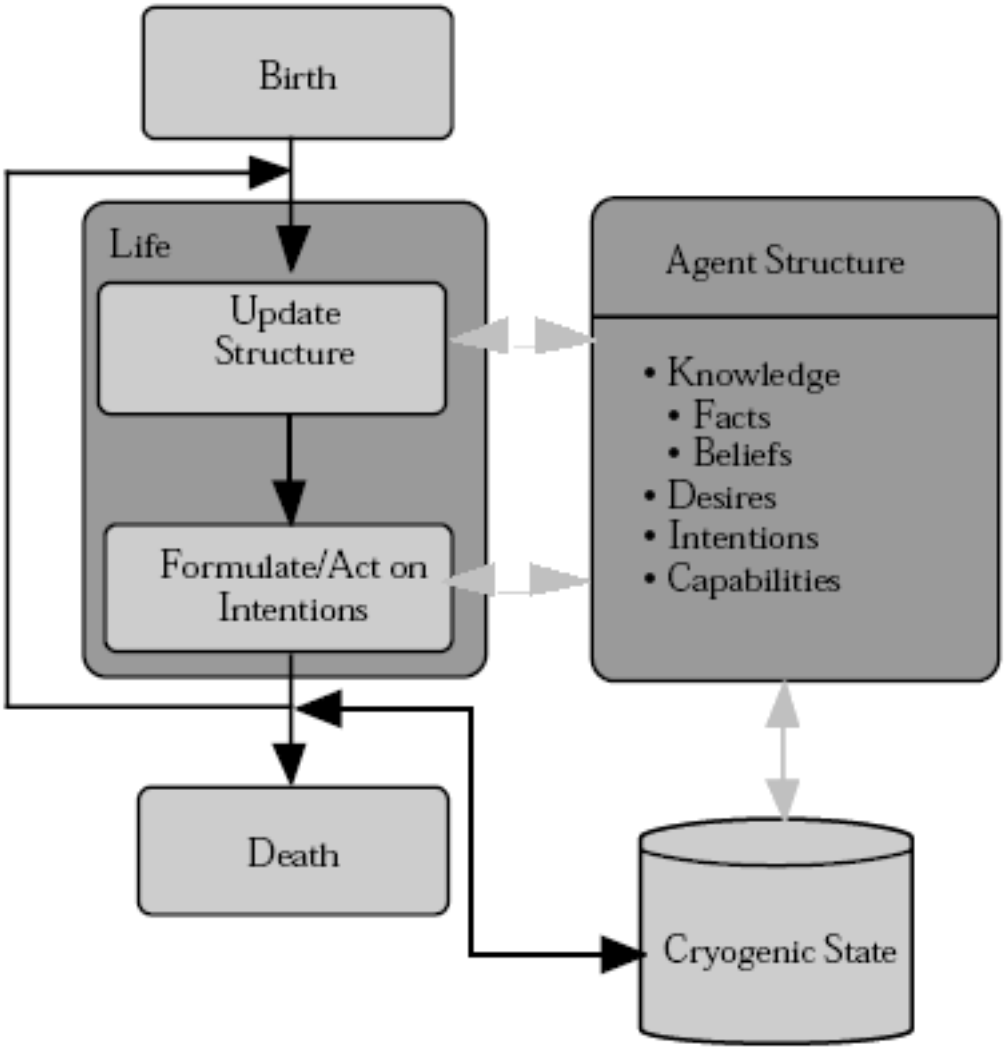

Figura 3.1: Estrutura e Dinâmica da Arquitetura *KAoS*. Setas pretas representam transições de estado e as cinza, fluxos de dados — reproduzido de Bradshaw et al. [1997].

Cada agente está constantemente atualizando sua estrutura para refletir as mudanças de estado do mundo. Eventualmente, o agente pode decidir por atuar no mundo, se isso lhe convir. Além disso, a qualquer momento ele pode assumir um estado *criogênico*, em que ele permanece suspenso ou inativo por certo tempo — uma espécie de vida latente —, podendo ser reativado e dar prosseguimento às suas tarefas, recuperando o estado exato em que ele se encontrava quando sofreu a transição para o estado criogênico.

Ainda na **Figura 3.1**, observa-se que o conhecimento detido pelo agente é constituído de fatos — crenças com total confiança do agente sobre si mesmo e sobre o mundo — e crenças — aquilo que o agente acredita ser verdade, mas que pode não estar consistente com o real estado do mundo, que na maioria dos casos será apenas parcialmente conhecido pelo agente. Desejos representam objetivos e preferências que motivam o agente [Bradshaw et al., 1997]. Para definir intenções, Bradshaw *et al.* referenciam Cohen and Levesque [1990], cuja definição é: "[intenções] representam o compromisso do agente a permanecer em um estado no qual ele acredita que de fato realizará algum conjunto de ações pretendidas". O último componente da estrutura do agente corresponde a um conjunto de módulos específicos, denominados habilidades ou *competências*, e relativos principalmente àquelas tarefas que o agente é capaz de realizar.

3.2.3.1 Taxonomia Deliberativo/Reativo/Híbrido

Analisando arquiteturas de agentes, pode-se distingui-los em categorias que auxiliam sua compreensão. Uma taxonomia importante no contexto deste trabalho é a que identifica diferenças entre agentes em relação à detenção de um estado interno — ou memória.

A arquitetura *KAoS*, exposta na **Figura 3.1**, integra uma categoria de agentes conhecidos como *deliberativos*, cognitivos, ou ainda intencionais. A caraterística determinante dos agentes deliberativos é que eles possuem estado interno simbólico, ou *memória*, o qual é utilizado pelo agente no processo de raciocínio que determina suas ações, cujos resultados podem inclusive ser previstos e previsões influenciar o processo de decisão [Wooldridge and Jennings, 1995, Nwana, 1996, Stone and Veloso, 2000]. Assim, um agente deliberativo não depende apenas de estímulos do ambiente para exercer alguma ação. Além disso, um agente deliberativo pode considerar postergar o processo de tomada de ação mediante a ocorrência de um estímulo, bem como aprender com suas experiências e elaborar planos visando seus objetivos.

KAoS, assim como outras arquiteturas, desenvolveu-se como uma variação de uma

arquitetura mais geral que constitui um exemplo clássico da categoria deliberativa: BDI (*Belief-Desire-Intention*), cuja teoria foi introduzida por Bratman [1987] e posteriormente mais fundamentada e estendida [Rao and Georgeff, 1991, Georgeff et al., 1999]. *Beliefs*, ou crenças, representam as percepções que o agente possui sobre o mundo; *desires*, ou desejos, representam estados que, caso fossem observados, seriam de grande satisfação para o agente; *intentions*, ou intenções, representam aqueles desejos que o agente decidiu tentar alcançar e para isso estabeleceu compromissos, geralmente elaborando planos para cumpri-los. A teoria original foi de fato implementada como arquitetura de agente em um trabalho posterior de Bratman et al. [1988].

Outro membro do ramo deliberativo desta taxonomia é `HOMER` [Vere and Bickmore, 1990]. Atuando em um "mundo-mar" bi-dimensional simulado sobre o qual ele obtém apenas percepções parciais, esse agente pode receber instruções, que limitam-se praticamente a apanhar e mover objetos, em um conjunto restrito de palavras em linguagem natural que podem ter referências temporais. Para atingir seus objetivos, `HOMER` cria planos e os executa, podendo alterá-los durante a execução se requisitado. Além disso, ele pode responder questões sobre suas experiências passadas.

Na outra extremidade desta taxonomia, encontram-se os agentes *reativos*. Desprovidos de estado interno, ou memória, seus comportamentos são basicamente respostas imediatas e sem raciocínio complexo sobre informação simbólica a alterações no estado do ambiente [Wooldridge and Jennings, 1995, Bradshaw, 1997]. Embora um agente reativo não possa aprender no mesmo sentido em que um agente deliberativo, ele pode certamente adaptar-se ao seu ambiente mudando o seu comportamento conforme transições que alterem significativamente o estado do mundo.

Um exemplo conhecido de agentes reativos é o trabalho de Agre and Chapman [1987], um pingüim denominado `PENGI` que, de forma simulada, joga *Pengo*, um jogo de computador. Partindo do princípio de que a maioria das atividades cotidianas são "rotinas" — no sentido de que requerem pouco raciocínio —, eles propuseram uma arquitetura com decisões codificadas em baixo nível e atualizações apenas periódicas, se necessário. Em *Pengo*, o mundo é bi-dimensional e constituído de blocos, abelhas e `PENGI`. Tanto `PENGI` quanto as abelhas podem empurrar blocos para deslocá-los. O objetivo de `PENGI` é não permitir que as abelhas entrem em contato com ele, pois o matarão. Agre e Chapman utilizaram uma representação de mundo deveras interessante, senão original: como `PENGI` é um agente reativo, ele não pode manter representações complexas do ambiente; assim, em vez de representar um bloco arbitrário como b_i, por exemplo, é utilizada uma re-

Ramo	Características
Reativo	• Ações apenas como resposta a estímulos • Não mantém estado interno • Sem raciocínio complexo sobre informação simbólica
Deliberativo	• Comportamento pró-ativo orientado a objetivos • Mantém estado interno • Previsões apuradas baseadas em memória influenciam decisões
Híbrido	• Alterna ou combina comportamentos reativo e deliberativo

Tabela 3.1: Taxonomia Deliberativo/Reativo/Híbrido. Uma nomenclatura para distinguir agentes em função da presença de estado interno em suas arquiteturas.

presentação relativa, ou funcional, da forma $o\_bloco\_em\_frente$. Isso permite ao agente concentrar-se apenas nos itens do mundo relevantes para si.

Ainda na taxonomia relativa à detenção de estado interno, existem também as abordagens *híbridas*. Agentes com essa característica podem, por exemplo, alternar comportamentos deliberativos e reativos, ou utilizar comportamentos deliberativos para melhor guiar os comportamentos reativos. A última opção foi desenvolvida para uma arquitetura de agentes inteligentes adaptativos que foi aplicada em diversos domínios, entre eles monitoração de pacientes em UTIs [Hayes-Roth, 1995].

A **Tabela 3.1** compila as informações apresentadas acerca da classificação de agentes em relação a memória. Taxonomias alternativas têm sido sugeridas; o leitor interessado é remetido a Franklin and Graesser [1996], Nwana [1996], e Stone and Veloso [2000].

3.2.3.2 Nomenclatura Alternativa para Estado Interno Simbólico

Considerando agentes deliberativos, uma distinção importante — ao menos em termos de nomenclatura — em relação ao conhecimento detido por um agente e comumente referenciada por autores é a de *self-model*, *acquaintance-model* e *environment-model*.

Self-model, ou modelo de si, representa todo o conhecimento que um agente possui sobre si, podendo incluir identificações de si, dos recursos por ele administrados ou dos quais ele possa eventualmente dispor, dos serviços que ele pode prestar ou as competências que ele pode exibir, seus objetivos, e as experiências a que ele foi submetido.

Por sua vez, o *acquaintance-model*, ou modelo(s) de outros agentes, corresponde ao conhecimento que um agente possui, ou adquire mediante interações, sobre outros agentes que possam existir no ambiente[4]. Os modelos de outros agentes podem tanto ser mapeados de forma semelhante ao *self-model*, como podem ser reconhecidos como pertencentes a outras arquiteturas e modelados de acordo. Geralmente, o *acquaintance-model* é um conjunto de modelos, um para cada agente do ambiente. Conhecimento sobre outros agentes que co-habitam o ambiente pode ser uma importante consideração nos processos de tomada de decisão do agente.

Demais itens são representados no *environment-model*, ou modelo do ambiente. Nesse modelo, são incluídas informações referentes, por exemplo, aos recursos existentes, à topologia do ambiente e às crenças que o agente possui sobre a evolução do tempo; a origem das informações são as percepções que o agente recebe do ambiente, geralmente correspondendo apenas parcialmente ao estado do mundo, podendo até mesmo serem ruidosas. O *environment-model* pode (deve) até englobar o *acquaintance-model*.

Os três modelos da nomenclatura alternativa[5] são ilustrados na **Figura 3.2**. Duas arquiteturas que se utilizam desses conceitos de modo bastante semelhante ao que foi apresentado são `ARCHON` [Wittig, 1992] e `INTERRRAP` [Müller and Pischel, 1994][6]. São comuns também abordagens que confidenciam mais ou menos informações a cada um desses modelos, geralmente omitindo a noção de *environment-model* [Wooldridge et al., 1999].

3.2.4 Agente x Multiagente

Como mencionado na **Subseção 3.2.3**, um ambiente pode ser habitado por mais de um agente. Tem sido amplamente designado *multiagente* qualquer sistema que é, ou pode-se considerar ser, (i) composto de múltiplos agentes e (ii) entre os quais há interação [Durfee and Rosenschein, 1994]. Ambientes com mais de um agente são também denomi-

[4] Abordagens com mais de um agente são denominadas *multiagente*, assunto da **Subseção 3.2.4**.

[5] Nomenclatura alternativa é um termo utilizado apenas neste trabalho e não deve ser encarado como uma convenção para denominar o uso de arquiteturas com os referidos modelos.

[6] É importante ressaltar que como não há um consenso em termos de arquitetura e nomenclatura, cada autor pode citar particularidades em arquiteturas de acordo com suas preferências pessoais.

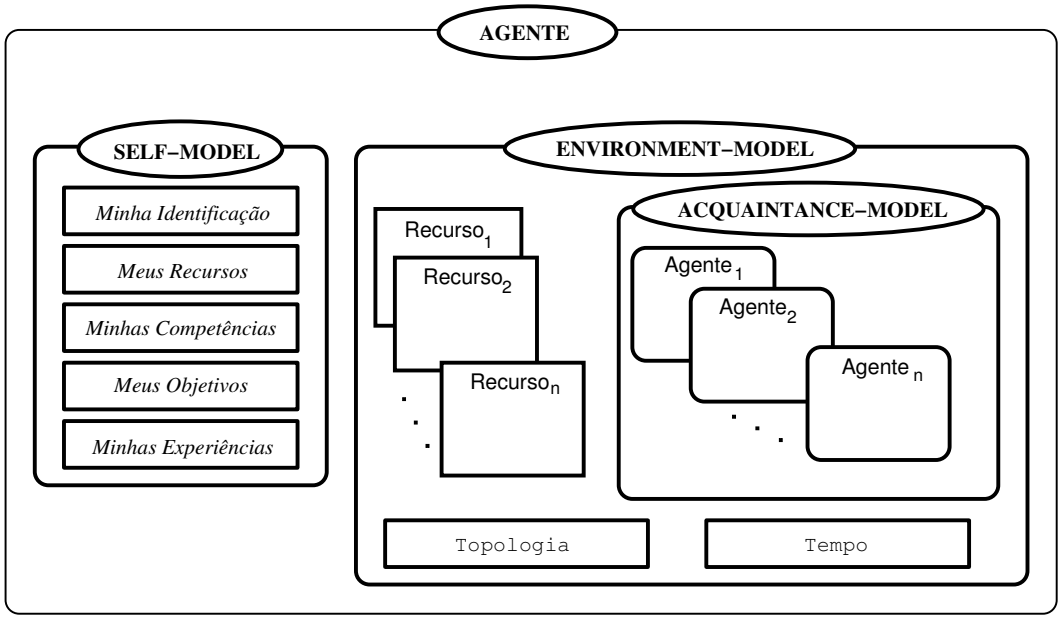

Figura 3.2: Modelos Internos de um Agente. Uma arquitetura parcial de um agente ilustrando seus *self-model*, *acquaintance-model* e *environment-model*.

nados distribuídos, enquanto que abordagens que utilizam um único agente são conhecidas como centralizadas [Stone and Veloso, 2000].

A abordagem distribuída apresenta os benefícios de operar separadamente sobre partes distintas de um problema. Sistemas multiagentes permitem que agentes detenham arquiteturas mais simples e módulos mais facilmente desenvolvidos, pois os integrantes do sistema detêm apenas visões parciais do mundo e são dotados de competências específicas para resolver somente partes de um problema, dispondo portanto de uma visão limitada [Jennings et al., 1998]. Cada agente mantém modelos de outros agentes e afeta o ambiente de forma imprevisível, aumentando as incertezas inerentes ao domínio [Stone and Veloso, 2000]. Em um sistema multiagente, tanto o controle quanto os dados encontram-se descentralizados [Sycara, 1998]. Um sistema multiagente é ainda uma abordagem mais natural à resolução de problemas que são fisicamente distribuídos ou cuja interpretação conceitual é mais facilmente compreensível quando são subdivididos em problemas de menor porte. Assim, a granularidade dos componentes de *software*, ou agentes, tende a ser diretamente proporcional à granularidade dos problemas que eles objetivam tratar, dos serviços que providenciam ou das tarefas que eventualmente possam desempenhar.

Entretanto, agrega-se ainda a necessidade de se estabelecer políticas de coordenação

para que um grupo de agentes, seja ele uma organização ou uma sociedade[7], dirija-se à satisfação de objetivos globais com eficácia. Uma série de problemas enfrentados na concepção de sistemas multiagentes é contemplada em Jennings et al. [1998].

Alternativamente, pode-se conceber um sistema composto de apenas um agente, embora tal opção seja bem menos apreciada pela comunidade científica. Apesar de a princípio não parecer, ambientes com um único agente geralmente são mais complexos de serem desenvolvidos do que ambientes multiagentes; se a tarefa a ser desempenhada pelo agente for suficientemente complexa, o esforço de desenvolvimento a ser empregado em um sistema com um único agente tende a ser maior, uma vez que a complexidade do agente será também maior [Stone and Veloso, 2000].

Em um ambiente centralizado, o agente modela o ambiente e a si mesmo. Mesmo se há outros entes autônomos nesse ambiente, eles não são reconhecidos como agentes — não são modelados como tendo objetivos e com todo o aparato constituinte de um agente — e são considerados apenas como sendo parte do ambiente [Stone and Veloso, 2000]. Em contraste, embora em uma esfera multiagente os seus integrantes sejam também parte do ambiente, um integrante arbitrário modelaria os demais explicitamente como agentes, a fim de melhor diferenciá-los no ambiente e analisar seus comportamentos individualmente.

Apesar de ser tentador apontar sistemas multiagentes como tecnologia superior a sistemas com um único agente, é imprudente assumir tal posição na ausência de um contexto. Assim como um programa convencional pode em circunstâncias específicas ser preferível a um agente, conforme argumentado na **Subseção 3.2.2**, um sistema centralizado pode ser uma melhor alternativa em uma aplicação específica em comparação com a abordagem concorrente. Aproveitando-se do argumento apresentado na **Subseção 3.2.2**, analogamente este trabalho considera que um sistema com um único agente pode ser a opção mais viável desde que não haja suficientes motivações para a escolha de um projeto multiagente.

Portanto, os critérios para determinação da opção mais vantajosa a ser escolhida para a implementação de algum sistema pairam mais uma vez sobre simplicidade na compreensão do problema, no desenvolvimento do sistema e na sua posterior utilização. Mais do que simplesmente desejável, é recomendável que todo problema possa ser abordado da forma mais simples e fácil possível.

[7]Não é objetivo deste trabalho o aprofundamento em conceitos organizacionais de sistemas multiagentes. De qualquer forma, para fins de situação do leitor, pode-se considerar (i) uma sociedade como toda uma população de agentes que estão de alguma forma relacionados e (ii) uma organização como um grupo determinado de agentes, incluso em uma sociedade, que possui características de uma entidade e um conjunto de objetivos próprios que seus integrantes buscam alcançar de forma cooperativa.

3.3 Aprendizado em Agentes

Certos ambientes — leia-se ambientes dinâmicos — requerem que agentes[8] que nele habitam alterem seu comportamento se desejam atuar com eficácia. As alterações de comportamento de agentes são reflexo das mudanças ocorridas em seu *habitat*. Embora muitas dessas alterações de comportamento possuam um caráter subjetivo por estarem relacionadas aos objetivos de um agente, não é insensato apontar que agentes buscam acomodar perturbações externas — no sentido de que o agente deve procurar aprender a prever estados do mundo que não estão sendo apropriadamente previstos, o que dificulta planejamentos eficazes para o cumprimento de seus objetivos.

A visão de aprendizado em seres humanos como um processo contínuo e temporal de acomodação a perturbações já foi defendida por Vygotsky [Luria, 2001] e Piaget [1983b][9]. Como apresentado na **Subseção 2.3.1**, Schank [1999] compartilha dessa interpretação.

Retornando ao discurso corrente, o aprendizado pode ser uma propriedade manifestada por agentes de *software*. Por meio da manutenção de uma memória, por exemplo, um agente pode modificar seu comportamento para se adaptar a alterações no ambiente e assim melhor perseguir seus objetivos. Esta seção é dedicada à apresentação de alguns sistemas computacionais que utilizam agentes com algum grau de capacidade de aprendizado atuando em ambientes dinâmicos.

3.3.1 Agentes Adaptativos

O termo *agente adaptativo* tem sido amplamente difundido e utilizado pela comunidade de IA para referenciar agentes que exibem comportamentos relacionados a aprendizado. Porquanto se estará tratando de aprendizado em agentes, torna-se importante contextualizar o uso do termo aprendizado neste trabalho em função da forma como os termos *adaptação* e *agente adaptativo* têm sido usados segundo as tendências correntes.

Agentes basicamente reativos contudo dispondo de recursos que os permitem alterar seu comportamento constituem uma parcela representativa na literatura que aborda agentes atuando em ambientes dinâmicos. Embora não armazenem progressivamente mais informação, há inclusive autores que consideram esse dinamismo comportamental, bem

[8] Entenda-se, neste parágrafo específico, o uso da palavra *agente* referindo-se a seu significado mais abrangente, ou original, e não apenas indicando o *sub*caso agente de *software*.

[9] Para fins informativos, o leitor pode conferir uma breve exposição de idéias de Vygotsky e Piaget, obtidas a partir de extratos de alguns de seus escritos, no **Apêndice B**, juntamente com informações a respeito do contexto histórico de seus trabalhos.

como formas de raciocínio que não agregam informação, como uma manifestação de uma propriedade que pode ser denominada adaptação [Odell, 1998, Greenstein and Odell, 1999, Odell, 2000b]. O relaxamento do conceito estende-se ao extremo de se considerar que um átomo de carbono ou um termostato são agentes adaptativos [Greenstein and Odell, 1999, Odell, 2000a] — e, portanto, todo agente de *software* viria também a ser.

De forma mais ampla, e possivelmente com mais coerência, tem sido considerado que ao estar apto a modificar seu comportamento em função de alterações significativas no que é captado pelos seus sensores em relação a estados prévios do ambiente, e com efeito fazê-lo, um agente exibe uma propriedade denominada *adaptação* [Sycara, 1998]. Para Decker and Sycara [1997], *adaptação* é o comportamento de um agente em resposta a eventos inesperados — de baixa probabilidade — em um ambiente dinâmico, tais como falhas não-programadas de um agente, de uma plataforma de agentes ou de uma fonte de informação. Franklin and Graesser [1996] consideram *agentes adaptativos* os que respondem com modificações em seu comportamento a alterações significativas no ambiente em que habitam baseando-se em experiência prévia.

Uma visão um pouco mais exigente de *agentes adaptativos* é encontrada em Brustolini [1991]. Segundo esse autor, *agentes adaptativos* são capazes não apenas de planejar, mas também de adquirir o conhecimento requerido para planejamento — conhecimento de domínio. Brustolini adverte ainda que aprendizado de domínio é bastante diferente do que algumas técnicas ditas de aprendizado realizam: aprendizado de domínio garante ao agente competências para realizar tarefas que ele antes não poderia efetuar, enquanto que há técnicas cujo ganho limita-se a melhorias de desempenho na execução de tarefas[10]. Embora essas técnicas sejam comumente referenciadas como de aprendizado, Brustolini prefere qualificá-las como *aprendizado de desempenho*, uma vez que elas não conferem a um agente qualquer habilidade que ele não pudesse já previamente desempenhar.

Agentes que, baseados em experiência, melhoram as competências que possuem para satisfazer seus objetivos ou motivações em um ambiente dinâmico são *agentes adaptativos* para Maes [1994b]. Ela também assumiu uma posição semelhante à de Brustolini, embora concentre suas críticas no Aprendizado por Reforço [Watkins, 1989]. Levando em conta as necessidades que *agentes adaptativos* deveriam suprir em ambientes complexos e dinâmicos, a pesquisadora enumera uma série de desvantagens dessa classe de algorit-

[10]Não cabe aqui referenciar essas técnicas, uma vez que o leitor interessado pode verificar o trabalho de Brustolini [Brustolini, 1991]. Para fins de informação, uma das técnicas referidas é conhecida como *Explanation-based Learning* (EBL), por acaso a mais referenciada por Brustolini.

mos, os quais são orientados a encontrar uma política de atuação que maximize o prêmio cumulativo conquistado a cada tomada de ação.

> (i) eles não tratam objetivos variando através do tempo — a política de ação aprendida considera um conjunto fixo de objetivos;
>
> (ii) se os objetivos mudam, os agentes devem reaprender tudo do início;
>
> (iii) para aplicações realísticas, o tamanho do espaço de estados — ou número de pares *situação–ação* — é tão grande que o aprendizado requer muito tempo para ser praticável;
>
> (iv) leva-se muito tempo para aprender longas seqüências de ação;
>
> (v) o modelo assume que o agente sabe a qualquer momento em que situação ele se encontra;
>
> (vi) é difícil inserir conhecimento inicial neste tipo de arquitetura;
>
> (vii) o modelo não pode aprender quando múltiplas ações são tomadas paralelamente.
>
> *Maes* [Maes, 1994b]

Convém citar que alguns dos inconvenientes salientados por Maes têm sido contornados em alguns trabalhos por ela própria citados. Outro trabalho posterior que forneceu idéias para superar alguns problemas em sistemas multiagentes foi o de Stone [1998], uma das quais é generalizar e dividir o espaço de estados entre os diversos agentes em um sistema multiagente, para reduzir a complexidade da tarefa de aprendizado.

A fim de condensar a discussão desta subseção e proporcionar uma categorização mais objetiva para o assunto, expõe-se na **Tabela 3.2** diferentes graus de mudança de comportamento em agentes, de forma ordenada[11]. A síntese parece estar de acordo com as visões dos autores mais representativos e seus trabalhos [Brustolini, 1991, Maes, 1994b].

3.3.2 Exemplos de Agentes Dotados de Aprendizado

Nesta subseção, apresentar-se-á alguns exemplos de trabalhos utilizando agentes e aprendizado na literatura. Alguns desses trabalhos [Pierce and Kuipers, 1997, Kaelbling et al., 2001, Baum and Durdanovic, 2000] exploram problemas relativamente próximos daquele tratado nesta dissertação.

[11] Entretanto, a categorização tem por único fim fornecer uma referência mais objetiva para este trabalho; não deve ser encarada como uma sugestão de categorização de agentes em relação a graus de alteração de comportamento.

Grau	Descrição e Exemplo	Categoria
I	*Um agente com comportamento constante em um ambiente sem dinamismo significante.* Ex.: Um agente virtual cuja única tarefa é periodicamente buscar comida em um ambiente simples. Embora a localização de sua comida possa também ser alterada periodicamente, o agente não descreve nenhum comportamento diferente em sua empreitada.	*Monocomportamento*
II	*Um agente que está hábil a optar sobre qual comportamento deseja exibir em determinada circustância dentre uma série deles que lhe foram previamente programados.* Ex.: Um agente tal qual o anterior, com o agravante de que o ambiente possui predadores dos quais o agente não pode se aproximar, mesmo que sua comida esteja próxima a eles, sob pena de ser devorado. O agente opta, então, entre os comportamentos `buscar_comida` e `evitar_predador`.	*Multicomportamento*
III	*Um agente que modifica gradualmente seu comportamento para melhor assimilar alterações significativas no estado do mundo — contudo, modificações tendem a ser apenas variações em parâmetros numéricos escalares. Embora lhe possa ser possível restabelecer um determinado comportamento já descrito, isso requer praticamente o mesmo tempo que lhe seria exigido caso esse comportamento ainda não houvesse sido exibido.* Ex.: Seja um sistema multiagente, onde agentes são nós de uma rede de roteamento e devem trafegar informação entre si. Cada agente só possui comunicação direta com um conjunto restrito de agentes que estão em sua vizinhança. Os agentes são dotados de um algoritmo de Aprendizado por Reforço, o qual lhes permite gradualmente convergir para as rotas mais eficientes de acordo com a carga da rede em um momento arbitrário.	*Adaptação*
IV	*Um agente adquire novas competências sem que isso necessariamente comprometa ou envolva o detrimento de outras já logradas.* Ex.: Um robô que, já sabendo andar de bicicleta, não perde essa competência depois de aprender a andar em um monociclo; ou ainda um robô que, já estando apto a se deslocar em um ritmo lento, não perde essa competência ao aprender a se locomover em um ritmo mais veloz.	*Aprendizado*

Tabela 3.2: Graus de Mudança de Comportamento.

Assistentes Pessoais de Maes [1994a] Uma das mais representativas linhas de desenvolvimento de agentes concentra-se no uso desses artefatos de *software* para assistir usuários de computador em algumas de suas tarefas cotidianas. Essa é uma classe de agentes que está profundamente associada às interfaces dos aplicativos; elementos constituintes das interfaces dos aplicativos passam de um comportamento passivo, em que toda tarefa é executada mediante intervenção direta do usuário, para um comportamento ativo, em que um agente aprende a partir da interação com o usuário a realizar tarefas monótonas que esse usuário periodicamente repete.

Para isso, Maes utiliza agentes que observam e aprendem interesses, hábitos e preferências de um usuário e também da comunidade à qual esse usuário pertence. Tipicamente, seus agentes:

- executam tarefas para o usuário — filtragem de informação, recuperação de informação, gerenciamento de *e-mail*, agenda, seleção de livros, filmes e música, entre outras atividades;
- podem treinar ou ensinar o usuário;
- ajudam diferentes usuários a colaborar;
- monitoram eventos e procedimentos.

Maes identifica dois problemas principais que dificultam a tarefa de desenvolvimento de assistentes pessoais: (i) competência — de que forma o agente adquire o conhecimento de quando, como e com o que ajudar o usuário — e (ii) confiança — como garantir que um usuário sentir-se-á confortável ao delegar tarefas a um assistente.

Duas abordagens clássicas para a construção de interfaces são comentadas por Maes. De acordo com ela, na primeira abordagem, na qual requer-se que o usuário crie agentes e programe todo o seu conhecimento, há um sério problema em relação ao critério de *competência*: o usuário pode ter dificuldades para reconhecer a oportunidade de utilizar um agente, pode não estar disposto a ter a iniciativa de criá-lo, alimentá-lo com conhecimento e manter esse conhecimento conforme necessário. Na segunda, bem mais comum, segundo a qual agentes são programados com uma extensa base de conhecimento sobre o domínio da aplicação e sobre o usuário — respectivamente, *modelo de domínio* e *modelo de usuário* —, Maes identifica problemas com ambos os critérios: há sobrecarga de trabalho para o engenheiro de conhecimento, pouco desse conhecimento ou dos módulos

da arquitetura do agente pode ser reutilizado — porque a maior parte é específica para a aplicação —, o conhecimento do agente é fixo — perdendo, portanto, a chance de ganhos com personalização — e uma vez que o agente foi programado por outra pessoa, é difícil evitar que o usuário mantenha desconfiança a respeito de suas competências.

A abordagem utilizada por Maes é a de um sistema multiagente cujo aprendizado de seus integrantes é baseado em técnicas de Aprendizado de Máquina. Dispondo de um mínimo de conhecimento do domínio, cada agente deve adquirir suas competências a partir do usuário e de outros agentes. Maes advoga que um agente que aprende pode gradualmente tornar-se mais útil e que esse desenvolvimento gradual possibilita que o usuário também construa um modelo de como o agente toma decisões, fundamento essencial para que o agente adquira a confiança do usuário. Compreender como e por que um agente realiza suas tarefas assegura confiança ao usuário.

Há diversas vantagens nessa abordagem na visão de Maes: o agente pode explicar por que tomou determinada ação, pode ser explicitamente instruído ou corrigido pelo usuário, requer menos trabalho tanto do usuário como do desenvolvedor, pode adaptar-se através do tempo e tornar-se mais personalizado a hábitos e preferências individuais e organizacionais, e ainda colabora no compartilhamento de conhecimento adquirido de diferentes usuários de uma comunidade.

Do ponto de vista técnico, Maes argumenta que seus agentes atingiram graus de competência e confiança bastante satisfatórios. As técnicas de aprendizado por ela empregadas são todas de Aprendizado de Máquina. O leitor interessado é remetido ao artigo de Maes [1994a], no qual ela descreve inclusive pormenores das implementações.

Outro sistema multiagente que utiliza-se de assistentes pessoais em uma arquitetura distribuída é o trabalho de Enembreck [2003]. Consistindo de agentes assistentes — que possuem um modelo do usuário e conhecem os serviços e as tarefas que o sistema oferece —, de coordenação — especializados na resolução de tarefas complexas que são por eles decompostas e executadas por agentes mais simples — e de serviço — cujas competências são simples e realizam tarefas atômicas —, o sistema implementa uma solução de recuperação de documentos personalizada. Dispondo de uma base de documentos de referência, o usuário seleciona aqueles que melhor representam o perfil de sua busca. Esses documentos são posteriormente utilizados para a criação de um modelo do usuário por meio de um algoritmo de Aprendizado de Máquina incremental proposto no trabalho. Em seguida, o modelo do usuário é utilizado para filtrar os resultados preliminares de uma recuperação baseada no modelo vetorial regular.

Agentes de Entretenimento de Grand and Cliff [1998] — Creatures Um *software* de entretenimento consistindo de agentes comportando-se como animais de estimação virtuais implementados como agentes, cuja arquitetura foi baseada em fundamentos biológicos substanciais, utiliza um método de aprendizado subsimbólico. Denominado *Creatures*, o *software* permite interação direta dos usuários com os agentes — ou "*creatures*" (*criaturas*) —, cujo *habitat* é um mundo virtual.

De acordo com os autores, o ambiente consiste em um mundo $2\frac{1}{2}$-*dimensional*, pois embora a visualização seja bidimensional, objetos podem ser localizados à frente ou atrás de outros, de acordo com alguns níveis. O ambiente contém diversos objetos com os quais os agentes podem interagir — brinquedos, alimentos, elevadores. Tais objetos, embora passivos, possuem *scripts* próprios que determinam como eles interagem com outros objetos no mundo e podem ainda ser movimentados pelo usuário.

Os agentes possuem tanto características fixas — bípedes — quanto outras geneticamente variáveis — cor, tipo do cabelo. O tamanho dos agentes aumenta conforme eles vão ficando mais velhos, até aproximadamente um terço de suas vidas, quando atingem a maturidade. Eles possuem sentidos simulados de visão, audição e tato, todos baseados em processamento subsimbólico — quando um objeto está dentro do campo de visão de uma criatura, um neurônio representando a presença desse objeto no campo visual torna-se ativo. De forma semelhante, sons são atenuados conforme a distância ou abafados por objetos entre o agente e a fonte de som.

Outra capacidade da qual as *criaturas* dispõem é a de aprender uma linguagem *verbo-objeto* simples, tanto por meio de interação do usuário com o teclado como de uma *máquina* existente no mundo com as quais elas podem brincar.

Os comportamentos de um agente são produzidos por uma rede neural sub-dividida em objetos chamados lobos, que definem características elétricas, químicas e morfológicas de um grupo de células. Células de um lobo interconectam-se com as de outros lobos para executar as várias funções da rede, ocasionando um alto número de sinapses.

O sistema bioquímico das *criaturas* possui, entre outras, a responsabilidade de controlar seu sistema imunológico. Algumas *bactérias* que co-habitam o ambiente podem causar inclusive a morte das *criaturas*, que são vulneráveis a ações de *bactérias*. Todavia, as *criaturas* estão hábeis a adaptar-se e aquelas que se revelarem menos vulneráveis a *bactérias*, por exemplo, tendem a contribuir mais para a evolução da espécie.

O ambiente é, em si, evolutivo. As *criaturas* podem se reproduzir quando atingem a maturidade. Uma vez que possuem genes, suas características são propagadas através de

gerações. Os genes de uma prole são obtidos por intermédio de operações de *cruzamento* e *mutação*, em probabilidades controladas, sobre os genes de seus pais.

Os autores garantem ainda que em certas ocasiões é possível observar até algumas evidências de comportamento social, tais como a cooperação de agentes em brincadeiras com uma bola ou cenas de perseguição. Além disso, *criaturas* podem aprender umas com as outras e mover-se de máquina a máquina por *e-mail* e *download/upload*. Segundo os autores, a forma mais objetiva pela qual o sucesso do projeto pode ser medido é a grande demanda que ele gerou entre os consumidores, tendo alcançado 400.000 unidades vendidas até o momento de redação do artigo. O trabalho de Grand and Cliff [1998] que baseou esta descrição apresenta ainda aspectos bem mais minuciosos de implementação, principalmente da rede neural e do sistema bioquímico.

Clone de Piloto Automático Orientado a Objetivo de Isaac and Sammut [2003] Em antítese ao discurso corrente, o referido trabalho não intitula o programa objeto como um agente, tampouco é esse programa dotado de aprendizado. Entretanto, o trabalho relata experimentos relevantes no contexto desta dissertação.

O trabalho traz contribuições para a pesquisa em *clonagem de comportamento* (*behavioural cloning*), além de ser uma extensão do trabalho original da mesma área [Sammut et al., 1992] e de trabalhos posteriores a ele. Em seu trabalho original, Sammut *et al.* empregaram o classificador C4.5, desenvolvido por Quinlan [1993], para abstrair o comportamento de um humano competente em um simulador de avião a partir da mineração de um conjunto de *logs* de simulações controladas. O experimento foi avaliado executando-se o simulador no modo *piloto automático*, com o comportamento do *piloto automático* sendo derivado da saída induzida pelo classificador C4.5. Embora o *piloto automático*, cujo comportamento consistia em uma abordagem exclusivamente reativa, obtivesse sucesso ao ser submetido ao mesmo percurso utilizado nas simulações controladas, ele possuía pouca robustez quando aplicado a percursos que exigissem manobras diferentes; além disso, sua compreensão era difícil pela inexistência de uma estrutura com objetivos explícitos [Isaac and Sammut, 2003].

Uma tendência observada em trabalhos subseqüentes foi a de se descobrir os efeitos de ações de controle e o modelo dos objetivos direcionando o controle. Isaac e Sammut reconheceram a relevância dessas abordagens, porém adotaram uma estratégia ligeiramente diferente: sua abordagem consistia em aprender ações de controle como uma aproximação das reações de um operador a diferenças entre o estado antecipado e o estado real do

sistema; o modelo dos objetivos direcionando o controle não recebeu alteração em relação a trabalhos anteriores que fizeram uso dessa técnica.

As competências foram então representadas por uma decomposição hierárquica em dois níveis: o nível de objetivo (antecipação) e o nível de controle (reação). O nível de objetivo modela como o operador escolhe metas de ajuste para sua estratégia de controle, enquanto que o nível de controle modela reações do operador a quaisquer desvios entre a meta e o estado real do sistema. Manobras foram então aprendidas individualmente, conferindo robustez às competências adquiridas.

Os autores realizaram um teste para comparar o desempenho de um clone com as especificações descritas contra o de um clone no qual apenas o nível de controle foi automaticamente aprendido, sendo o nível de objetivo manualmente codificado; o resultado foi amplamente favorável ao clone com objetivos também automaticamente aprendidos. Quando comparado aos clones tradicionais, o novo clone também se demonstrou bem mais robusto em relação a novos percursos, inclusive incluindo manobras não trabalhadas em tempo de treinamento.

A aplicação dessa estratégia é, conforme ressaltado pelos autores, restrita a domínios onde a decomposição hierárquica requerida é possível. A inclusão de aprendizado em tempo real é considerada pelos autores assunto para pesquisas futuras. Embora o trabalho não se constitua propriamente um exemplo de aprendizado em agentes, pode-se identificar características interessantes nas estratégias empregadas para a aquisição de conhecimento *offline*: a utilização de objetivos, a aquisição de competências a partir da exploração de manobras individuais e a clonagem de comportamento em si. Detalhes mais específicos da técnica são fornecidos no trabalho de Isaac and Sammut [2003].

O Agente que Aprende a Controlar seu Aparato e a Navegar no seu Ambiente, por Pierce and Kuipers [1997] Inicialmente, o trabalho supracitado define os conceitos de robô e agente, uma vez que essa distinção é contextualmente essencial. Nesse trabalho, o robô consiste de todo o material (físico ou simulado) que o agente deve aprender a usar. Não é conferido ao agente nenhum conhecimento prévio sobre o aparato sensorial ou sobre os atuadores de que ele dispõe. A representação desses elementos consiste de um vetor escalar s para os sensores e outro u para os atuadores. Cada elemento do vetor s corresponde a um número real representando o valor captado pelo respectivo sensor. Por sua vez, o vetor u contém números reais produzidos pelo agente para serem repassados diretamente aos atuadores do robô.

O trabalho objetiva a definição de um agente que possa evoluir modelos hierárquicos do ambiente e de controle dos sensores e atuadores do respectivo robô. Para isso, o agente dispõe de uma base de conhecimento que incorpora métodos descritos pelo autor como sendo de matemática básica, análise multivariada e teoria de controle. Nenhuma das técnicas faz qualquer suposição a respeito da estrutura ou dimensionalidade do domínio. Ainda de acordo com os autores, o domínio do experimento é estático — não muda a não ser por resultado de uma ação do agente[12] — e contínuo — utiliza representações de entradas/saídas agregadas em vetores de variáveis que assumem valores escalares pertencentes ao conjunto dos números reais.

Do ponto de vista experimental, o objetivo do agente é compreender seu mundo a fim de sobre ele realizar previsões e nele navegar. Por previsão entende-se a habilidade de prever os efeitos de um vetor u qualquer. Uma vez que não existem quaisquer posições referenciáveis no mundo *a priori*, o agente utiliza algoritmos *generate-and-test* para produzir objetivos randômicos correspondendo a coordenadas às quais o agente deve se deslocar.

Uma das heurísticas assumidas no trabalho é a de que dois sensores similares possuem leituras similares tanto de uma perspectiva pontual quanto na análise de sua distribuição. Entretanto, esse é apenas o primeiro passo em uma seqüência de camadas de aprendizado que vão sendo construídas passo a passo. Em um determinado momento, por exemplo, o agente aprende quais são os tipos de movimento que o robô é capaz de executar e quais os correspondentes valores do vetor u que executam cada movimento. Pode-se notar novamente o uso de um algoritmo *generate-and-test*, cuja persistência se dá ao longo de toda a cadeia de aprendizado.

Uma das contribuições do artigo está em evidenciar um conjunto de métodos que efetivamente resolvem o problema proposto, juntamente com um conjunto mínimo de requisitos para que a aplicação da técnica seja permitida. O artigo que descreve todo o processo é extenso e altamente técnico; entretanto, são demonstrados detalhes de implementação, exemplos e testes em situações diversas. Os autores fazem ainda uma afirmação forte ao dizerem que "pessoas não compreendem seu mundo em termos de seqüências de imagens visuais — elas usam abstrações de cenas visuais para locais e objetos".

Uma Discussão sobre um Agente em um Mundo de Blocos, por Kaelbling et al. [2001] Os referidos autores têm investigado o desenvolvimento de agentes que integram formas de representação clássicas em IA com métodos modernos de aprendizado e ra-

[12] No contexto desse trabalho, uma ação corresponde a um vetor u com ao menos um valor não nulo.

ciocínio na presença de incertezas. Trabalhando no domínio clássico mundo dos blocos, seu objetivo é o de construir um agente que possa aprender a executar determinadas tarefas com blocos. Essas tarefas consistem basicamente em construir torres de blocos.

A estratégia adotada é a da atribuição de prêmios para uma determinada ação, característica de Aprendizado por Reforço. Entretanto, ao contrário de outros trabalhos em Aprendizado por Reforço, o agente opera na presença de ruído. Kaelbling *et al.* preferem adotar formas de representação baseadas em lógica e com capacidades de representar um objeto segundo uma referência relativa a um indivíduo ou outro objeto — tal como a abordagem utilizada em Agre and Chapman [1987].

Apesar de não consistir na efetiva implementação de um agente, o trabalho supracitado faz importantes considerações a respeito da implementação de um agente com aprendizado segundo as técnicas e formalismos existentes em IA. Como referência, uma abordagem cooperativa de evolução de agentes cujos experimentos foram realizados, entre outros, no domínio mundo dos blocos é encontrada em Baum and Durdanovic [2000].

Pode ser destacado ainda o trabalho de Dzeroski et al. [1998], que apresenta uma variação do Aprendizado por Reforço, por eles denominado Aprendizado por Reforço Relacional. Os autores exploram meios de utilização de representações estruturais — estados, ações e objetivos — para abordar aplicações como planejamento no domínio de aplicação mundo dos blocos.

3.4 Ambientes de Software e suas Conveniências

Alguns dos trabalhos apresentados na **Subseção 3.3.2** foram desenvolvidos em ambientes cuja substância constituinte resumia-se predominante, senão exclusivamente, a *software*. Além de casos em que um problema reside naturalmente em um ambiente de *software*, a opção por um ambiente sintético pode também estar relacionada a outras razões comparavelmente pertinentes e vantajosas.

O advento de ambientes de *software* que permitem interação humana direcionando o fluxo de atividades simuladas deu origem a novas aplicações para agentes. Ambientes de simulação interativos têm tido aplicação em áreas como educação e treinamento [Rickel and Johnson, 2000], entretenimento [Grand and Cliff, 1998, Maes, 1995] e combate aéreo virtual [Tambe et al., 1995], entre outras.

Segundo Tambe et al. [1995], é com freqüência possível diminuir sensivelmente custos e requisitos com ambientes de *software* em comparação com sistemas robóticos ou sensoriais,

por exemplo, sem comprometer o desenvolvimento de competências em um agente e sem perder a riqueza do domínio de aplicação; ambientes de *software* não precisam tratar percepção e controle motor em baixo nível. Tambe *et al.* ainda diferenciam três tipos de ambientes relacionados a *software*: *sintético*, *de software* e *test-bed*.

Segundo essa classificação, o ambiente *sintético* diferencia-se do ambiente *de software* na medida em que requer interação em tempo real com mundos dinâmicos e sobre os quais se dispõe de acesso a apenas um limitado conjunto de informações; dessa forma, técnicas de planejamento tradicionais seriam aplicáveis apenas a ambientes *de software* e não a sintéticos. Os ambientes *test-bed*, por sua vez, seriam ambientes construídos especialmente para validar uma determinada técnica ou sobre os quais se possuiria amplo controle; ambientes sintéticos seriam domínios *reais* desenvolvidos comercialmente para entidades governamentais ou empresas que deles precisam para uso privado. Assim, os desenvolvedores de agentes para ambientes sintéticos não teriam altos graus de liberdade para pré-estruturar o ambiente, escolher quais de seus aspectos são importantes ou instrumentá-lo para propósitos experimentais [Tambe et al., 1995].

Em razão da confusão que pode ser causada com a utilização do termo *ambiente de software* para indicar apenas um dentre três ambientes constituídos possivelmente em sua totalidade de *software*, este trabalho não adota tal taxonomia e refere-se a ambiente de *sofware* como sendo qualquer ambiente abrangido por uma dessas categorias. Entretanto, por razões de informação, o sistema descrito ao longo do **Capítulo 4** comporta-se como um ambiente *test-bed*, segundo a classificação sugerida por Tambe et al. [1995].

Extensões das formas de interação mais usuais entre agentes de um simulador de ambiente podem ser encontradas em ambientes de realidade virtual. No trabalho de Rickel and Johnson [2000], um agente com representação visual de seu corpo, *Steve*, possui capacidades de interação com humanos por diálogo em linguagem natural, gestos, expressões faciais, movimento corporal e movimento dos olhos. O objetivo de *Steve* é colaborar com estudantes humanos tal qual um tutor, a fim de que eles possam aprender a executar tarefas procedimentais. Ao demonstrar a execução de tarefas, bem como solicitar e acompanhar ativamente sua execução por estudantes — eventualmente apontando equívocos e indicando o procedimento correto ou simplesmente fornecendo ajuda na forma de respostas a questionamentos —, *Steve* e seu mundo proporcionam um rico ambiente de aprendizado para estudantes [Rickel and Johnson, 2000]. Uma diferença importante, no entanto, é que o ambiente virtual habitado por *Steve* é mapeado para corresponder a um ambiente real onde estudantes podem interagir.

3.5 Considerações Finais

Ambientes de *software* podem proporcionar uma plataforma útil ao aprendizado. Seja o sujeito da aprendizagem um ser humano ou um agente de *software*, um ambiente de *software* pode garantir desafios semelhantes aos encontrados no ambiente objetivo, enquanto que reduzindo os custos de um protótipo. Além disso, um agente pode receber constantemente informações a respeito do estado do mundo em um nível de representação mais alto do que o obtido com sensores que captam informações do mundo real.

Ao se dispor de observações de um mundo localizadas temporalmente, pode ser possível construir modelos para a sua interpretação mais fiéis do que aqueles que eventualmente descartassem tal informação. Um agente detentor de uma memória devidamente preparada para acompanhar alterações no mundo através do tempo poderia evoluir um modelo útil para prever eventos futuros.

Ao estar hábil a prever eventos futuros e os próximos estados do mundo a partir de observações, um agente poderia acompanhar até certo ponto com eficiência o comportamento descrito por outros agentes no *habitat* que compartilham. Ora, competências exibidas pelos agentes coabitantes seriam, então, candidatas a integrar o conjunto de competências detidas pelo agente observador.

Uma estratégia para a implementação de previsões poderia ser o reconhecimento de padrões de seqüências de eventos que, sob a forma de abstrações, configurar-se-iam como fontes para previsões. O processo de reconhecimento de uma nova seqüência em execução consistiria no acompanhamento das previsões obtidas a partir das abstrações. Ao final do reconhecimento da nova seqüência, uma nova instância poderia ser adicionada em memória para representar o novo caso e alterações em estruturas abstratas poderiam ser necessárias para melhor refletir o novo estado da memória.

Capítulo 4

Abstração de Seqüências de Eventos em um Ambiente de Software

4.1 Considerações Iniciais

O aprendizado é uma das competências mais sofisticadas que um agente pode apresentar. Seja no nível simbólico ou no subsimbólico, essa competência capacita um agente a aperfeiçoar seu processo de decisão, adaptar-se a alterações no ambiente ou mesmo exibir um comportamento novo. A escolha do método que conferirá essa capacidade a um agente depende, entre outros, do objetivo almejado e do domínio de aplicação.

Este trabalho possui forte influência da Teoria de Memória Dinâmica [Schank, 1982] e de trabalhos precedentes que nela culminaram [Schank, 1972, Schank and Abelson, 1977]. Trabalhos mais recentes, derivados dessa teoria, também constituem fonte de influência para as idéias desenvolvidas neste capítulo; em particular, os Cenários Baseados em Objetivo (*Goal-based Scenarios*) [Schank, 1992] podem ser comparados ao modelo adotado neste trabalho para a organização de uma simulação em um ambiente de *software*. Os Cenários Baseados em Objetivo são situações criadas especialmente para a condução de uma tarefa de aprendizado; são construídos com base nas competências que se deseja que sejam adquiridas por um conjunto de indivíduos em um processo educacional. Cenários Baseados em Objetivo são aplicados em situações reais de aprendizado.

A organização e a execução de uma simulação na qual um agente observador perceberá o estado do ambiente a cada ciclo da simulação possui semelhanças com os Cenários Baseados em Objetivo. Uma simulação como essa poderia ser descrita por um conjunto de

configurações especialmente escolhidas para a tarefa de aprendizado. O agente observador poderia abstrair seqüências de eventos de forma a poder empregá-las no reconhecimento de novas seqüências e conseqüente previsão de eventos e estados futuros. Ao estar hábil a fornecer essas previsões, o agente exibiria evidências de algum grau de compreensão das atividades descritas no ambiente.

A estrutura de um ambiente de *software* que comporta um modelo simples em que a memória de um agente observador é perturbada por eventos do mundo e evolui abstrações relacionadas a esses eventos é descrita neste capítulo. O curso de uma simulação, a criação e a modificação de estruturas de memória e a obtenção de previsões são abordadas com o auxílio de exemplos. Inicialmente, no entanto, é necessário familiarizar-se com a topologia global dos elementos constituintes do ambiente e seus papéis em uma simulação.

4.2 Macrovisão dos Componentes e o Modelo de Interação

O sistema opera como um reprodutor de simulações descritas em arquivos manualmente construídos. Em toda simulação, o objetivo principal é que a memória de um agente que observa o mundo seja propriamente perturbada a fim de que ele possa eventualmente, após suficientes observações, prever com alguma eficiência novos estados do mundo. Esse agente é externo ao ambiente e não interage com o ambiente em momento algum, exceto pela recepção de informações sensoriais.

Uma simulação de sucesso seria aquela em que o agente *observador* obtivesse suficientes e adequadas observações do mundo e sua memória fosse modificada propriamente, de forma a poder prever estados futuros semelhantes àqueles por ele observados. O sucesso de uma simulação depende tanto dos tipos de modificações que podem ocorrer na memória do agente observador quanto da diligência com que as descrições da simulação foram projetadas — o compromisso com o treinamento antecede um eventual desafio.

Entre dois ciclos subseqüentes de uma simulação, o estado do mundo pode se alterar em decorrência de um *evento* previsto nas descrições da simulação. Eventos correspondem a alterações simples em alguma estrutura componente do estado do mundo. Todo evento deve respeitar um *padrão de evento* do qual o agente observador tenha conhecimento, do contrário nenhuma associação entre as diferenças encontradas nos dois estados subseqüentes do mundo poderá ser detectada. Encadeamentos de eventos correspondem

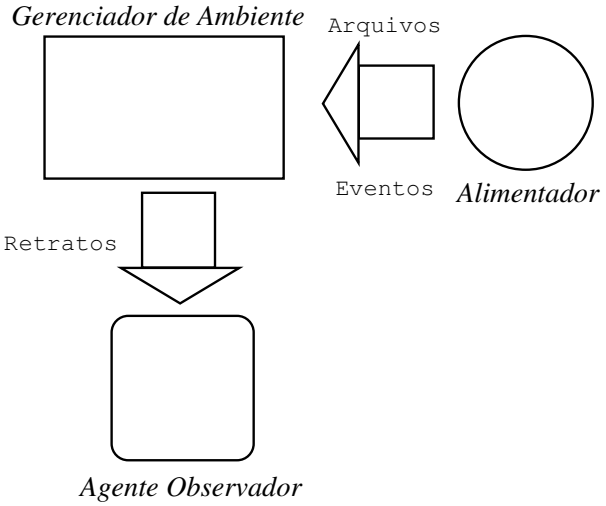

Figura 4.1: Fluxo de Comunicação entre os Módulos. O Alimentador e o Agente Observador possuem comunicação com o Gerenciador de Ambiente, mas não entre si.

a *seqüências*; a abstração de seqüências de eventos constitui fonte para previsões em possíveis acompanhamentos futuros de seqüências de eventos semelhantes.

4.2.1 Organização dos Módulos e Arquivos Pré-definidos

Três módulos integram o sistema, a saber: *Módulo Alimentador*, *Módulo Gerenciador de Ambiente* e *Módulo Observador*. Os módulos Alimentador e Observador comunicam-se com o módulo Gerenciador de Ambiente, mas não se comunicam entre si. O Gerenciador de Ambiente, no entanto, não é um mediador entre seus dois módulos adjacentes; sua função é realizar a simulação. Os três módulos e a forma como a comunicação flui entre eles estão dispostos na **Figura 4.1**. Cabe citar que o Gerenciador de Ambiente é o primeiro módulo a ser executado, uma vez que ele é um servidor que espera conexões de seus dois clientes — Alimentador e Observador.

Além dos módulos que constituem o sistema, para que uma simulação possa ocorrer são necessários também arquivos pré-definidos pelo projetista da simulação representando o seu "conteúdo", ou determinando o seu curso, e como as evoluções nela ocorrem, uma vez que os módulos apenas interpretam essas informações. Há quatro tipos de arquivos pré-definidos: *cenário*, *roteiro*, *transições de estado* e *padrões de eventos*; os quatro serão vistos em mais detalhes ao longo desta subseção.

4.2.1.1 O Módulo Alimentador

Alimentador é um módulo trivial e sua separação neste trabalho só é justificada para fins de organização modular de código. Sua função é a de proporcionar uma interface de interação com o sistema.

Estando o Gerenciador de Ambiente em execução, o Alimentador pode ser executado. Após estabelecer uma conexão com o Gerenciador, é dada ao operador da simulação a oportunidade de escolher quais arquivos serão usados para guiar e permitir a realização da simulação. Na atual versão, apenas um nome é informado; o Alimentador assume que os arquivos necessários para a simulação são identificados pelo nome informado e pela extensão padrão[1] definida para cada tipo de arquivo[2].

Uma vez selecionados os arquivos que definirão o conteúdo da simulação, o Alimentador pode enviar, a comando do operador, as informações de qual cenário e qual arquivo de transição de estado deverão ser carregados ao Gerenciador. Após isso, o Alimentador lê os eventos do roteiro selecionado e os envia, um por vez, ao Gerenciador de Ambiente.

O Arquivo Roteiro Um roteiro contém uma seqüência de estruturas denominadas *eventos* — a serem discutidos na **Subseção 4.4.2**. Um evento pode ou não acarretar uma alteração no estado do ambiente. Sendo assim, é o roteiro quem confere dinamismo a uma simulação e determina seus ciclos de execução, pois cada evento corresponde ao avanço de uma unidade de tempo na simulação.

4.2.1.2 O Módulo Gerenciador de Ambiente

Embora as evoluções de estado do mundo encontrem-se estaticamente pré-definidas em arquivos, elas acontecem de fato no contexto do Gerenciador de Ambiente. Todo estado $s_i \in S$ é produzido e mantido no Gerenciador de Ambiente e corresponde a um instante de tempo t_i. Inicialmente, após consultar os devidos arquivos contendo o cenário e as transições de estado, o Gerenciador envia ao Observador um retrato do estado inicial s_0 do ambiente no instante t_0.

Em seguida, um processo iterativo é iniciado no qual cada passo consiste em receber um evento do Alimentador, aplicar as devidas transições de estado sobre o estado s_i do

[1] Arquivos roteiros, cenários e de transições de estados são identificados, respectivamente, pelas extensões *scp*, *scn* e *trs* neste trabalho.

[2] Esta comodidade deverá ser removida em versões futuras para possibilitar que vários roteiros compartilhem um mesmo cenário inicial, por exemplo.

ambiente, com a simulação no instante de tempo t_i, a fim de produzir o estado s_{i+1}, o que incrementa o tempo de simulação para t_{i+1}. O Gerenciador então envia o estado s_{i+1} ao Observador para finalizar a execução de um ciclo da simulação[3].

Avanços no tempo virtual de uma simulação t não correspondem ao tempo real de execução. O tempo virtual t é discreto e assíncrono com o tempo real, porém sua evolução é síncrona em si mesma.

O Arquivo Cenário Este arquivo contém o estado inicial s_0 do ambiente. Mais detalhadamente, ele contém um conjunto de estruturas denominadas *estados*, cujo conceito será apresentado na **Subseção 4.4.1**, que serão mantidas pelo Gerenciador de Ambiente e por ele alteradas conforme necessário na ocorrência de eventos. Juntos, os estados representam um *retrato* da simulação.

O Arquivo Transições de Estado Estão descritos neste arquivo os procedimentos executados sobre as estruturas que compõem o estado de uma simulação na ocorrência de um evento e que efetuam as devidas alterações no estado do mundo. Esses procedimentos são aplicados pelo Gerenciador de Ambiente, que é o módulo que mantém as estruturas que representam o estado do ambiente. Os procedimentos de transição de estado referenciam primitivas padrão do Gerenciador para alteração de memória, além de executarem cálculos locais específicos definidos pelo projetista.

4.2.1.3 O Módulo Observador

Este módulo é um agente[4] que recebe "retratos" do estado do ambiente a cada ciclo de simulação e tenta relacioná-los a fim de possuir uma visão abstrata e em alto nível da organização do mundo. Nenhuma informação do arquivo de transições de estado é participada ao Observador; da mesma forma, nenhum evento tal como descrito em roteiro é enviado diretamente ao Observador. Esse agente trabalha apenas com as estruturas

[3]Este trabalho focou-se em proporcionar uma visão completa, livre de ruídos, do ambiente ao Observador. Embora irreal, é uma medida que facilita a realização de um estudo preliminar.

[4]Embora, como será visto adiante, o Observador não execute ações sobre o mundo, ele será tratado como agente por razões específicas deste trabalho: (i) ele percebe o mundo e suas percepções alteram sua memória simbólica; (ii) suas estruturas de memória e os padrões de modificação a que elas estão sujeitas denotam uma representação de nível mais alto do que uma simples base de dados de informações sensoriais ou uma base de conhecimento pequena baseada em regras simples; (iii) pretende-se, em trabalhos futuros de extensão, conferir a esse agente competências necessárias para atuar no mundo.

existentes nas informações de estado do mundo tal como recebidas do Gerenciador de Ambiente a cada ciclo da simulação.

O agente Observador assume papel central na simulação, uma vez que são as modificações em suas estruturas de memória o foco de estudo deste trabalho. À recepção de uma mensagem do Gerenciador contendo estruturas que representam um estado do ambiente, a memória do agente Observador inicia um processo de reconhecimento comparando-as estruturalmente com sua própria composição. Diferenças entre as estruturas de dois estados subseqüentes do mundo s_i e s_{i+1} podem render o reconhecimento de um evento caso haja um respectivo padrão conhecido.

O Arquivo Padrões de Eventos Este último arquivo, também modelado pelo projetista da simulação, é composto por padrões indicativos dos tipos de eventos passíveis de serem encontrados em uma simulação. O agente Observador utiliza esses padrões para reconhecer instâncias de eventos em uma simulação. Podem ser reconhecidos somente os eventos para os quais existem padrões. Um padrão é uma estrutura representando um tipo de evento — e não um evento em si — que pode ser usado para explicar as diferenças encontradas entre estruturas de estados subseqüentes s_i e s_{i+1}.

4.2.2 Método e Ciclo de Execução

Conforme exposto na **Seção 3.4**, ambientes de *software* e simulações têm sido amplamente explorados na área de Inteligência Artificial, especialmente em aplicações focadas em agentes, em virtude de razões como economia de recursos e menor conjunto de requisitos [Tambe et al., 1995, Grand and Cliff, 1998, Rickel and Johnson, 2000, Isaac and Sammut, 2003]. Embora seja verdade que em alguns domínios é necessário que simulações proporcionem um ambiente que reproduza com alta fidelidade as mais finas granularidades do ambiente real — porque sinais sensoriais sutis podem ser relevantes —, nem todo domínio exige tal exatidão [Tambe et al., 1995]. Em outras palavras, um ambiente de *software* poderia, em certas circunstâncias, executar um melhor trabalho se fornecesse informações sensoriais expressas em um formalismo elaborado, facilitando a tarefa dos agentes nele embutidos ao remover a necessidade de se identificar padrões relacionando sinais puros em uma representação de baixo nível.

Se o estudo focasse competências que não operam diretamente sobre o mundo real por meio de sinais puros, poderia ser mais adequado servir-se de informações mais refi-

nadas, escondendo-se dados sensoriais de baixo nível dos módulos responsáveis por essas competências alvo. Mesmo em ambientes que fornecem apenas dados de baixo nível aos dispositivos captadores de um agente, essas sensações poderiam nunca alcançar um módulo de competência de alto nível: um tratamento prévio sobre os dados simples poderia ser feito pelos próprios sensores, ou por dispositivos/módulos a eles acoplados — anteriores a módulos de decisão —, que por sua vez enviariam representações mais significativas a módulos de alto nível. Trabalhos recentes [Dzeroski et al., 1998, Kaelbling et al., 2001], discutidos na **Subseção 3.3.2**, adotaram essa abordagem no domínio mundo dos blocos.

Este trabalho faz uso de um ambiente de *software* — controlado pelo Gerenciador de Ambiente — que dispõe de algumas semelhanças com o domínio mundo dos blocos. A tarefa de aprendizado está centralizada em um agente — Observador — que observa o comportamento de agentes virtuais enquanto eles executam ações a fim de alcançar objetivos triviais instanciados por algum ente externo. Assume-se que os agentes virtuais são competentes para projetar uma seqüência de ações requerida para realizar algum objetivo; o Observador, em contraste, pode reconhecer apenas um evento isolado *a priori* — baseado em um conjunto de padrões de eventos. Toda informação recebida do Gerenciador de Ambiente pelo Observador respeita uma forma de representação de alto nível. Em nenhum momento representações de baixo nível são manipuladas em quaisquer dos módulos integrantes do sistema.

Conforme pode ser visualizado na **Figura 4.2** e segmentado na **Subseção 4.2.1**, um ciclo de simulação consiste no envio de um evento de um roteiro pelo Alimentador ao Gerenciador de Ambiente, o qual aplica as devidas funções de transição de estado sobre s_i em t_i, produzindo s_{i+1} em t_{i+1} e enviando s_{i+1} ao Observador, o qual tem sua memória perturbada e, eventualmente, reconhece a ocorrência de um evento. O estado inicial s_0, em t_0, é obtido do arquivo Cenário e corresponde ao primeiro retrato do mundo enviado ao Observador, consistindo no início da simulação do ponto de vista desse agente.

O objetivo principal do Observador é evoluir um conjunto de abstrações que permitam a ele prever o comportamento dos agentes virtuais quando para eles forem instanciados objetivos. A instanciação de um objetivo corresponde a uma seqüência de eventos representando termos de uma linguagem simples e limitada. Nessa seqüência, são identificados o agente virtual que deve realizar uma ação e a tarefa que ele deve realizar. Nenhum conhecimento sobre a linguagem — exceto padrões de eventos — é detido pelo Observador *a priori*. O agente Observador detém um módulo de conhecimento que possui semelhanças com uma memória dinâmica [Schank, 1982], porém bastante primitiva e simplificada.

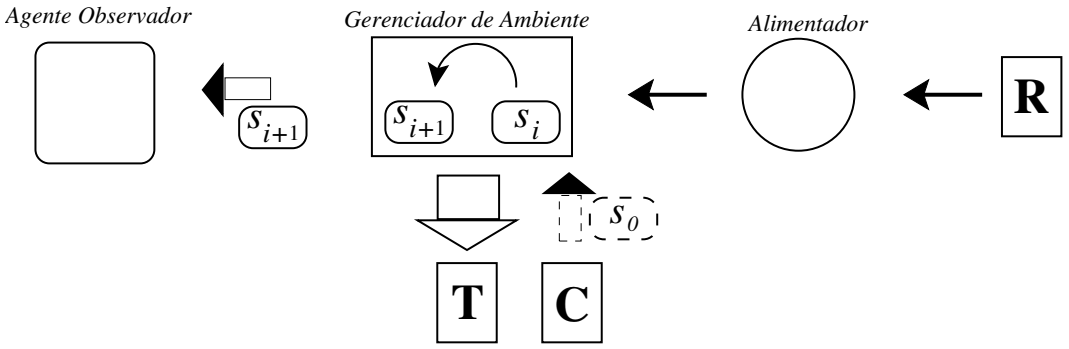

Figura 4.2: Ciclo de Execução. A interação entre os módulos do sistema e os arquivos de usuário durante um ciclo de uma simulação.

Com objetivos simples, um conjunto limitado de ações e seus padrões de eventos correspondentes, e uma simulação adequada, o Observador pode aprender as formas pelas quais agentes virtuais reagem a eventos do mundo, ou instanciações de objetivos, e assim prever estados futuros do mundo ao ser lembrado de circunstâncias semelhantes do passado. Esse método lembra a abordagem utilizada nos Cenários Baseados em Objetivo (*Goal-based Cenarios*) [Schank, 1992], ainda que focado em um agente que carece de participação ativa no mundo, conforme visto na **Subseção 4.2.1**[5]. A ausência de participação ativa, entretanto, não impede o estudo de como estruturas abstratas poderiam vir a se desenvolver na memória do agente Observador e conferir a ele capacidade para efetuar previsões sobre evoluções futuras no ambiente.

Para que o Observador possa ser lembrado de circunstâncias do passado, sua memória é perturbada e estruturas são desenvolvidas a cada mensagem vinda do Gerenciador de Ambiente. As estruturas que compõem o estado do mundo representado em uma mensagem são hierarquicamente comparadas no sentido *top-down* com as estruturas existentes em memória. Falhas nesse processo rendem a necessidade de adaptação da memória para que futuras ocorrências de percepções semelhantes possam ser reconhecidas. Ao estar hábil a acompanhar as relações entre dois retratos subseqüentes do mundo, a memória fornece previsões sobre estados e eventos futuros em determinadas janelas de tempo ao longo do curso de execução de uma simulação.

[5]Para atuar no mundo, entre outros, seria requerido que o agente possuísse conhecimento prévio de quem ele é — ou a habilidade para reconhecer isso em tempo de simulação — e reconhecer que objetivos que lhe fossem instanciados somente seriam satisfeitos mediante sua participação ativa. Além disso, o agente necessitaria também de um módulo responsável por efetivamente produzir ações sempre que a necessidade de atuar fosse reconhecida.

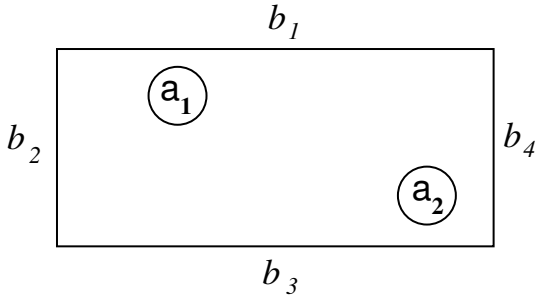

Figura 4.3: O Mundo. Topologia do ambiente de *software*.

4.3 A Organização Estrutural e a Representação no Ambiente de Software

O curso da simulação a ser acompanhada ao longo deste capítulo realiza-se em um mundo simples, sem aspecto de desdobramento espacial, bi-dimensional, limitado e desprovido de informações de localização[6]. Como pode ser observado na **Figura 4.3**, o mundo é delimitado por quatro bordas $B = \{\mathtt{b}_1,\ \mathtt{b}_2,\ \mathtt{b}_3,\ \mathtt{b}_4\}$ que são utilizadas para representar a orientação de um agente virtual.

Neste trabalho, os agentes virtuais $a_k \in A$ foram representados como estruturas atômicas que executam uma ação simples — `vire_direita`, por exemplo. Um retrato do mundo é representado como um conjunto de estruturas denominadas *estados*[7]. Um tipo de estado utilizado na simulação que servirá de exemplo ao longo deste capítulo representa o fato de um agente virtual estar voltado para — cujo predicado utilizado para representação é `voltado_para` — alguma borda b_j do mundo.

Cada estrutura *estado* presente em um retrato do mundo é analisada separadamente na memória do Observador e sua acomodação dará origem a pelo menos uma instância representando-a. Estados podem ser aninhados, porém cada estrutura atômica ou composta acarretará a formação de uma estrutura de memória correspondente, não importando quão interna ou externa ela seja.

[6]Pretende-se estender o trabalho para futuramente suportar localização em um mundo discreto por meio de coordenadas respeitando unidades não-fracionadas.

[7]Deve-se atentar para o fato de que ao longo deste trabalho o termo *estado* pode ser associado a diferentes contextos: (i) um retrato do mundo, (ii) o estado da memória do agente Observador — as estruturas em memória —, (iii) uma única estrutura componente de um retrato do mundo ou (iv) uma única unidade estrutural em memória.

4.4 A Memória do Agente Observador Sujeita a uma Simulação

Inicialmente, o agente Observador dispõe de uma memória cujas únicas estruturas representam raízes para cada tipo de estrutura que a partir delas podem vir a se formar, portanto sem qualquer conteúdo histórico. Cada estado proveniente de um retrato do mundo é potencialmente uma fonte de perturbação que precisará ser acomodada. Tipicamente, as primeiras estruturas evoluídas em memória corresponderão a estados.

4.4.1 Evolução de Estados

A fim de melhor compreender como estados são evoluídos em memória, distinguir-se-á primeiramente seus componentes estruturais. Em seguida, serão introduzidos os processos de reconhecimento e abstração de estados.

4.4.1.1 Composição de Estados

Um estado, enquanto unidade estrutural, assume representações diferentes dependendo do contexto onde está localizado, seja esse contexto um retrato do mundo ou a memória do agente Observador. Ora, um retrato do mundo limita-se a um conjunto de estados; por conseguinte, essa é a única estrutura encontrada em um retrato do mundo. Um estado que compõe um retrato do mundo possui o formato especificado em (4.1).

$$\texttt{state(}\mathit{Substance}\texttt{, [slot(}\mathit{Role}_1\texttt{, }\mathit{Filler}_1\texttt{), ..., slot(}\mathit{Role}_n\texttt{, }\mathit{Filler}_n\texttt{)]).} \tag{4.1}$$

Por outro lado, uma estrutura do tipo estado que é integrante da memória do agente Observador é dotada de componentes adicionais. Esse acréscimo de complexidade é devido às relações mantidas com outras estruturas integrantes da memória, as quais precisam constar na sua representação. Ao ser reconhecida pela memória do agente Observador, uma estrutura do tipo estado que integra um retrato do mundo dá origem a uma estrutura denominada *estado observado*[8], cujo formato é explicitado em (4.2).

[8]Este trabalho prefere o termo *observado* a instância pelo fato de que as abstrações são geradas a partir daquilo que é percebido no mundo, portanto as observações são anteriores às abstrações. O fato de o termo instância representar um objeto construído a partir de uma descrição de classe segundo o paradigma de Orientação a Objetos e até algumas formas de representação de Conhecimento em Inteligência Artificial — sugerindo, numa interpretação rasa mas amplamente difundida, que a abstração seria a concepção

$$\begin{aligned}&\texttt{sensed\_state(}ID\texttt{,}\\ &\qquad\qquad Abstraction\texttt{,}\\ &\qquad\qquad Substance\texttt{,}\\ &\qquad\qquad \texttt{[slot(}Role_1\texttt{, }Filler_1\texttt{), ..., slot(}Role_n\texttt{, }Filler_n\texttt{)],}\\ &\qquad\qquad Linkage\texttt{,}\\ &\qquad\qquad Timestamp\texttt{).}\end{aligned} \tag{4.2}$$

Para evidenciar as diferenças entre as duas representações, ao passo que examinando suas minúcias, a **Tabela 4.1** é apresentada. Pode-se notar alguma semelhança das representações de estado adotadas neste trabalho com formas de representação clássicas em Inteligência Artificial, mais especificamente Dependência Conceitual e Frames [Schank, 1972, Minsky, 1975] — como visto na **Subseção 2.4.1**, é creditada ao formalismo de Dependência Conceitual a idéia original de *slot-filling*. Os *slots* permitem que estados componham outros de forma aninhada[9]. Trabalhos recentes realizados nesta instituição utilizaram também representações semelhantes, principalmente pela utilização do conceito de estado para denotar um predicado [Soares, 2001, dos Santos et al., 2004].

A noção de estado captura com fidelidade a informação expressa por um elemento constituinte de um retrato do mundo: um estado relaciona elementos de forma pontual, evidenciando suas relações associadas a um determinado instante de tempo. Dois estados subseqüentes estruturalmente semelhantes podem ser reconhecidos como uma constância efêmera ou ainda encorajar a presunção de um elemento estável. Estados oferecem ainda a oportunidade para o reconhecimento de eventos — na **Subseção 4.4.2**, será visto como um conjunto formado por dois estados destoantes, pertencentes a dois retratos do mundo consecutivos, podem dar origem à evolução de um evento e representar uma transição de estado[10]. Estados propiciam uma fundação para as demais estruturas a serem utilizadas neste trabalho e balizam o processo de reconhecimento que se realiza na memória do agente Observador, pois são sempre as primeiras estruturas reconhecidas em cada ciclo[11].

original — poderia induzir a uma associação do conceito abstração de estado com o conceito classe do paradigma de Orientação a Objetos e causar ambigüidades.

[9] É importante ressaltar que não necessariamente um estado precisa dispor de *slots* — ele pode ter uma lista de *slots* vazia.

[10] Neste caso, a transição de estado limita-se a essas estruturas internas e não deve ser confundida com um novo retrato do mundo, embora as transições de estado internas transcendam um retrato do mundo.

[11] Para Kaelbling et al. [2001], "é difícil imaginar um agente verdadeiramente inteligente que não concebe o mundo em termos de objetos e suas propriedades e relações com outros objetos". Este trabalho não objetiva opor-se à visão de Kaelbling *et al.*, porém estados parecem capturar melhor a informação que se deseja representar a fim de satisfazer os objetivos deste trabalho. Embora não haja a intenção de se postular novas estruturas ou formas de representação neste trabalho, existe de fato uma preferência para que as estruturas utilizadas convirjam para, ou ao menos guardem razoável semelhança com, estruturas

Componente	Contexto	
	Retrato do Ambiente	Memória do Agente
`state/sensed_state`	Predicado constante que identifica uma estrutura do tipo estado.	Predicado constante que identifica uma estrutura do tipo estado observado.
`slot`	Constante que identifica um elemento agregado que compõe um estado.	Constante que identifica um elemento agregado que compõe um estado.
ID		Um identificador global da estrutura. Seu valor é único dentro do universo das estruturas em memória.
Abstraction		Mantém o ID da estrutura que é uma abstração imediata para a estrutura em questão.
Substance	Termo que dá nome ao estado — tal como um predicado.	Termo que dá nome ao estado — tal como um predicado.
Linkage		Indica se ocorreu uma nova observação desta estrutura no instante t_{i+1}. Assume valores no domínio {`yes`, `no`}.
Role	Termo que denota o papel exercido pela substrutura conteúdo do agregado no estado que a contém.	Termo que denota o papel exercido pela substrutura conteúdo do agregado no estado que a contém.
Filler	Mantém a substrutura que é o conteúdo do agregado.	Mantém o ID da substrutura que é o conteúdo do agregado.
Timestamp		Mantém a estampilha de tempo correspondente ao instante t_i em que o estado manifestou-se no ambiente.

Tabela 4.1: Estados, Componentes e Contextos. Confrontação das representações da estrutura estado segundo seu contexto.

4.4.1.2 Reconhecimento e Abstração de Estados

Para requisitar a realização de uma simulação, um usuário utiliza o módulo Alimentador para informar quais arquivos serão utilizados como Cenário, Transições de Estado e Roteiro[12]. Uma simulação inicia-se com a intervenção de um usuário na interface propiciada pelo módulo Alimentador. Do ponto de vista do Gerenciador de Ambiente, o início da simulação consiste em carregar o arquivo Cenário, ajustar o seu conteúdo como estado inicial s_0 da simulação e enviar esse estado ao Observador. Por sua vez, o Observador efetua o processo de reconhecimento das estruturas contidas nessa primeira mensagem. No exemplo a ser descrito nesta seção, o arquivo Cenário está explicitado em (**4.3**).

```
state(voltado_para,
      [slot(agente, state(a1, [])), slot(borda, state(b2, []))]).
state(voltado_para,
      [slot(agente, state(a2, [])), slot(borda, state(b2, []))]).
```

(4.3)

Ao chegar ao agente Observador, a mensagem consiste de uma lista de estados — no caso da primeira mensagem, os estados são exatamente os representados em (**4.3**). Cada um dos membros dessa lista é analisado separadamente e dará origem a ao menos uma instância em memória representando-o. Na verdade, uma vez que estados podem estar aninhados, cada um dos estados componentes de uma estrutura composta dará origem a sua própria instância em memória.

O resultado do reconhecimento da primeira estrutura integrante do estado s_0 corresponde à evolução, na memória do agente Observador, das estruturas apresentadas pela **Figura 4.4**. Como não havia nenhuma outra estrutura já evoluída em memória, uma

semelhantes às previstas por Schank and Abelson [1977], Schank [1982] e Minsky [1975]. Ainda que estados não tenham sido privilegiados como capitais nesses trabalhos, os autores reconhecem a necessidade de sua representação, como visto na **Subseção 2.4.5**. Um motivo pelo qual a representação de estados não foi mais abordada por esses autores é que eles buscavam representações de níveis mais altos: Schank, por exemplo, tinha como objetivo principal a compreensão de linguagem natural; já Minsky faz uso de situações complexas para exemplificar a forma de representação em *frames*. Estados, por outro lado, são estruturas mais básicas, o que corrobora a sua utilização neste trabalho pois o domínio de aplicação e os objetivos do trabalho referenciam conceitos simples. Embora a abordagem possa parecer errante quando comparada com os trabalhos de Schank e Minsky, os trabalhos recentes de Kaelbling et al. [2001] e Dzeroski et al. [1998] dão suporte ao estudo da representação e do aprendizado em mundos elementares, desprovidos de ornamentos.

[12] O arquivo Padrões de Eventos não está sendo informado pelo Alimentador na atual implementação; o Observador toma conhecimento desse arquivo por intermédio de uma configuração local.

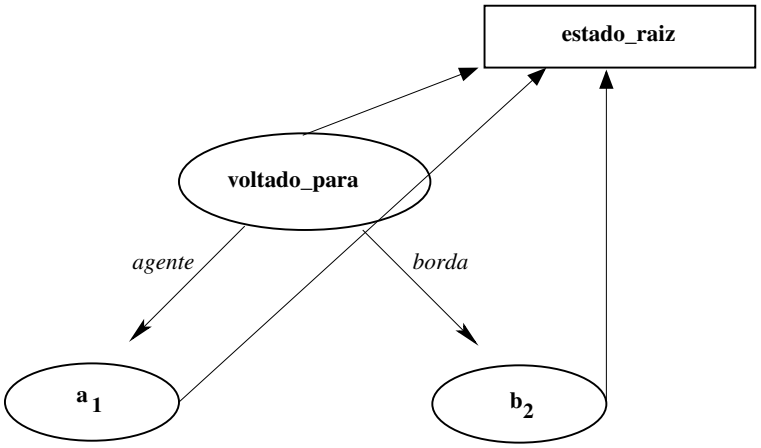

Figura 4.4: Reconhecimento da Primeira Estrutura. Todas as instâncias são anexadas à estrutura raiz.

vez que a memória encontra-se vazia no início do processo, três novas estruturas são geradas: duas delas representam estruturas internas (`a1` e `b2`) e uma representa o estado composto (`voltado_para`). As estruturas mais internas são reconhecidas primeiro porque o reconhecimento da estrutura mais externa requer que elas já estejam reconhecidas, a fim de dispor das referências necessárias para preencher propriamente seus *slots*. Todas essas estruturas são, *ab initio*, anexadas diretamente a uma estrutura estado raiz.

O processo de reconhecimento, até aqui, consistiu simplesmente de um mapeamento de um formato de estrutura estado — utilizado pelo Gerenciador de Ambiente no envio de mensagens ao agente Observador — para outro — correspondente a uma instância na memória do agente Observador. Esse mapeamento está de acordo com o apresentado na **Tabela 4.1**. Aliás, cabe lembrar que, tal como enunciado nessa tabela, há informações adicionais em cada estrutura que não foram contempladas na **Figura 4.4** — embora as estruturas evoluídas estejam sendo referenciadas pelo componente *Substance*[13], o que determina suas identidades é o componente *ID*.

No instante em que se inicia o processo de reconhecimento da segunda estrutura presente no primeiro retrato do mundo s_0, já há informação em memória com potencial para influenciar esse processo; com isso, o resultado final do processo de reconhecimento do primeiro retrato do mundo s_0 é o apresentado na **Figura 4.5**. Pode-se observar que a instância originária do reconhecimento da primeira estrutura presente no primeiro estado

[13]Componentes estão definidos na **Tabela 4.1**.

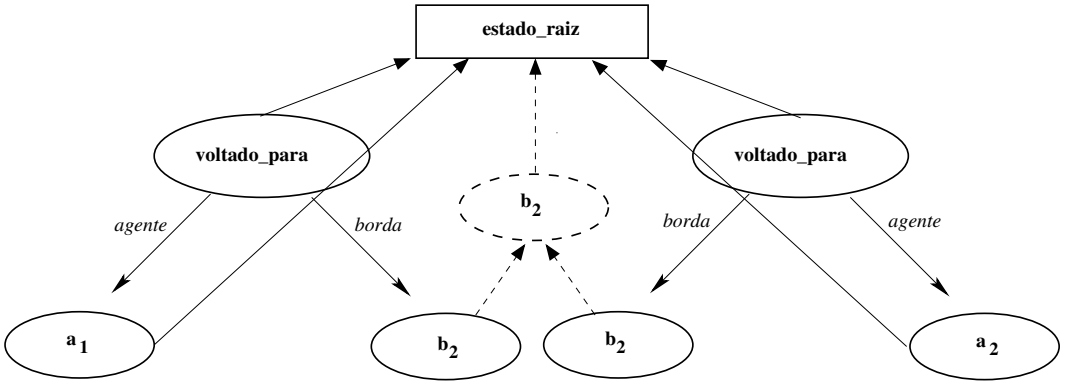

Figura 4.5: Reconhecimento do Primeiro Retrato. Evolução de uma abstração de estado.

do mundo e cujo componente *Substance* é b_2 não está mais diretamente anexada ao estado raiz tal como na **Figura 4.4**. Durante o reconhecimento da segunda estrutura presente em s_0, exatamente após a criação da instância cujo componente *Substance* é b_2, a memória é lembrada da existência prévia de outra instância cujo componente *Substance* também é b_2; como ambas as instâncias são atômicas — não-aninhadas —, abstrair a similaridade entre as duas é uma tarefa trivial. Como resultado, tem-se a evolução de uma estrutura abstrata representando essa semelhança; nesse caso, a semelhança restringe-se ao componente *Substance* e, por conseqüência, a abstração também é uma estrutura atômica.

Embora a evolução dessa abstração tenha sido trivial, ela foi baseada em um processo de reconhecimento complexo. Cada reconhecimento de uma estrutura estado particular, externa ou interna, inicia-se a partir da estrutura estado raiz; a seqüência desse processo consiste em recuperar dentre os nós imediatamente abaixo da estrutura raiz aqueles que possuem alguma similaridade estrutural com a nova estrutura que se deseja reconhecer. No caso simples, em que os nós a serem comparados não possuem *slots*, considera-se apenas o componente *Substance*; ainda nesse caso, pode haver no máximo um nó com o mesmo componente *Substance*: ou uma instância, ou uma abstração agrupando um conjunto de instâncias com o mesmo componente *Substance*[14].

O sentido do reconhecimento é sempre *top-down*. Assim, as abstrações exercem a função de índices, uma vez que abrigam conjuntos de estruturas semelhantes e reduzem a quantidade de comparações necessárias em cada passo do processo. Além disso, as abstrações de estado também funcionarão como base para a evolução de abstrações mais

[14]Cabe lembrar que cada ocorrência de uma estrutura no mundo dá origem a uma instância em memória.

complexas a serem vistas adiante. Em suma, começa a se delinear um sistema de memória no qual abstrações atuam como pilares de organização e instâncias simbolizam registros de observações do mundo — o referencial histórico da uma simulação.

Ao contrário do analisador DMAP [Riesbeck and Martin, 1985, Martin, 1993], visto na **Subseção 2.2.5**, no qual os índices são constituídos de elementos de linguagem natural, possuem estrutura própria e são anexados aos conceitos em memória, a abordagem introduzida neste trabalho modela a memória de forma que a sua própria estrutura forneça bases para um processo de indexação orientado a similaridade estrutural. Outra diferença basilar entre as duas abordagens é que enquanto DMAP realiza um processo de reconhecimento estritamente *bottom-up*, uma vez que no início do processo todos os índices em memória estão disponíveis para o reconhecimento de uma nova estrutura e, ao longo do processo de reconhecimento, alguns deles vão sendo descartados, a abordagem introduzida neste trabalho considera apenas as estruturas imediatamente abaixo do nível corrente como possíveis caminhos a serem seguidos. No entanto, essas diferenças categóricas podem ser explicadas pelo fato de que os domínios de trabalho são terminantemente diferentes em cada abordagem: DMAP objetiva o reconhecimento de textos em linguagem natural, por isso suas entradas são *tokens* em linguagem natural; este trabalho recebe como entrada um tipo de informação de mais alto nível, razoavelmente estruturada, e foca-se em estudar um conjunto pequeno e simples de exemplos.

Para clarificar ainda mais como todas essas estruturas são representadas de fato em memória, e a fim de elucidar as diferenças entre instâncias diferentes com componentes iguais, por exemplo, a exata representação das estruturas correspondentes às apresentadas na **Figura 4.5** está apresentada em (**4.4**). De posse das informações da **Tabela 4.1** e do formato descrito em (**4.2**), e analisando-se as estruturas apresentadas em (**4.4**), verifica-se que os estados cujos componentes *ID* são respectivamente 11 e 14 estão anexados à estrutura cujo *ID* é 15, enquanto que todas as demais estão anexadas à estrutura cujo *ID* é 1 e corresponde ao estado raiz. Ainda nesse exemplo, é possível verificar como é feita a referência entre estados compostos e estados aninhados: o *filler* de um *slot* em um estado composto é tão somente o *ID* do estado aninhado[15]. O elemento novo nesse exemplo é a estrutura de *ID* igual a 14 representando a abstração evoluída: além de seu predicado ser diferente, `abstract_state`, ela também possui um componente adicional na terceira posição; esse componente mantém referências para as estruturas sob ela agrupadas, ou

[15]Estruturas internas não possuem qualquer referência para a estrutura mais externa que as compõe.

seja, é uma referência da abstração para as suas especializações. Estruturas abstratas não dispõem de componente *Linkage*, pois uma vez criadas não há como desaparecerem como fruto de uma observação que deixou de referenciar instâncias para elas.

```
sensed_state(10, 1, a1, [], no, 1).
sensed_state(11, 15, b2, [], no, 1).
sensed_state(12, 1, voltado_para,
             [slot(agente, 10), slot(borda, 11)], no, 1).
sensed_state(13, 1, a2, [], no, 1).
sensed_state(14, 15, b2, [], no, 1).
abstract_state(15, 1, [14, 11], b2, [], 1).
sensed_state(16, 1, voltado_para,
             [slot(agente, 13), slot(borda, 14)], no, 1).
```
(4.4)

Apesar de haver duas estruturas de componente *Substance* igual a `voltado_para` em (**4.4**), até esse momento não havia sido concebida uma abstração para agrupá-las. A implementação discutida neste trabalho considera que para que uma abstração de dois estados que possuem *slots* seja evoluída, os *roles* desses estados devem coincidir e seus *fillers* precisam apontar para estruturas que sejam cobertas por abstrações comuns, as quais possam ser referenciadas na abstração que se deseja criar. Como não há nenhuma abstração agrupando as estruturas de *ID*s 10 e 13, tal critério não se satisfaz. Abstrações parciais de estados não são de interesse deste trabalho porque, entre outras razões, permitiriam a existência de hierarquias não-monotônicas de estados; uma vez que não há razão para isso mediante os objetivos do trabalho, essa concepção — e seus problemas — não é implementada, tampouco discutida em mais detalhes.

Desde que não ocorresse nenhuma alteração no estado do mundo, retratos subseqüentes renderiam, no máximo, a evolução de abstrações para `a1`, `a2` e `voltado_para`, além das instâncias indicando as estruturas observadas em cada ciclo da simulação. Enquanto estruturas mais complexas não são evoluídas, a memória do agente Observador espera sempre encontrar uma unificação perfeita entre os retratos de dois ciclos subseqüentes da simulação. Falhas nesse processo originariam a necessidade de explicações que, neste trabalho, correspondem à presunção da ocorrência de um evento entre os respectivos ciclos.

4.4.2 Evolução de Eventos: Explicações de Falhas

Embora nos ciclos iniciais — e em determinadas janelas de tempo ao longo do ciclo de uma simulação — o agente Observador assuma que seja mais sensato prever a ausência

de alterações no mundo entre dois ciclos subseqüentes da simulação, isto é, $s_{i+1} = s_i$, seria incoerente se esse agente não estivesse preparado para comportar modificações no momento em que elas se manifestassem. Está implícito nesse requisito o fato de que essas modificações possuem algum grau de arbitrariedade, uma vez que é difícil prever o momento em que uma seqüência de modificações terá início.

Este trabalho utiliza informação provida pelo projetista da simulação, além de lógica interna no agente Observador, para permitir o reconhecimento de alterações no mundo por meio da suposição da ocorrência de eventos. O projetista da simulação contribui para o reconhecimento de eventos fornecendo o arquivo Padrões de Eventos, onde serão descritos modelos de estruturas representando os tipos de eventos que podem ser reconhecidos e quais são as condições necessárias para o efetivo reconhecimento de cada um deles.

Ao fim do reconhecimento dos estados provenientes de um retrato s_i do mundo, dá-se início a um processo de busca por estruturas originárias do retrato s_{i-1} que correspondam às estruturas originadas do reconhecimento de s_i. Dessa forma, para cada nova instância gerada a partir de s_i, efetua-se uma busca em memória por uma instância cujo componente *Timestamp* detém o valor $i-1$ e com a qual a primeira possa ser estruturalmente unificada.

Quando não se verifica a existência prévia de uma estrutura originária de um retrato s_i — não há estrutura unificável com *Timestamp* igual a $i-1$ —, a memória inicia um processo de reconhecimento de um padrão de evento que possa ser usado para justificar tal anomalia. Todo padrão de evento consiste de (i) uma identificação para a sentença, (ii) uma identificação para o tipo de evento, (iii) um padrão de estado a ser encontrado em s_{i-1} — `<Before>` —, (iv) um padrão de estado a ser encontrado em s_i — `<After>` — e (v) um conjunto de restrições baseadas nos respectivos padrões de estado que devem ser satisfeitas a fim de que o evento possa ser reconhecido.

O objetivo é de que um tipo de evento possa ser descrito por tipos de estado presentes em dois ciclos subseqüentes da simulação — a propósito, os dois últimos ciclos progredidos — e por relações adicionais não contempladas na representação do mundo[16]. Cada par de estruturas de s_{i-1} e s_i que não se unifica deve casar respectivamente com os padrões de estado `<Before>` e `<After>` constituintes de um padrão de evento, a fim de que esse padrão de evento possa ser usado para explicar a disparidade entre a estrutura em s_i e sua contraparte em s_{i-1}. O formato de um padrão de evento é disposto em (**4.5**).

[16] Ao se permitir que relações extra-mundo possam ser verificadas, abre-se a possibilidade de se trabalhar com uma representação mais simples do mundo e tratar de complexidades adicionais apenas nos momentos necessários, externamente ao núcleo da memória.

```
event_pattern(ID,
              Event_Type,
              Before,
              After,
              Constraints).
```
(4.5)

O conteúdo do arquivo Padrões de Eventos utilizado na simulação referenciada nesta seção está disposto em (4.6) e (4.7). De imediato, é possível notar em (4.6) que além de sentenças representando diretamente os padrões de eventos, há também sentenças estabelecendo relações entre padrões de estruturas do tipo estado; essas relações consistem em informações adicionais necessárias para verificar se as condições exigidas em cada um dos padrões de evento são satisfeitas.

```
a_esquerda(state(b1, []), state(b2, [])).
a_esquerda(state(b2, []), state(b3, [])).
a_esquerda(state(b3, []), state(b4, [])).
a_esquerda(state(b4, []), state(b1, [])).
a_direita(state(b2, []), state(b1, [])).
a_direita(state(b3, []), state(b2, [])).
a_direita(state(b4, []), state(b3, [])).
a_direita(state(b1, []), state(b4, [])).
```
(4.6)

Tomando-se como exemplo o padrão de evento de tipo `vire_esquerda`, definido em (4.7), nota-se que ele é reconhecido quando os estados do mundo que causaram a falha no processo de unificação padrão possuem componente *Substance* igual a `voltado_para` e *slots* de *roles* iguais a `agente` e `borda`, como especificado nos componentes *Before* e *After* do padrão de evento. Constitui-se ainda condição necessária que os *fillers* dos *slots* de *role* igual a `borda` satisfaçam a relação `a_esquerda`, requerida no componente *Constraints*.

```
event_pattern(1, vire_esquerda,
              state(voltado_para, [slot(agente, A), slot(borda, W1)]),
              state(voltado_para, [slot(agente, A), slot(borda, W2)]),
              [a_esquerda(W1, W2)]).
event_pattern(2, vire_direita,
              state(voltado_para, [slot(agente, A), slot(borda, W1)]),
              state(voltado_para, [slot(agente, A), slot(borda, W2)]),
              [a_direita(W1, W2)]).
event_pattern(3, mensagem, nil, state(ecoando, [slot(som, _)]), []).
```
(4.7)

Para exemplificar a evolução de um evento, um fragmento do arquivo Roteiro utilizado na simulação descrita nesta seção é apresentado em (4.8). Conforme já introduzido na

Subseção 4.2.1, esse arquivo é composto de uma série de eventos; no caso apresentado em (4.8), os três primeiros — `ecoe` — representam a emissão de sons. Dentro do projeto da simulação, esses sons correspondem à instanciação de um objetivo para algum agente virtual. Nesse exemplo, o primeiro som referencia um agente que atende por `agente_x` e o instrui a executar uma ação, a qual consiste em um movimento — `vire` — em uma determinada direção — `a_esquerda`. O último evento não promove alteração alguma em memória e corresponde apenas ao avanço simples de um ciclo na simulação[17]. É importante lembrar que além de não possuírem significado algum para o agente Observador, os eventos tais como descritos no arquivo Roteiro nunca chegam ao Observador, pois são processados ainda no Gerenciador de Ambiente; e é somente dessa forma que novos ciclos da simulação são produzidos, pois eventos do Roteiro determinam ciclos em uma simulação.

```
ecoe([slot(objeto, sound(agente_x))]).
ecoe([slot(objeto, sound(vire))]).
ecoe([slot(objeto, sound(a_esquerda))]).
vire_esquerda([slot(ator, state(a1, []))]).
avance_ciclo([]).
```
(4.8)

Como também já visto na **Subseção 4.2.1**, quem produz novos estados da simulação a partir da aplicação de eventos — obtidos do módulo Alimentador, o qual executa a leitura do arquivo Roteiro — é o Gerenciador de Ambiente. Utilizando-se do arquivo Transições de Estado, ele produz um novo estado do mundo para cada evento do arquivo Roteiro, o que garante o avanço de um instante no tempo da simulação. Cada ciclo da simulação comporta no máximo um evento[18]. O arquivo Transições de Estado utilizado neste trabalho é visualizado em (4.9) e (4.10).

[17] Avanços simples podem ser importantes para melhor delimitar seqüências de eventos e facilitar os processos de reconhecimento e abstração.

[18] Estabelece-se aqui semelhanças entre os limites impostos na produção de novos retratos do mundo neste trabalho e na evolução de passos na resolução de um problema pela família de planejadores que adota a representação Strips [Fikes and Nilsson, 1971]: (i) apenas um único operador, na representação Strips, ou evento de roteiro, neste trabalho, pode ser aplicado em um ciclo para especificar em que consistirá uma transição de estado do mundo, o que impede que se confira concorrência à evolução do sistema; (ii) todas as conseqüências das modificações efetuadas por ações são explicitamente especificadas por meio de operações de adição e remoção; e (iii) um operador/evento consiste em uma ação instantânea, portanto uma ação não se estende no tempo e só é representada no resultado das transformações que ela produz. Essas afirmações constituem as chamadas *hipóteses* Strips [Waldinger, 1977]. Ferber and Müller [1996] propuseram um modelo alternativo baseado em influências e reações para uso em sistemas multiagentes, segundo o qual efeitos das ações de agentes são distinguidos de fenômenos do ambiente; esse modelo é didaticamente discutido e exemplificado em Ferber [1999]. Schank [1999], por sua vez, acredita que planos possam evoluir como estruturas mais sofisticadas em uma memória dinâmica e recuperados sempre que necessário, mas nunca construídos.

```
adjacente(state(b1, []), state(esquerda, []), state(b2, [])).
adjacente(state(b1, []), state(direita, []), state(b4, [])).
adjacente(state(b2, []), state(esquerda, []), state(b3, [])).
adjacente(state(b2, []), state(direita, []), state(b1, [])).
adjacente(state(b3, []), state(esquerda, []), state(b4, [])).
adjacente(state(b3, []), state(direita, []), state(b2, [])).
adjacente(state(b4, []), state(esquerda, []), state(b1, [])).
adjacente(state(b4, []), state(direita, []), state(b3, [])).
dimensao(state(horizontal, []), state(b2, []), state(b4, []), 6).
dimensao(state(vertical, []), state(b1, []), state(b3, []), 5).
direcao(Dimensao, Terminacao1, Terminacao2) :-
        dimensao(Dimensao, Terminacao1, Terminacao2, _).
direcao(Dimensao, Terminacao1, Terminacao2) :-
        dimensao(Dimensao, Terminacao2, Terminacao1, _).
```

(4.9)

A primeira parte do arquivo, exposta em (**4.9**), é constituída de informações adicionais ao processamento de cada tipo de transição de estado; já a segunda, exibida em (**4.10**), contém declarações específicas ao processamento de cada tipo de evento suportado. Basicamente, cada declaração consiste na remoção de uma estrutura específica do estado do mundo e na adição de uma nova, produzida a partir daquela removida.

```
avance_ciclo([]).
ecoe([slot(objeto, Som)]) :-
        registre_estado(state(ecoando, [slot(som, Som)])),
        agende_tarefa(remova_estado(state(ecoando, [slot(som, Som)]))).
vire_esquerda([slot(ator, Ator)]) :-
        remova_estado(state(voltado_para, [slot(agente, Ator),
                                           slot(borda, Origem)])),
        once(adjacente(Origem, state(esquerda, []), Terminacao),
        registre_estado(state(voltado_para, [slot(agente, Ator),
                                             slot(borda, Terminacao)]))).
vire_esquerda([slot(ator, Ator)]) :-
        remova_estado(state(voltado_para, [slot(agente, Ator),
                                           slot(borda, Origem)])),
        once(adjacente(Origem, state(direita, []), Terminacao)),
        registre_estado(state(voltado_para, [slot(agente, Ator),
                                             slot(borda, Terminacao)]))).
```

(4.10)

Uma novidade particular em (**4.10**) é o uso das primitivas `registre_estado`, `remova_estado` e `agende_tarefa`, as quais são implementadas no Gerenciador de Ambiente e constituem uma interface para o Projetista da Simulação requisitar alterações no

estado do mundo a partir do arquivo Transições de Estado. As duas primeiras primitivas são auto-explicativas; a terceira corresponde ao agendamento de uma das outras duas para que seja executada no próximo ciclo da simulação.

A cada um desses ciclos, o motor de inferência do Gerenciador de Ambiente simplesmente invoca um *goal*[19] que corresponde fielmente ao evento lido do arquivo Roteiro; esse *goal* é interpretado de acordo com as cláusulas consultadas do arquivo Transições de Estado. Interagindo com as primitivas de alteração de memória disponíveis pelo próprio módulo Gerenciador de Ambiente, a execução de um *goal* pode resultar em alterações no estado de uma simulação; mesmo que nenhuma alteração em memória ocorra, o avanço de um ciclo na simulação é sempre executado[20].

A execução da seqüência de eventos descrita em (4.8) resulta na evolução e permanência em memória das estruturas expostas na **Figura 4.6**[21]. A instância correspondendo ao evento `vire_esquerda` foi evoluída exatamente após o ciclo determinado pela execução do respectivo evento do arquivo Roteiro. Vale ressaltar que as estruturas empacotadas sob os componentes *Before* e *After* pertencem a retratos subseqüentes do mundo; portanto, um evento representa algo ocorrido no mundo entre esses dois retratos.

O formato de uma estrutura do tipo evento, evoluída em memória imediatamente após seu reconhecimento, é apresentado em (4.11). O componente novo, *Pattern*, promove acesso ao exato padrão de evento que foi base para a construção da estrutura em memória[22]. A estrutura total é empacotada pelo predicado `recognized_event`.

```
recognized_event(ID,
                 Abstraction,
                 Pattern,
                 Substance,
                 Before,
                 After,
                 Timestamp).
```
(4.11)

[19] O sistema todo foi implementado em Prolog. Um *goal* corresponde a uma chamada ou consulta à base de conhecimento em memória.

[20] Como a linguagem de implementação do sistema é Prolog, pode-se utilizar com o devido cuidado todo o aparato da linguagem na concepção do arquivo Transições de Estado.

[21] A **Figura 4.6** expõe apenas as estruturas que permaneceram em memória ao final da execução da seqüência de eventos em (4.8). Entretanto, ao longo da execução, estruturas representando sons também foram criadas, porém suas manifestações tiveram a duração de apenas um ciclo de simulação — um som é considerado, neste trabalho, uma manifestação pontual de duração definida e unitária. As estruturas evoluídas a partir de sons serão abordadas em mais detalhes na **Subseção 4.4.3**.

[22] Os demais componentes já são familiares em função de discussão prévia ao longo deste trabalho.

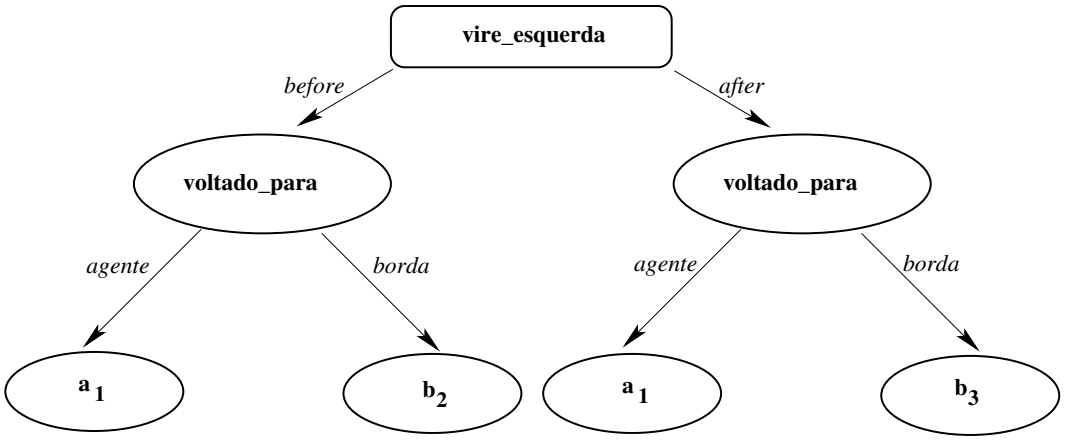

Figura 4.6: Evento `vire_esquerda`. Explicação para uma falha no processo de unificação padrão.

Um extrato da memória do agente Observador imediatamente após a execução dos eventos em (**4.8**) é apresentado em (**4.12**). Pode-se notar que o evento reconhecido promove interconexão entre dois estados com diferentes componentes *Timestamp*, isto é, estados que se manifestaram em diferentes (subseqüentes) ciclos da simulação. Para fins de informação, caso não houvesse ocorrido tal evento, o componente *Linkage* do estado cujo *ID* é igual a 42 deveria ser `yes`, ou seja, uma manifestação do mesmo estado no ciclo subseqüente teria sido observada.

```
sensed_state(40, 18, a1, [], no, 4).
sensed_state(41, 15, b2, [], no, 4).
sensed_state(42, 21, voltado_para,
             [slot(agente, 40), slot(borda, 41)], no, 4).
sensed_state(52, 18, a1, [], no, 5).
sensed_state(53, 1, b3, [], no, 5).
sensed_state(54, 1, voltado_para,
             [slot(agente, 52), slot(borda, 53)], no, 5).
recognized_event(55, 4, 1, vire_esquerda, 42, 54, 5).
```
(4.12)

A memória do agente Observador reconhece um evento do tipo `vire_esquerda` porque seu padrão de evento `vire_esquerda`, definido em (**4.7**), é o único que possui suas condições satisfeitas de acordo com os estados que causaram a falha de unificação. Cabe lembrar que esse conhecimento é provido pelo Projetista da Simulação. Por fim, o evento reconhecido está inicialmente anexado à estrutura evento raiz, cujo componente ID é 4.

É importante salientar que a evolução de um evento em memória se dá em dois passos

distintos: (i) a detecção da ocorrência de um evento, que consiste na percepção de uma falha no processo de unificação padrão entre dois retratos subseqüentes do mundo e em encontrar um padrão de evento que permita explicar essa falha e (ii) no próprio reconhecimento do evento, que consiste em estimular a memória a registrar um evento com as características detectadas. Esse último passo é análogo ao reconhecimento de um estado do mundo, guardadas as peculiaridades de cada processo.

Assim, reforça-se a idéia de reconhecimento *top-down*: ao se estimular a memória a ser lembrada de um evento, inicia-se um trabalho a partir da estrutura evento raiz. Em cada iteração nessa árvore, assim como no caso de estados, busca-se por um nó filho abstrato que comporte o novo evento e executa-se uma nova iteração a partir desse nó; caso um nó abstrato nessas características não exista, cria-se uma instância de evento no local corrente. Uma diferença fundamental entre os reconhecimentos de estados e de eventos é que um estado vem diretamente do Gerenciador de Ambiente, enquanto que um evento é detectado pela memória do agente Observador, obviamente mediante o auxílio do arquivo Padrões de Eventos. Ainda assim, pode-se dizer que o reconhecimento de eventos assemelha-se a um processo de retroalimentação, em que a evolução de instâncias de estados constitui-se a entrada para o reconhecimento de eventos.

No trabalho de Rickel and Johnson [2000], *Steve* recebe em cada ciclo tanto uma lista com o estado completo do mundo quanto uma lista dos eventos que ocorreram. Portanto, *Steve* tem acesso a informações sobre o mundo que não se restringem a uma mera representação de seu estado. No sistema descrito nesta dissertação, o agente Observador precisa identificar os eventos que porventura possam ter ocorrido; a memória precisa extrair o conteúdo semântico implícito relacionado à essas ocorrências no estado do mundo. Para isso, o arquivo Padrões de Eventos desempenha um papel primordial.

Assim como no reconhecimento de estados, o reconhecimento de eventos também pode dar origem à evolução de abstrações. Conforme já citado nesta mesma subseção, as abstrações guiam o processo de reconhecimento de um novo evento. Para entender como uma abstração de evento simples pode ser evoluída, uma abstração de estado composto — um estado com *slots* — serve de analogia: para que uma abstração de estado com dois *slots* possa ser evoluída, é necessário que já existam abstrações para os estados empacotados pelos *slots* das instâncias que serão agrupadas sob a nova abstração[23]. No caso de uma

[23]Nesse caso, as abstrações das instâncias empacotadas ainda precisam ser comuns para *slots* de mesmo *role*. Assim, quando se cria uma abstração de estado que possui um *slot* apontando para uma abstração, essa última abstração cobre todas as especializações empacotadas pelas instâncias da abstração que se

abstração de evento simples, a mesma estratégia pode ser empregada: uma abstração pode ser evoluída desde que existam abstrações para os estados empacotados sob os componentes *Before* e *After* e, logicamente, que ao menos dois eventos estruturalmente semelhantes já tenham sido reconhecidos pela memória[24].

Contudo, considerando-se particularmente eventos, dispõe-se de informação para a evolução de formas ainda mais sofisticadas de abstrações. Beneficiando-se novamente dos padrões de eventos, um tipo de abstração de evento baseado em restrições foi definido e agregou-se à memória o aparato necessário para evoluí-lo. O formato estrutural desse tipo mais sofisticado de abstração é apresentado em (**4.13**).

```
constraintbased_abstract_event(ID,
                               Abstraction,
                               Specialisations,
                               Pattern,
                               Substance,          (4.13)
                               Before,
                               After,
                               Constraints).
```

O predicado `constraintbased_abstract_event` tem a função de envolver a estrutura. Os componentes *ID*, *Abstraction*, *Pattern* e *Substance* são análogos às suas contrapartes na estrutura `recognized_event`. Para tornar mais clara a representação utilizada nos demais componentes, recorre-se à exemplificação encontrada em (**4.14**).

```
constraintbased_abstract_event(108,
                               4, [107, 55], 1, vire_esquerda,
                               state(voltado_para,
                                     [slot(agente, 18),
                                      slot(borda, constraint(1, A1))]),
                               state(voltado_para,
                                     [slot(agente, 18)),
                                      slot(borda, constraint(1, A2))]),
                               [constraint(1, a_esquerda(A1, A2))]).
```

(4.14)

O componente *Specialisations* contém as especializações agrupadas sob a estrutura. Diferentemente da instância de evento vista em (**4.12**), os componentes *Before* e *After*

está criando, respeitando os respectivos *roles*.

[24] Essa última regra é válida para qualquer estrutura em memória: só se evolui abstrações se há ao menos duas instâncias em memória a serem cobertas.

não mais são estados observados no mundo, tampouco estados abstratos; o conteúdo desses componentes tornou-se padrões respeitados pelos eventos cobertos pela abstração. Esses padrões são destinados a, obviamente, representarem restrições para os componentes *Before* e *After* das especializações. Portanto, cada uma das estruturas encontradas em um componente *Before* ou *After* é um padrão de estado e assemelha-se aos padrões de estado dos componentes *Before* e *After* dos padrões de eventos do arquivo informado pelo Projetista da Simulação. A diferença entre os componentes da abstração em questão e os componentes dos elementos do arquivo Padrões de Eventos é que os primeiros agregam mais pormenores e estão devidamente localizados na memória do agente Observador, além de terem sido evoluídos a partir de instâncias que já existiam. Já o componente *Constraints* é remetido à sua contraparte nos padrões de eventos, no entanto com um formato particular em razão do novo contexto.

Em cada um dos componentes, o primeiro *slot* tem *role* igual a `agente` e *filler* igual a 18; esse é o valor do componente *ID* de uma estrutura estado abstrata que agrupa os estados cujo componente *Substance* é `a1`, como será visto adiante. Assim, o *slot* determina que toda instância coberta pela abstração tenha um *slot* de *role* igual a `agente` e como *filler* alguma instância estado cujo componente *Substance* seja `a1`.

No caso do segundo *slot*, a situação é um pouco mais detalhada. Ainda que semelhante à restrição de variável livre encontrada nos padrões de eventos, ela é rotulada com o envolvente `constraint`, que possui um índice associando-o a uma restrição disposta na lista *Constraints*[25]. A lista contém exatamente a restrição encontrada no padrão de evento utilizado para gerar a abstração, porém envolvida pelo predicado `constraint`. Como já citado, a integração das restrições encontradas nos componentes *Before*, *After* e *Constraints*, além do componente *Substance*, determinarão se um novo evento reconhecido poderá ser classificado como uma instância dessa abstração. Eventos abstratos baseados em restrições desempenham um papel importante como fontes de informações para a evolução de estruturas mais sofisticadas, como será visto na **Subseção 4.4.3**.

As estruturas em memória que formaram a base para a evolução do evento abstrato baseado em restrições apresentado em (4.14) são as dispostas em (4.15). Embora trate-se de apenas um extrato da memória, contendo apenas as estruturas relevantes para o caso, está explícito o conceito de ordem e evolução gradual da memória; acompanhando-se a variação do componente *ID* em cada estrutura, é possível examinar em quais instantes

[25] O novo formato facilita a manipulação da informação para auxiliar o processo de evolução de estruturas mais sofisticadas.

a memória evolui cada estrutura e quais requisitos para essas evoluções são satisfeitos a cada um desses instantes. Como foi possível observar em (**4.14**), a evolução do evento abstrato ocorreu imediatamente após a evolução da última estrutura em (**4.15**), pois seus componentes *ID* são subseqüentes.

```
abstract_state(18, 1, [104, 94|...], a1, [], [], 10).
sensed_state(40, 18, a1, [], no, 4).
sensed_state(41, 15, b2, [], no, 4).
sensed_state(42, 21, voltado_para,
             [slot(agente, 40), slot(borda, 41)], no, 4).
sensed_state(52, 18, a1, [], no, 5).
sensed_state(53, 61, b3, [], no, 5).
sensed_state(54, 63, voltado_para,
             [slot(agente, 52), slot(borda, 53)], yes, 5).
recognized_event(55, 108, 1, vire_esquerda, 42, 54, 5).
sensed_state(92, 18, a1, [], no, 9).
sensed_state(93, 61, b3, [], no, 9).
sensed_state(94, 63, voltado_para,
             [slot(agente, 92), slot(borda, 93)], no, 9).
sensed_state(104, 18, a1, [], no, 10).
sensed_state(105, 1, b4, [], no, 10).
sensed_state(106, 1, voltado_para,
             [slot(agente, 104), slot(borda, 105)], no, 10).
recognized_event(107, 108, 1, vire_esquerda, 94, 106, 10).
```
(4.15)

Mais uma vez, é possível identificar o emprego de uma estratégia de reconhecimento *top-down*, ao passo que as estruturas se desenvolvem de forma *bottom-up*. Todo reconhecimento inicia-se sempre a partir de uma estrutura raiz. A criação de estruturas em memória, porém, acontece sempre nas folhas da árvore, para instâncias, ou logo após uma perturbação em um nível mais baixo, no caso de abstrações.

4.4.3 Evolução de Seqüências: Fontes de Previsão

Eventos observados podem vir a descrever um estereótipo; dois eventos imediatamente sucessivos podem revelar uma particularidade interessante. Dentro de uma progressão temporal tal como a simulação descrita nesta seção, uma relação entre dois eventos consecutivos pode ser a ordem de suas ocorrências. Estendendo-se essa relação de forma transitiva, obtém-se a concepção de cadeias de eventos ou *seqüências*. Neste trabalho, uma seqüência é uma sucessão limitada e contígua de eventos. Conforme Schank and Abelson [1977], o mundo real é um ambiente dinâmico onde situações são caracterizadas por even-

tos sucessivos ao longo do curso de uma cena. Claramente, eventos são pontuais, enquanto que seqüências representam uma série temporal ordenada de eventos.

No ambiente de *software* referenciado como *mundo* no presente trabalho, eventos podem ocorrer em seqüência. Enquanto consideradas isoladamente e como objetos únicos, seqüências não se revelam muito úteis. Entretanto, à medida em que se considera a possibilidade de que padrões de seqüências de eventos repitam-se com alguma regularidade, elas passam a ser importantes ferramentas para auxílio à compreensão do mundo; a informação mantida pelas seqüências estereotipadas torna-se fonte de previsões para um observador. O acompanhamento de seqüências configura-se como um arsenal que proporciona uma enumeração de prováveis estados e eventos futuros. Conforme as previsões vão se concretizando ou não, as seqüências podem continuar a serem consideradas ou não como fontes de previsões.

É difícil prever quando uma seqüência de modificações no estado do mundo terá início, entretanto é sensato prever que, uma vez iniciada, ela será completada ou ao menos será repetida até algum ponto[26]. Assim, o que determina a escolha de uma ou mais seqüências para servirem de fontes de previsão para os próximos estados do mundo é a ocorrência de um evento que se assemelhe suficientemente com o primeiro evento referenciado na(s) seqüência(s) em memória; se não há eventos no mundo, não há seqüências sendo consideradas como fontes de previsão.

À ocorrência de um evento qualquer, e na ausência de quaisquer previsões, pesquisa-se em memória por seqüências abstratas que comportem em sua primeira posição o evento referenciado; assim, as seqüências precisam fazer referência a abtrações de eventos e não a eventos observados particulares. Em caso positivo, as abstrações de seqüência que satisfizeram a condição são consideradas como fontes de previsão para os próximos estados do mundo e continuam assim enquanto forem se confirmando as respectivas previsões.

A competência instituída com as previsões caracterizar-se-ia como uma importante vantagem competitiva e teria um grau de utilidade bastante razoável para um agente que necessitasse sobreviver em um mundo dinâmico. Ao estar hábil a ser lembrado de estereótipos de seqüências observadas, um agente demonstraria uma evidência de que possui algum grau de compreensão do mundo e interpretação de suas transições. Em outras palavras, haveria justificativa para se utilizar uma memória dinâmica como fonte de conhecimento e componente de módulo cognitivo em um agente.

[26] O que de modo algum significa que as previsões oriundas dessa seqüência efetivamente serão verificadas; a nova seqüência pode perfeitamente não repetir eventos de nenhuma outra existente em memória.

A fim de exemplificar como se dá o processo de evolução de seqüências e seus estereótipos, serão revistos os eventos descritos em (**4.8**). Os três primeiros eventos do *script* metaforizam a execução de *sons*. Na memória do agente Observador, uma estrutura do tipo som é semelhante a uma estrutura do tipo estado observado atômica; estruturas do tipo som não possuem *slots*. O componente correspondente ao *Substance* dos estados observados em uma estrutura do tipo som é o *Token*, que nada mais é do que uma cadeia de texto simples que identifica o som emitido. O *Token* não possui nenhum significado prévio para o agente Observador, portanto termos da linguagem natural humana não são interpretados como tal; no entanto, seqüências permitem associar estruturas do tipo som a eventos que com elas se relacionam na linha do tempo.

Ao não possuir significado explícito para o agente Observador, um *token* pode conter literalmente qualquer cadeia de texto; somente com uma constante persistência de um determinado evento de som em preceder um mesmo evento de ação é que a memória do agente Observador pode adquirir um relacionamento que exprima uma possível associação entre os dois, sugerindo assim um significado provável implícito a um *token*.

Retornando à seqüência de instruções providas pelo projetista da simulação, o extrato de *script* apresentado em (**4.8**) corresponde à instanciação de um objetivo. Quem definira que esse objetivo deveria ser instanciado foi o projetista da simulação; o responsável pela execução desse objetivo, de acordo com a sua definição, era o agente virtual representado pelo estado observado atômico de *Substance* igual a `a1`[27]. Pelo fato de os *tokens* escolhidos para representar o objetivo terem sido análogos aos utilizados na linguagem natural humana, é simples inferir que o intuito consiste em determinar que um agente realize a operação de voltar-se para a sua esquerda, não importando o lado para o qual ele esteja voltado no momento de instanciação do objetivo.

A razão da decomposição da mensagem em mais de um *token* é a de se abrir a possibilidade para a evolução de abstrações mais úteis, que associem a informação expressa em cada *token* particular. Isso poderia permitir identificar a relação entre o *token* `vire` e a ação de executar um movimento desse tipo independentemente da direção à qual o movimento é requisitado. Para fins de situação no estado da arte, cabe lembrar que, como visto na **Subseção 3.3.2**, Grand and Cliff [1998] conferiram a seus agentes um modelo de aprendizado de linguagem primitivo, com o qual cada agente pode vir a internalizar o

[27] Essa informação advém do fato de que há um evento no *script* instruindo o Gerenciador de Ambiente a transformar o estado do mundo de forma que se produza a noção de que o agente virtual identificado por `a1` tenha se voltado para a sua esquerda.

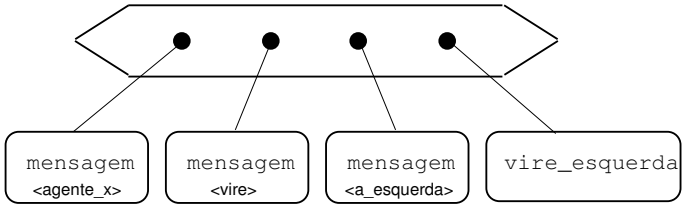

Figura 4.7: Reconhecimento da Primeira Seqüência. A instanciação de um objetivo e a sua execução.

significado de uma tupla *verbo-objeto* simples. Já Rickel and Johnson [2000], como discutido na **Seção 3.4**, trabalharam com um tutor virtual de nome *Steve* que estabelecia diálogos orientados a tarefa com seres humanos; apesar de não dispor de um sistema de aprendizado de linguagem, *Steve* era capaz de estabelecer diálogos relativamente complexos com seres humanos, podendo fornecer explicações para as ações que executava e para as recomendações que fazia.

O consumo dos eventos listados em (**4.8**) dá origem à seqüência representada na **Figura 4.7**, além dos respectivos estados e eventos que são evoluídos. A seqüência construída agrega todos os eventos detectados, uma vez que não há descontinuidade entre eles. Os eventos advêm dos padrões de eventos definidos em (**4.7**).

Seqüências mantêm um tipo de informação que pode se distribuir ao longo de diversos ciclos de uma simulação. Neste trabalho, a memória inicia o processo de construção de uma seqüência imediatamente após o reconhecimento de dois eventos subseqüentes. Novos eventos, desde que em uma sucessão contígua, podem ser anexados a uma seqüência em construção, ou ativa. Uma seqüência em construção torna-se estável, ou inativa, assim que é produzido um ciclo da simulação em que nenhum evento seja detectado; nesse momento, sela-se a seqüência até então ativa e quaisquer eventos consecutivos posteriores darão início à construção de uma nova. Portanto, após iniciada, uma seqüência permanece ativa, ou em composição, até que não se observe nenhum evento em algum instante posterior.

A representação em memória da seqüência apresentada na **Figura 4.7** está explícita em (**4.16**). É possível notar que uma seqüência, cujo predicado é `composed_sequence`, possui apenas 3 componentes, a saber: *ID*, *Abstraction* e *String*; desses, apenas *String* é um componente ainda não visto em outras estruturas. Esse componente corresponde simplesmente à lista dos eventos abrangidos pela seqüência[28]. Atentando-se para o com-

[28]Pode-se notar que eventos do tipo `mensagem` possuem um valor nulo em *Before*, indicando que não

ponente *Timestamp* de cada evento, é simples reconhecer que eles são consecutivos e não há descontinuidade. Apesar de representar um tipo de informação mais elaborado, a composição de uma seqüência é a mais simples dentre as estruturas já apresentadas[29]. No entanto, é a primeira estrutura a explicitar uma informação distribuída ao longo de diversos ciclos de uma simulação, informação que contém naturalmente uma representação temporal. Ao passo que essa característica temporal constitui a primeira estrutura cuja geração estende-se ao longo de alguns ciclos, reconhece-se uma tímida evidência de que a memória começa a administrar uma visão dinâmica de suas observações.

```
recognized_event(29, 4, 3, mensagem, nil, 28, 2).
composed_sequence(30, 5, [29, 39, 48, 55]).
recognized_event(39, 4, 3, mensagem, nil, 38, 3).
recognized_event(48, 4, 3, mensagem, nil, 47, 4).
recognized_event(55, 4, 1, vire_esquerda, 42, 54, 5).
```
(4.16)

Assim, uma seqüência torna explícita na memória dinâmica uma interligação intuitiva entre dois ou mais eventos consecutivos. Além disso, cria-se um novo plano de reconhecimento em memória, pois detém-se agora um conhecimento cuja descrição alastra-se por mais de um ciclo da simulação. Padrões desse tipo abrem a possibilidade de se fornecer previsões sobre estados futuros do mundo. Além disso, faz-se perceptível também a utilização de mais uma forma de retroalimentação, porém com a sutil diferença de que neste ponto é o reconhecimento de eventos, que antes havia sido disparado em função do reconhecimento de estados, que ocasiona o reconhecimento de seqüências.

De acordo com o paradigma seguido neste trabalho, instâncias possuem um papel fundamental na evolução de uma memória mas, sozinhas, pouco valor apresentam. A fim de se obter informação útil para fins de previsão, é necessário evoluir-se abstrações de seqüências. De forma trivial, é possível obter uma abstração que agrega uma seqüência de abstrações de eventos, seguindo a linha até aqui desenvolvida neste trabalho. Ainda assim, essa abstração simples constitui-se em uma fonte limitada de previsões para as necessidades de uma memória dinâmica.

Para continuar a evolução de abstrações mais sofisticadas, é necessário conhecer o conteúdo do restante do arquivo Roteiro; além do extrato já apresentado em (**4.8**), o arquivo ainda dispõe das instruções apresentadas em (**4.17**).

se espera nenhum estado particular no ciclo anterior ao evento.

[29] Um fato interessante é que o *ID* da seqüência é imediatamente maior que o *ID* do primeiro evento, acusando que a construção da seqüência iniciou-se imediatamente após o primeiro evento; de fato, essa foi uma escolha para fins de facilidade de implementação.

```
...
ecoe([slot(objeto, sound(agente_x))]).
ecoe([slot(objeto, sound(vire))]).
ecoe([slot(objeto, sound(a_esquerda))]).
vire_esquerda([slot(ator, state(a1, []))]).
avance_ciclo([]).
ecoe([slot(objeto, sound(agente_y))]).
ecoe([slot(objeto, sound(vire))]).
ecoe([slot(objeto, sound(a_esquerda))]).
vire_esquerda([slot(ator, state(a2, []))]).
avance_ciclo([]).
ecoe([slot(objeto, sound(agente_y))]).
ecoe([slot(objeto, sound(vire))]).
ecoe([slot(objeto, sound(a_esquerda))]).
vire_esquerda([slot(ator, state(a2, []))]).
avance_ciclo([]).
ecoe([slot(objeto, sound(agente_x))]).
ecoe([slot(objeto, sound(vire))]).
ecoe([slot(objeto, sound(a_direita))]).
vire_direita([slot(ator, state(a1, []))]).
avance_ciclo([]).
ecoe([slot(objeto, sound(agente_x))]).
ecoe([slot(objeto, sound(vire))]).
ecoe([slot(objeto, sound(a_direita))]).
vire_direita([slot(ator, state(a1, []))]).
avance_ciclo([]).
ecoe([slot(objeto, sound(agente_y))]).
ecoe([slot(objeto, sound(vire))]).
ecoe([slot(objeto, sound(a_direita))]).
vire_direita([slot(ator, state(a2, []))]).
avance_ciclo([]).
ecoe([slot(objeto, sound(agente_y))]).
ecoe([slot(objeto, sound(vire))]).
ecoe([slot(objeto, sound(a_direita))]).
vire_direita([slot(ator, state(a2, []))]).
avance_ciclo([]).
```

(4.17)

Após o processamento dos ciclos determinados pelas instruções em (4.17), a memória terá evoluído as abstrações simples de seqüência apresentadas em (4.18). Cada uma das abstrações agrega em suas seqüências três eventos abstratos simples e um baseado em restrições[30]. Quando comparadas entre si, essas seqüências abstratas simples revelam

[30]Portanto, todas as estruturas de memória até aqui apresentadas já foram de alguma forma empregadas e possuem algum tipo de interligação mútua.

semelhanças entre as cadeias de eventos abstratos que agregam; abstrair os padrões dessas semelhanças pode ser uma forma de confeccionar um tipo mais sofisticado de abstração de seqüência que represente de forma mais explícita essa informação.

```
abstract_sequence(117, 5, [76, 30], [75, 88, 100, 108]).
abstract_sequence(206, 5, [172, 127], [171, 88, 100, 198]).
abstract_sequence(294, 5, [258, 217], [75, 88, 279, 287]).
abstract_sequence(378, 5, [345, 304], [171, 88, 279, 371]).
```
(4.18)

A fim de satisfazer essas necessidades, um tipo de abstração parcial de seqüências foi concebido. Essa estrutura, identificada pelo predicado `partial_abstract_sequence`, é evoluída sobre abstrações de seqüência simples. Em comparação com as abstrações simples, sua estrutura substitui algumas referências diretas a eventos abstratos por *referências livres*. Ao final da execução da simulação, obtém-se as duas abstrações de seqüência com referências livres apresentadas em (**4.19**). Em uma abstração de seqüência com referências livres, a cadeia de eventos possui elementos que não representam restrições, além de referências diretas para eventos abstratos como em abstrações de seqüência simples. Cada referência livre consiste em um elemento `free_event(`i`)`, onde i é um índice identificando-a. Abstrações de seqüência com referências livres possuem um componente adicional denominado *Associations*, o qual é formado por listas de referências para abstrações de eventos que completam as referências livres, isto é, são as referências encontradas nas cadeias das especializações e que não são contempladas na cadeia da abstração de seqüência com referências livres por não serem comuns a todas as especializações. Cada lista de referências encontra-se ordenada em correspondência com as respectivas referências livres.

```
partial_abstract_sequence(207, 5, [117, 206],
                          [free_event(1), 88, 100, free_event(2)],
                          [[171, 198],
                           [75, 108]]).
partial_abstract_sequence(379, 5, [294, 378],
                          [free_event(1), 88, 279, free_event(2)],
                          [[171, 371],
                           [75, 287]]).
```
(4.19)

Para melhor se compreender a representação, é necessário examinar como uma estrutura do tipo `partial_abstract_sequence` agrega as informações das abstrações de seqüência simples. Analisando-se a estrutura de *ID* 207 em (**4.19**) e constrastando-a com

as estruturas de *ID*s 117 e 206 em (**4.18**), observa-se que as duas últimas foram em algum momento relacionadas como especializações da primeira. Além disso, o componente *String* da estrutura mais sofisticada contém duas referências diretas, 88 e 100, a abstrações de eventos, exatamente as que são comuns às cadeias das abstrações de seqüência simples; o restante da cadeia é formado por duas referências livres, de índices 1 e 2, as quais são satisfeitas pelas referências contidas nas listas de associação $[75, 108]$ e $[171, 198]$. Cada lista de associação corresponde a uma forma de se satisfazer todas as referências livres existentes no componente *String* da estrutura de predicado `partial_abstract_sequence`. Para uma nova seqüência ser posteriormente reconhecida pela abstração de seqüência com referências livres, basta que ela respeite as referências diretas; em caso positivo, uma nova entrada no componente *Associations* seria criada para ela se as existentes não a comportassem, isto é, se nenhuma abstração de seqüência simples a abrangesse.

Semanticamente, a estrutura de *ID* igual a 207 representa uma aproximação ao conceito de toda a atividade relacionada à execução das instruções `vire_esquerda`. Sua cadeia contém referências diretas às estruturas 88 e 100 que, segundo pode-se conferir em (**4.20**), são abstrações de evento representando a seqüência de *tokens* `vire` e `a_esquerda`. A estrutura de *ID* igual a 379, também observada em (**4.19**), é análoga à de *ID* 207, porém com referência à seqüência de *tokens* `vire` e `a_direita`.

```
abstract_sound(84, 3, [352, 311|...], vire).
abstract_state(86, 1, [353, 312|...], ecoando, [slot(som, 84)], [], 10).
abstract_event(88, 4, [354, 313|...], 3, mensagem, nil, 86).
abstract_sound(96, 3, [188, 143|...], a_esquerda).
abstract_state(98, 1, [189, 144|...], ecoando, [slot(som, 96)], [], 10).
abstract_event(100, 4, [190, 145|...], 3, mensagem, nil, 98).
```

(4.20)

A maior diferença entre uma seqüência abstrata com referências livres e uma seqüência abstrata simples resume-se ao fato de que a última restringe tipos de eventos que podem ser reconhecidos em cada um dos itens de sua seqüência, enquanto que a primeira permite que eventos de qualquer tipo preencham suas referências livres. Dessa forma, efetua-se restrições parciais na análise de novas seqüências em reconhecimento. Uma nova seqüência em composição que satisfaça todas as referências diretas de uma seqüência abstrata com referências livres toma proveito das previsões disponíveis por essa abstração mesmo que não haja nenhuma seqüência abstrata simples que comporte a nova seqüência em composição; isso compete a memória a oferecer previsões para uma seqüência cujo primeiro

Estrutura	Informação Ofertada	Aplicação Potencial
estado	Acompanhamento histórico da configuração do ambiente.	Determinação de transições de estado do ambiente.
evento	Acompanhamento histórico das transições de estado relevantes sofridas pelo ambiente.	Identificação de relações entre transições de estado.
seqüência	Acompanhamento histórico das instanciações de objetivos e correspondentes interferências de agentes virtuais.	Previsão de evoluções no estado do ambiente.

Tabela 4.2: Estruturas de Memória: Informação e Aplicação.

evento nunca tenha sido antes observado[31]. Com essa adição, a memória agora sustenta o fornecimento de previsões úteis — previsões para situações novas e não apenas para aquelas que já são conhecidas.

Entretanto, ao mesmo tempo em que uma seqüência abstrata com referências livres compartilha melhor suas previsões, ela trabalha com a suposição de que suas referências diretas, ou restrições parciais, são suficientes para representar algum conceito a que ela se propõe; embora nenhuma das previsões fornecidas por outras estruturas em memória forneça ou constitua-se em garantias, as previsões disponíveis por uma seqüência abstrata com referências livres possuem um grau de certeza ainda mais baixo em virtude da forma como a abstração é obtida. Portanto, previsões oriundas de seqüências abstratas com referências livres podem ser consideradas, a rigor, ainda menos confiáveis.

4.5 Síntese dos Processos de Reconhecimento da Memória

A fim de melhor elucidar o ganho conseguido com a concepção de cada um dos três tipos básicos de estrutura, as quais promovem a evolução de três planos de representação em memória, a **Tabela 4.2** resume algumas noções sobre cada tipo de estrutura. O apanhado concentra-se em definir qual o exato tipo de informação que é conseguido com cada estrutura e quais vantagens elas podem conferir a uma memória dinâmica.

Embora o acompanhamento de uma simulação possa ser interessante para introduzir

[31] Assim, no caso específico da simulação descrita neste trabalho, seria possível oferecer previsões mesmo na atribuição de uma tarefa a um agente virtual ainda não conhecido.

gradualmente cada estrutura, a quantidade de conceitos e processos a serem explanados alonga o processo de exemplificação e, conseqüentemente, a compreensão da dinâmica global é prejudicada. Para sintetizar esses passos e melhor situar o leitor, a **Figura 4.8** sintetiza os processos envolvidos na evolução de cada plano de representação e na ativação de estruturas em memória.

A **Figura 4.8** certifica a ordem em que cada passo é executado e clarifica as condições que determinam os caminhos seguidos no fluxo dos processos de memória a cada ciclo da simulação. Em acréscimo, estão explícitos também os passos do reconhecimento que possuem orientação *bottom-up* ou *top-down*.

4.6 Considerações Finais

Embora todo o aparato apresentado ainda encontre-se em um estado bastante prematuro de desenvolvimento, as estruturas apresentadas conseguem estabelecer uma relação entre o *token* `agente_x`, introduzido em (**4.8**), e a estrutura estado de *Substance* `a1` que representa um agente virtual, observada em (**4.12**), por meio de uma seqüência, cuja instância foi apresentada em (**4.16**), mesmo na ausência de uma conexão explícita entre eles. Isso ocorre porque a seqüência que relacionou todos os eventos em questão sugere que `a1` seja o sujeito da ação invocada por intermédio da instanciação do objetivo expresso pelos *tokens* das mensagens. Embora essa relação seja de fato pouco explícita em virtude do número de estruturas que está sendo considerado, ela está disponível enquanto informação. É possível que com o amadurecimento do trabalho, dispondo-se de estruturas mais sofisticadas e com uma simulação mais rica, essa relação torne-se mais explícita e, por conseqüência, útil.

As abstrações permitem à memória utilizar seus registros de observações como fontes de previsão para eventos e estados futuros. Caso o agente fosse dotado de atuadores para interagir com o mundo, a análise prévia dessas informações poderia ajudá-lo a melhor executar suas ações. Claramente, o ciclo composto por reconhecimento, falha, explicação e previsão resume todo o processo. De posse de suas previsões e na busca por reconhecer uma nova observação, a memória pode incorrer em uma falha — quando nenhuma de suas previsões é satisfeita — e, com o objetivo de não mais apresentar tal deficiência, uma perturbação é gerada e novas abstrações podem ser evoluídas para melhor explicar a observação. As estruturas abstratas apresentadas em (**4.19**) são candidatas a sofrerem uma reavaliação que pode conduzir a concepções mais sofisticadas e apropriadas a fim de melhor contribuírem em um processo de fornecimento de expectativas sobre futuras

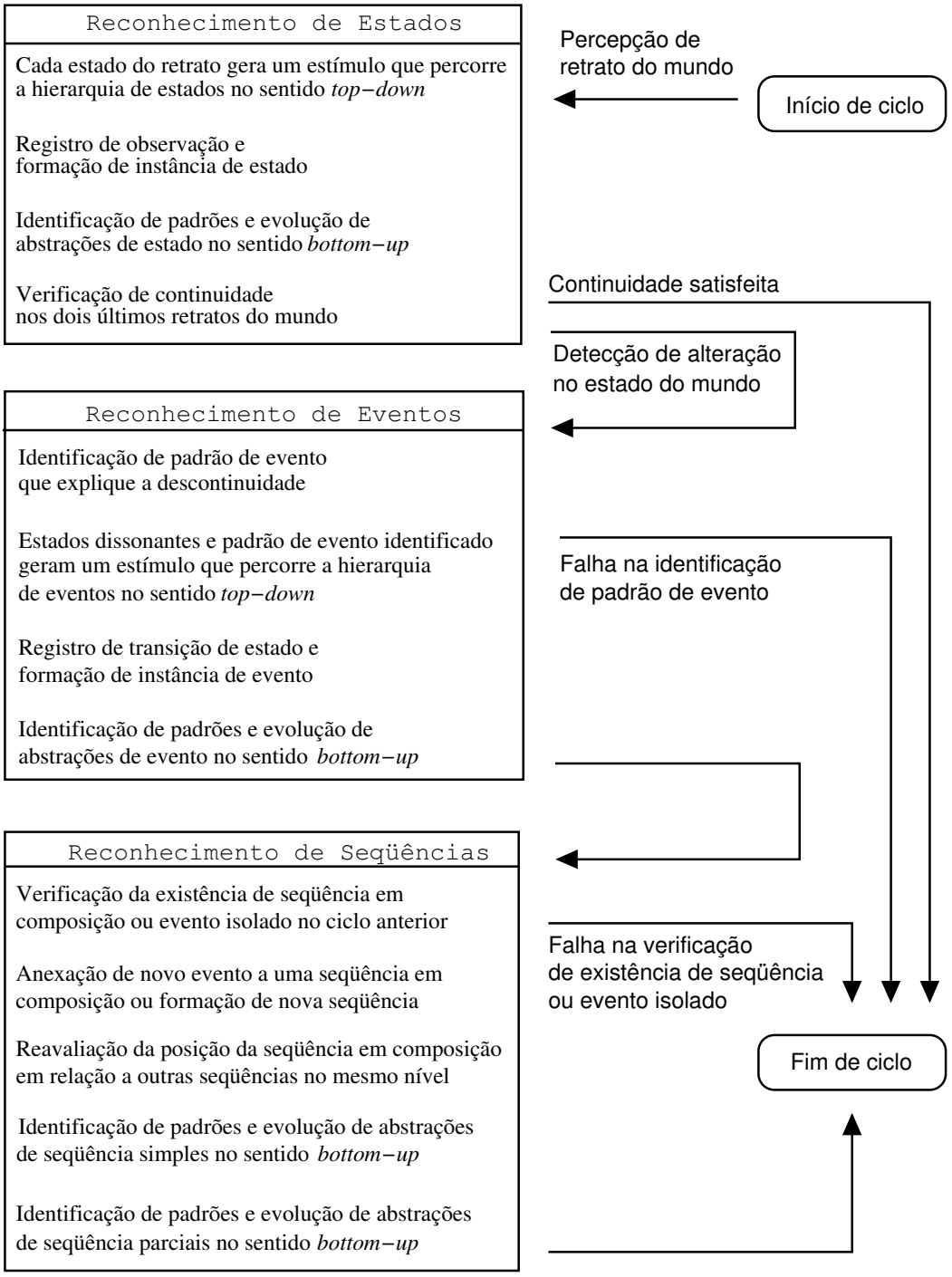

Figura 4.8: Síntese dos Processos de Memória. A seqüência de atividades que caracteriza os processos de reconhecimento.

evoluções na simulação e assim melhor qualificarem a memória do agente Observador.

Enquanto é comum na literatura a alteração de um algoritmo de aprendizado para comportar um esquema de representação formal ou o abandono de uma representação formal para se permitir a aplicação de um algoritmo de aprendizado, este capítulo apresentou um estudo segundo o qual processos e representações foram evoluídos concomitantemente. Ao passo que formas de representação de conhecimento tradicionais utilizam símbolos representando conceitos, algoritmos de aprendizado clássicos são geralmente orientados a um tipo de informação bem mais básico, tornando tais abordagens por vezes inadequadas a um problema que já possui uma representação formal. Esse desencontro é condenado por Schank [1999], o qual sugere que uma memória dinâmica robusta possivelmente possuiria tanta interdependência entre seus esquemas de representação e processos de alteração de memória que seria impossível ou inútil desmembrá-los.

Capítulo 5

Conclusão

Mesmo não tendo sido objeto de argumentação ao longo do trabalho, não se pode deixar de destacar a complexidade envolvida no tema: embora as estruturas desenvolvidas e os processos de modificação de memória tenham sido inspirados em trabalhos sólidos [Schank, 1972, Schank and Abelson, 1977, Schank, 1982], o presente trabalho introduz suas próprias versões de cada conceito, peculiarmente adequadas ao domínio de aplicação e aos objetivos fixados, e explora uma arquitetura flexível do sistema que permite a um projetista experimentar sob diversos aspectos. Além disso, na contramão de muitos trabalhos da literatura, buscou-se continuamente aproximar os conceitos introduzidos de estruturas defendidas como existentes nos seres humanos e bastante respeitadas em círculos de pesquisadores ligados à Psicologia e à Ciência Cognitiva.

O relativo pioneirismo do trabalho também corrobora uma percepção negativa que possa advir das deficiências a serem citadas na **Seção 5.1**. Por outro lado, se no âmbito científico isso for assumido como conseqüência de uma agenda que determina o estabelecimento de objetivos preliminares, a fim de que passos possam ser revistos e remodelados até a concepção de um trabalho mais robusto, os estudos realizados efetivamente contribuem para que possíveis trabalhos futuros possam levar o tema adiante e aproximar-se de um modelo de memória dinâmica que se alinhe às necessidades de um agente observador.

5.1 Limitações

Esta dissertação focou-se a versar sobre um assunto em maior ou menor grau desatualizado e centrado em uma definição de ambiente demasiado simplista, praticamente específica

para o exemplo apresentado; não há robustez a alterações de grandeza considerável na simulação. Além disso, o trabalho também pode ser classificado como de difícil comparação com outras técnicas — e, portanto, difícil de ser analisado a partir de uma postura científica conservadora — e com resultados palpáveis aquém daqueles esperados de um trabalho deste porte, portanto perfeitamente passível de ser massiva e severamente criticado. Apesar dessas ressalvas, argumenta-se que o estudo demonstra importantes reflexões sobre o tipo de estruturação e os processos internos possivelmente requeridos por uma memória dinâmica localizada em um agente cujo objetivo é compreender relações temporais entre estados do mundo a ponto de estar apta a fornecer previsões que garantam a seu detentor algum grau de compreensão do ambiente no qual ele habita ou do qual ele recebe informações sensoriais preparadas.

Partindo-se das mais notórias para as mais sutis restrições, o trabalho refere-se constantemente a um *agente observador*. O leitor tem o direito de argumentar que de forma alguma o programa referido pode ser denominado agente, pois lhe faltam até mesmo atuadores; conseqüentemente, o *agente* nunca executa qualquer ação sobre o mundo. Como se não bastasse, supondo a existência de um módulo de atuação sobre o mundo, não são feitas referências a como o agente poderia reconhecer a atribuição de um objetivo a si e como ele poderia utilizar o conhecimento obtido para ativar seus operadores apropriadamente. Para fins de delimitação de objetivos e para assegurar a viabilidade da implementação do trabalho no cronograma disponível, dispositivos atuadores foram preteridos. Com isso, a investigação minuciosa de como se estabelecer uma conexão entre o conhecimento adquirido e um módulo de atuação também foi mantida fora do plano seguido no trabalho. Entretanto, dadas as informações fornecidas ao longo do trabalho, não é muito difícil levantar idéias de como essas funções poderiam ser desenvolvidas. A auto-referência dependeria provavelmente de um preparo específico na memória, a qual poderia disparar um processo separado responsável pela atuação. A insistência na utilização do termo agente é fruto da visão que se tem desse trabalho como um passo no desenvolvimento de um programa que contará com as propriedades exibidas por agentes; em nenhum momento essa "incoerência" visa ferir definições como as apresentadas na **Subseção 3.2.1**.

Outro ponto discutível é a inexistência de uma representação no mundo para o agente observador. Porém, uma vez que esse agente não interage com os agentes virtuais, seria um fator complicador desnecessário inserir sua representação no ambiente.

Pode ser surpreendente também o fato de o trabalho ignorar noções como ruído — presente na maioria dos domínios de aplicação — e a dificuldade prática que reside em

se tentar fornecer uma visão completa do mundo a um agente — trabalhos orientados a agentes com freqüência modelam seus sistemas como ambientes onde agentes dispõem de visões apenas parciais, para melhor representar o mundo real. Acredita-se que em se considerando o estágio em que os fundamentos desta abordagem se encontram, essas características precisam ser omitidas para viabilizar o estudo. Conforme possíveis estudos ulteriores agreguem mais robustez às abordagens, é desejável que essas propriedades sejam gradativamente inseridas no ambiente.

Não se distingue em momento algum noções como memória de curto e de longo prazo[1]. Parece não haver dúvida de que é preciso inserir algum tipo de fator de decaimento nas estruturas em memória e nas ligações entre elas; entretanto, mais uma vez, essa característica é remetida a trabalhos futuros.

Um problema bastante conhecido pela comunidade de IA e não mencionado até aqui é o *problema do frame*[2], enunciado por McCarthy and Hayes [1969] no fim da década de 60. Apesar de ter sido originalmente identificado em sistemas baseados em lógica, foi posteriormente estendido e ainda é tido como um desafio para a pesquisa em IA. Nenhuma técnica para tratamento desse problema foi usada neste trabalho.

5.2 Contribuições e Discussão Final

Embora sejam feitas numerosas suposições simplificadas ao longo do texto, o conjunto de problemas tratados não é trivial. O trabalho, que adota uma estratégia de estudo de caso com o acompanhamento de uma simulação, introduzindo conceitos à medida em que eles se fazem necessários, atinge os objetivos enumerados no **Capítulo 1**.

Ao longo do trabalho, descreve-se uma simulação na qual um agente observador adquire informações para a evolução de uma memória dinâmica, a qual lhe fornece uma fundação para que previsões sobre estados futuros sejam obtidas. A arquitetura do sistema proporciona flexibilidade para que um projetista determine o estado inicial, as transições de estado e o que pode ser aprendido em cada simulação em arquivos com sintaxe legível para um projetista. A definição de padrões de eventos também pelo projetista facilita ao

[1] Essas noções são atribuídas a um famoso artigo de Miller [1956], onde ele defende a existência dessa distinção na memória humana baseando-se em resultados experimentais.

[2] O problema do *frame* fundamenta-se no aspecto não-monotônico de como determinar eficientemente quais itens permanecem sem alteração no ambiente após cada transição de estado ou, mais formalmente, o desafio de se representar os efeitos de uma ação em um sistema lógico sem representar explicitamente um número exponencial de *não-efeitos* intuitivamente óbvios [Shanahan, 2004].

agente observador o trabalho de reconhecer eventos.

A simulação consiste em um ambiente de *software* com agentes virtuais que executam ações mediante instanciações de objetivos. Recebendo um retrato do mundo por ciclo da simulação, o agente observador, cujo aparato cognitivo resume-se à memória, busca reconhecer os estados do mundo. O reconhecimento de estados habilita a memória a reconhecer eventos e estes, seqüências. Isso acarreta um ciclo de previsão, reconhecimento, falha e modificação que caracterizam uma memória dinâmica e lhe conferem a propriedade de auto-evolução. É visível, nesse ponto, a influência da Teoria de Memória Dinâmica [Schank, 1982] na definição das estruturas e dos processos de sua memória.

As informações sensoriais fornecidas ao agente Observador a cada ciclo da simulação respeitam um formalismo elaborado, desprovido de elementos armazenando sinais puros. Isso permite que se projete a memória para desenvolver-se a partir de um tipo de informação mais significativo, inibindo-se a necessidade de se buscar informação em dados de baixo nível. Embora seja fato que sensores reais restringem-se a repassar sinais puros, esses dados puros poderiam ser pré-processados por algum módulo anterior à memória, a fim de garantir a ela o envio de uma informação mais estruturada.

Todas as estruturas de memória, bem como sua organização e as formas de indexação que as regem, são peculiares a este trabalho. Embora inspiradas em trabalhos consagrados [Schank, 1972, Schank and Abelson, 1977, Minsky, 1975], as minúcias do domínio de aplicação requeriam adaptações com freqüência inovadoras. As expectativas deste trabalho é que estados, eventos e seqüências possam vir a derivar alicerces robustos para o desenvolvimento de uma memória dinâmica em domínios de aplicação simplificados semelhantes ao mundo dos blocos.

Cenários Baseados em Objetivos exigem participação ativa do sujeito da aprendizagem em uma estória na qual ele, interagindo, adquire conhecimento [Schank, 1992]. O agente observador não participa de forma ativa em nenhum momento ao longo da simulação, por razões já discutidas. A posição deste trabalho é que observações podem orientar e facilitar o aprendizado e a participação ativa atua como um importante estágio complementar para aperfeiçoamento, sendo que sua ausência não denigre as contribuições do trabalho. Aliás, Rickel and Johnson [2000] utilizam uma abordagem na área de educação que dispõe de uma carga elevada de observação, embora combinada com atuação.

Fern et al. [2002] utilizaram uma abordagem com diversas semelhanças em relação à adotada neste trabalho; em seu artigo, descrevem um algoritmo baseado em uma lógica temporal simples, por eles definida, que aprende definições de eventos visuais a partir de

seqüências de vídeo. O aprendizado segue uma estratégia do tipo *subsumption*[3], em que elementos mais específicos são combinados para se abstrair outros mais gerais. Assim como na presente dissertação, eles utilizam apenas exemplos positivos em seus experimentos. Para eles, eventos são relacionais e parametrizados por objetos específicos; na abordagem descrita nesta dissertação, no entanto, pode haver mais de um nível de abstração de um mesmo tipo de evento, sendo elas diferenciadas pela abrangência que possuem e pelo tipo de informação que carregam. O domínio de aplicação por eles tratado é não-determinístico e síncrono, assim como o ambiente objetivo deste trabalho. Os autores perfazem demonstrações e provas lógicas da validade de suas idéias; a visão de que um dos alicerces de interpretação do mundo pelos seres humanos consiste em eventos corrobora a utilização de uma noção semelhante dessa estrutura neste trabalho.

Como foi possível notar ao longo do trabalho, a representação do ambiente utiliza uma abordagem *relativa* ou *funcional*: elementos são associados com referência relativa a outros — `a_esquerda`, `a_direita`. Esse tipo de representação também foi utilizada por Kaelbling et al. [2001], no trabalho com o domínio mundo dos blocos, e por Agre and Chapman [1987], no agente `PENGI`, protagonista de um jogo. Os dois trabalhos são comentados nas **Subseção 3.2.3** e **Subseção 3.3.2**.

De posse de todas as conclusões a respeito do trabalho, pode-se dizer que uma memória dinâmica não seria uma competência adicional a um agente artificial; em um patamar de robustez ideal, uma memória dinâmica seria o cerne cognitivo que poderia desde conferir-lhe mais informações para atuar até auxiliá-lo a adquirir competências automaticamente.

5.3 Sugestões para Trabalhos Futuros

Um primeiro trabalho futuro certamente seria a revisão de todos os conceitos introduzidos neste trabalho, a fim de garantir mais robustez à abordagem. Na seqüência, poderia ser fixado como objetivo a evolução de abstrações mais sofisticadas, a fim de buscar a previsão exata de que um *token* como `agente_x` exija algum tipo de ação do agente virtual `a1`, qualquer que seja essa ação. Ampliando-se o número de exemplos de instanciações de objetivos no roteiro de forma apropriada, a evolução dessas abstrações mais sofisticadas

[3] O termo *subsumption* é mais conhecido por ser o nome da famosa arquitetura introduzida por Brooks [1986], na qual um agente é constituído de módulos simples que são combinados em camadas. A integração dessas camadas permite a exibição de comportamentos complexos, ainda que reativos. As entradas para as camadas superiores são as saídas das camadas inferiores. Não há módulo de raciocínio centralizado e não existe representação baseada em símbolos.

pode ser um objetivo palpável em um primeiro plano.

Outras características avançadas que podem ser conferidas à simulação e, por conseqüência, embutidas no sistema, uma vez que requerem alterações de grande porte desde os modelos do projetista até os processos da memória dinâmica, incluem, mas não estão limitadas a: expansão do mundo para que ele disponha de coordenadas discretas, possibilitando locomoção dos agentes virtuais, e ampliação das formas de representação para que comportem localização discreta; inclusão de objetos que possam ser apanhados por agentes virtuais, mediante objetivos apropriados a eles dirigidos, e transportados de um ponto a outro; um processo de previsão explícito que tenha uma influência maior sobre a evolução da memória; e atuadores para possibilitar ao agente observador ter participação ativa no (atual) *habitat* de agentes virtuais e responder por objetivos a ele endereçados.

Todas essas características fazem parte de um conjunto mínimo de requisitos para se obter um agente observador que exiba um grau de inteligência e robustez suficiente para se dar início a planos de adequá-lo a algum domínio de aplicação real. Em suma, é preciso que o agente observador exiba comportamentos semelhantes ao de `HOMER` [Vere and Bickmore, 1990], comentado na **Subseção 3.2.3**. Para isso, é preciso permitir que o agente observador esteja apto a internalizar as competências exibidas pelos agentes virtuais, apresentando graus de compreensão variando com a complexidade da tarefa executada.

Referências Bibliográficas

P. E. Agre and D. Chapman. PENGI: An Implementation of a Theory of Activity. In *Proceedings of the 6th National Conference of the American Association for Artificial Intelligence*, pages 268–272, Seattle, WA, 1987. Morgan Kaufmann Publishers. 39, 54, 99

C. Argyris and D. Schon. *Organizational Learning: A Theory of Action Perspective*. Addison-Wesley, Reading, MA, 1978. 10

B. C. Ávila. *Uma Abordagem Paraconsistente Baseada em Lógica Evidencial para Tratar Exceções em Sistemas de Frames com Múltipla Herança*. PhD thesis, Universidade de São Paulo, São Paulo, SP, 1996. 18

Eric. B. Baum and I. Durdanovic. Evolution of Cooperative Problem-Solving in an Artificial Economy. *Neural Computation*, 12(12):2743–2775, Dec 2000. 3, 46, 54

L. Birnbaum and M. Selfridge. Conceptual Analysis of Natural Language. In Schank and Riesbeck [1981a], chapter 13, pages 318–353. 16

A. H. Bond and L. Gasser, editors. *Readings in Distributed Artificial Intelligence*. Morgan Kaufmann Publishers, San Mateo, CA, 1988. 31

J. M. Bradshaw. An Introduction to Software Agents. In J. M. Bradshaw, editor, *Software Agents*, pages 3–46. AAAI Press/MIT Press, Cambridge, MA, 1997. 35, 36, 39

J. M. Bradshaw, S. Dutfield, P. Benoit, and J. D. Wooley. KAoS: Toward an Industrial-strength Open Agent Architecture. In J. M. Bradshaw, editor, *Software Agents*, pages 375–418. AAAI Press/MIT Press, Cambridge, MA, 1997. xi, 36, 37, 38

M. E. Bratman. *Intentions, Plans and Practical Reason*. Harvard University Press, 1987. 39

M. E. Bratman, D. J. Israel, and M. E. Pollack. Plans and Resource-bounded Practical Reasoning. *Computational Intelligence*, 4(4):349–355, 1988. 39

R. A. Brooks. A Robust Layered Control System for a Mobile Robot. *IEEE Journal of Robotics and Automation*, 2(1):14–23, Mar 1986. 99

J. C. Brustolini. Autonomous Agents: Characterization and Requirements. Technical Report CMU-CS-91-204, Carnegie Mellon University, Pittsburgh, PA, Nov 1991. 45, 46

D. Chapman. Planning for Conjunctive Goals. *Artificial Intelligence*, 32(3):333–377, 1987. 12, 29

Y. Chauvin and D. Rumelhart. *Backpropagation: Theory, Architectures and Applications*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1995. 11

P. Cheeseman, M. Self, J. Kelly, J. Stutz, W. Taylor, and D. Freeman. Bayesian Classification. In *Proceedings of the* 7^{th} *Annual Conference on Artificial Intelligence*, pages 607–611. AAAI, Morgan Kaufmann Publishers, 1988. 8

N. Chomsky. *Aspects of the Theory of Syntax*. MIT Press, Cambridge, MA, 1965. 17

M. D. Cohen, J. G. March, and J. P. Olsen. A Garbage Can Model of Organizational Choice. *Administrative Science Quarterly*, 17(1):1–25, Mar 1972. 10

P. R. Cohen and H. J. Levesque. Intention is Choice with Commitment. *Artificial Intelligence*, 42(2-3):213–261, 1990. 38

T. Cover and P. Hart. Nearest Neighbor Pattern Classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967. 7

N. C. A. da Costa, J. M. Abe, J. I. da Silva Filho, A. C. Murolo, and C. F. S. Leite. *Lógica Paraconsistente Aplicada*. Atlas, São Paulo, SP, 1999. 18

R. Davis, H. Shrobe, and P. Szolovits. What is a Knowledge Representation? *AI Magazine*, 14(1):17–33, 1993. 13

K. S. Decker and K. P. Sycara. Intelligent Adaptive Information Agents. *Journal of Intelligent Information Systems*, 9(3):239–260, 1997. 45

E. L. dos Santos, F. M. Hasegawa, B. C. Ávila, and F. Enembreck. Conceptual Information Retrieval. In F. F. Ramos, H. Unger, and V. Larios, editors, *Proceedings of the* 3^{rd} *International Symposium on Advanced Distributed Systems*, volume 3061 of *Lecture Notes in Computer Science*, pages 137–144, Guadalajara, Mexico, Jan 2004. Springer-Verlag. 67

E. L. dos Santos, F. M. Hasegawa, B. C. Ávila, and C. A. A. Kaestner. Paraconsistent Knowledge for Misspelling Noise Reduction in Documents. In J. M. Abe and J. I. da Silva Filho, editors, *Advances in Logic, Artificial Intelligence and Robotics*, volume 85 of *Frontiers in Artificial Intelligence and Applications*, pages 144–151, Amsterdam, The Netherlands, 2002. 3^{rd} Congress of Logic Applied to Technology, IOS Press. 18

R. O. Duda, P. E. Hart, K. Konolige, and R. Reboh. A Computer-based Consultant for Mineral Exploration. Technical report, SRI International, Menlo Park, CA, Sep 1979. 11

R. Duncan and A. Weiss. Organizational Learning: Implications for Organizational Design. In B. M. Staw, editor, *Research in Organizational Behavior*, volume 1, pages 75–123. JAI Press, Greenwich, CT, 1979. 10

E. H. Durfee and J. S. Rosenschein. Distributed Problem Solving and Multi-Agent Systems: Comparisons and Examples. In *Proceedings of the* 13^{th} *International Distributed Artificial Intelligence Workshop*, pages 94–104, Seattle, WA, Jul 1994. 41

S. Dzeroski, L. De Raedt, and H. Blockeel. Relational Reinforcement Learning. In *Proceedings of the* 15^{th} *International Conference on Machine Learning*, pages 136–143, San Francisco, CA, 1998. Morgan Kaufmann Publishers. 2, 54, 63, 69

F. Enembreck. Contribution à la Conception d'Agents Assistants Personnels Adaptatifs. PhD thesis, Université de Technologie de Compiègne, Compiègne, France, Dec 2003. 49

T. Erickson. Designing Agents as if People Mattered. In J. M. Bradshaw, editor, *Software Agents*. AAAI Press, Menlo Park, CA, 1997. 34

J. Ferber. *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*. Addison-Wesley, Boston, MA, 1999. 76

J. Ferber and J.-P. Müller. Influences and Reaction: A Model of Situated Multiagent Systems. In *Proceedings of the 2nd International Conference on Multi-Agent Systems*, pages 72–79, Kyoto, Japan, 1996. AAAI Press. 76

A. Fern, R. Givan, and J. M. Siskind. Specific-to-General Learning for Temporal Events with Application to Learning Event Definitions from Video. *Journal of Artificial Intelligence Research*, 17:379–449, 2002. 98

R. E. Fikes and N. J. Nilsson. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence*, 2(3/4):189–208, 1971. 12, 29, 76

S. Franklin and A. Graesser. Is it an Agent or just a Program? A Taxonomy for Autonomous Agents. In *Proceedings of the 3rd International Workshop on Agent Theories, Architectures and Languages*, pages 21–35. Springer-Verlag, 1996. 32, 33, 35, 40, 45

M. P. Georgeff, B. Pell, M. E. Pollack, M. Tambe, and M. J. Woold. The Belief-Desire-Intention Model of Agency. In J. P. Müller, M. P. Singh, and A. S Rao, editors, *Proceeding of the 5th International Workshop on Intelligent Agents V. Agent Theories, Architectures and Languages*, volume 1555 of *Lecture Notes in Computer Science*, pages 1–10, London, UK, 1999. Springer-Verlag. 39

S. Grand and D. Cliff. Creatures: Entertainment Software Agents with Artificial Life. *Autonomous Agents and Multi-Agent Systems*, 1(1):39–57, 1998. 2, 50, 51, 54, 62, 85

D. Greenstein and J. Odell. Agent-based Manufacturing: Part II — Putting Agents to Work in the Real World. *Distributed Computing*, pages 1–5, Jul 1999. 45

R. V. Guha, D. B. Lenat, K. Pittman, D. Pratt, and M. Shepherd. CYC: A Midterm Report. *Communications of the ACM*, 33(8), 1990. 1, 11

D. Haussler. Quantifying Inductive Bias: AI Learning Algorithms and Valiant's Learning Framework. *Artificial Intelligence*, 36(2):177–221, Sep 1988. 11

F. Hayes-Roth. Rule-based Systems. *Communications of the ACM*, 28(9):921–932, Sep 1985. 7

F. Hayes-Roth. An Architecture for Adaptive Intelligent Systems. *Artificial Intelligence*, 72(1-2):329–365, Jan 1995. 40

F. Hembecker. Parser Baseado em Casos na Compreensão de Diálogo: uma Proposta de Aplicação do Direct Memory Access Parsing. Master's thesis, Pontifícia Universidade Católica do Paraná, Curitiba, PR, Jul 2002. 17

C. Hewitt. Viewing Control Structures as Patterns of Passing Messages. *Artificial Intelligence*, 8(3):323–364, 1977. 32

J. H. Holland. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Ann Arbor, MI, 1975. 8, 11

M. Holsheimer and A. Siebes. Data Mining: The Search for Knowledge in Databases. Technical Report CS-R9406, CWI, Amsterdam, The Netherlands, 1994. 8, 11

E. B. Hunt, J. Marin, and P. T. Stone. *Experiments in Induction*. Academic Press, New York, NY, 1966. 7

A. Isaac and C. Sammut. Goal-directed Learning to Fly. In T. Fawcett and N. Mishra, editors, *Proceedings of the* 20^{th} *International Conference on Machine Learning*, pages 258–265, Washington, DC, 2003. AAAI Press. 2, 51, 52, 62

N. R. Jennings. On Agent-based Software Engineering. *Artificial Intelligence*, 117(2): 277–296, Mar 2000. 31

N. R. Jennings, K. P. Sycara, and M. J. Wooldridge. A Roadmap of Agent Research and Development. *Autonomous Agents and Multi-Agent Systems*, 1(1):7–38, 1998. 42, 43

L. P. Kaelbling. A Situated-automata Approach to the Design of Embedded Agents. *SIGART Bull*, 2(4):85–88, Aug 1991. 36

L. P. Kaelbling, T. Oates, N. Hernandez, and S. Finney. Learning in Worlds with Objects. In P. R. Cohen and T. Oates, editors, *Papers from 2001 AAAI Spring Symposium – Learning Grounded Representations*, Technical Reports, pages 31–36. AAAI Press, 2001. 2, 46, 53, 63, 67, 69, 99

M. Kearns and U. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, Cambridge, MA, 1994. 11

J. Kolodner. *Case-based Reasoning*. Morgan Kaufmann Publishers, San Mateo, CA, 1993. 12, 14, 15, 18

D. B. Lange and M. Oshima. *Programming and Deploying Java^{TM} Mobile Agents with Aglets^{TM}*. Addison-Wesley, Boston, MA, 1998. 34

D. B. Leake, editor. *Case-based Reasoning: Experiences, Lessons and Future Directions*. AAAI Press/MIT Press, Menlo Park, CA, 1996. 18

D. B. Lenat and R. V. Guha. *Building Large Knowledge-based Systems*. Addison-Wesley, Reading, MA, 1989. 1, 11

A. R. Luria. A Psicologia Experimental e o Desenvolvimento Infantil. In Vygotsky et al. [2001]. Tradução de M. da P. Villalobos. 44, 117

P. Maes. Agents that Reduce Work and Information Overload. *Communications of the ACM*, 37(7):30–40, Jul 1994a. 33, 48, 49

P. Maes. Modeling Adaptive Autonomous Agents. *Artificial Life*, 1(1-2):135–162, 1994b. 45, 46

P. Maes. Artificial Life Meets Entertainment: Lifelike Autonomous Agents. *Communications of the ACM*, 38(11):108–114, Nov 1995. 54

J. G. March and H. A. Simon. *Organizations*. Wiley, New York, NY, 1958. 9

C. E. Martin. Case-based Parsing. In Riesbeck and Schank [1989b], chapter 10, pages 319–352. 16

C. E. Martin. Micro DMAP. In Riesbeck and Schank [1989b], chapter 11, pages 353–389. 17

C. E. Martin. Direct Memory Access Parsing. Technical Report CS 93-07, University of Chicago, Chicago, IL, 1993. 16, 17, 72

J. McCarthy and P. J. Hayes. Some Philosophical Problems from the Standpoint of Artificial Intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence*, volume 4, pages 463–502. Edinburgh University Press, Edinburgh, Scotland, 1969. 97

W. S. McCulloch and W. H. Pitts. A Logical Calculus of Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943. 7

J. McDermott. R1: A Rule-based Configurer of Computer Systems. *Artificial Intelligence*, 19(1):39–88, 1982. 11

K. L. McGraw and K. Harbison-Briggs. *Knowledge Acquisition: Principles and Guidelines*. Prentice-Hall, Englewood Cliffs, NJ, 1989. 7

G. A. Miller. The Magical Number Seven, Plus or Minus Two – Some Limits on Our Capacity for Processing Information. *Psychological Review*, 63(2):81–97, 1956. 97

M. Minsky. A Framework for Representing Knowledge. In P. H. Winston, editor, *The Psychology of Computer Vision*, pages 211–277. McGraw-Hill, 1975. 3, 7, 12, 22, 25, 27, 29, 30, 67, 69, 98

M. Minsky. *The Society of Mind*. Touchstone Books, New York, NY, 1985. 8, 9

T. Mitchell. *Machine Learning*. McGraw-Hill, New York, NY, 1997. 11, 12

J. Mosterín. *Racionalidad y Acción Humana*. Alianza, Madrid, Espanha, 2ª edition, 1987. 116

J. P. Müller and M. Pischel. An Architecture for Dynamically Interacting Agents. *International Journal of Intelligent and Cooperative Information Systems*, 3(1):25–46, 1994. 41

A. Newell and H. A. Simon. GPS, A Program that Simulates Human Thought. In E. Feigenbaum and J. Feldman, editors, *Computers and Thought*, pages 279–293. McGraw-Hill, New York, NY, 1963. 7

H. S. Nwana. Software Agents: An Overview. *Knowledge Engineering Review*, 11(3): 1–40, Sep 1996. 33, 38, 40

J. Odell. Agents and Beyond: A Flock is not a Bird. *Distributed Computing*, pages 52–54, Apr 1998. 45

J. Odell. Agents (Part 1): Technology and Usage. *Executive Report*, 3(4), 2000a. 45

J. Odell. Agents (Part 2): Complex Systems. *Executive Report*, 3(6), 2000b. 45

C. Petrie. Agent-based Software Engineering. In P. Ciancarini and M. Wooldridge, editors, *Proceedings of the* 1^{st} *International Workshop on Agent Oriented Software Engineering*, volume 1957 of *Lecture Notes in Computer Science*, pages 58–76. Springer-Verlag, 2001. 31

J. Piaget. A Epistemologia Genética. In V. Civita, editor, *Piaget*, Os Pensadores, pages 3–10. Abril Cultural, São Paulo, SP, 2^{nd} edition, 1983a. Tradução de N. C. Caixeiro. 108, 117

J. Piaget. Problemas de Psicologia Genética. In Piaget [1983a], pages 3–10. Tradução de C. E. A. Di Piero. 44, 117

D. Pierce and B. J. Kuipers. Map Learning with Uninterpreted Sensors and Effectors. *Artificial Intelligence*, 92(1-2):169–227, May 1997. 46, 52

M. R. Quillian. Semantic Memory. In M. Minsky, editor, *Semantic Information Processing*, pages 227–270. MIT Press, Cambridge, MA, May 1968. 7, 25

M. R. Quillian. The Teachable Language Comprehender: A Simulation Program and Theory of Language. *Communications of the ACM*, 12(8), 1969. 16

J. R. Quinlan. Induction of Decision Trees. *Machine Learning*, 1(1):81–106, 1986. 11

J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, San Francisco, CA, 1993. 11, 51

A. S. Rao and M. P. Georgeff. Modeling Rational Agents within a BDI-Architecture. In *Proceedings of the* 2^{nd} *International Conference on Principles of Knowledge Representation and Reasoning*, number 473–484, Cambridge, MA, Apr 1991. Morgan Kaufmann Publishers. 39

E. Rich and K. Knight. *Artificial Intelligence*. McGraw-Hill, New York, NY, 1991. 8, 11

J. Rickel and W. L. Johnson. Task-oriented Collaboration with Embodied Agents in Virtual Worlds. In J. Cassel, J. Sullivan, S. Prevost, and E. Churchill, editors, *Embodied Conversational Agents*, chapter 4, pages 95–122. MIT Press, Boston, MA, 2000. 2, 54, 55, 62, 80, 86, 98

C. K. Riesbeck. From Conceptual Analyzer to Direct Memory Access Parsing: An Overview. In N. E. Sharkey, editor, *Advances in Cognitive Science*, Ellis Horwood Series in Cognitive Science, chapter 8, pages 236–258. Ellis Horwood, New York, NY, Dec 1986. 16, 17

C. K. Riesbeck and C. E. Martin. Direct Memory Access Parsing. Technical Report 354, Yale University, Yale, IA, Jan 1985. 16, 72

C. K. Riesbeck and R. C. Schank. Case-based Reasoning: An Introduction. In *Inside Case-based Reasoning* Riesbeck and Schank [1989b], chapter 1, pages 1–24. 10, 12

C. K. Riesbeck and R. C. Schank, editors. *Inside Case-based Reasoning*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1989b. 14, 17, 18, 106, 109

S. J. Russel and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Upper Saddle River, NJ, 1995. 34

E. D. Sacerdoti. The Non-linear Nature of Plans. In *Proceedings of the* 4^{th} *International Joint Conference on Artificial Intelligence*, pages 206–214, San Francisco, CA, 1975. Morgan Kaufmann Publishers. 12, 29

C. Sammut, S. Hurst, D. Kedzier, and D. Michie. Learning to Fly. In *Proceedings of the* 9^{th} *International Workshop on Machine Learning*, pages 385–393, Aberdeen, Scotland, 1992. Morgan Kaufmann Publishers. 51

R. C. Schank. Conceptual Dependency: A Theory of Natural Language Understanding. *Cognitive Psychology*, 3:552–631, 1972. 3, 4, 7, 22, 23, 24, 29, 57, 67, 95, 98

R. C. Schank. *Dynamic Memory – A Theory of Reminding and Learning in Computers and People*. Cambridge University Press, Cambridge, UK, 1982. 3, 4, 5, 12, 18, 20, 28, 30, 57, 63, 69, 95, 98

R. C. Schank. *Explanation Patterns: Understanding Mechanically and Creativily*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1986. 3, 18

R. C. Schank. Goal-based Scenarios. Technical Report 36, Northwestern University, Chicago, IL, Dec 1992. 2, 22, 57, 64, 98

R. C. Schank. Goal-based Scenarios. *Educational Technology*, 34(9):3–32, 1994. 22

R. C. Schank. *Dynamic Memory Revisited*. Cambridge University Press, Cambridge, UK, 1999. 2, 3, 8, 11, 12, 13, 14, 16, 17, 18, 19, 20, 21, 22, 44, 76, 94

R. C. Schank and R. Abelson. *Scripts, Plans, Goals and Understanding: An Inquiry into Human Knowledge Structures*. The Artificial Intelligence Series. Lawrence Erlbaum Associates, Hillsdale, NJ, 1977. 1, 3, 7, 11, 22, 27, 28, 29, 57, 69, 83, 95, 98

R. C. Schank and C. Cleary. *Engines for Education*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1995. 13, 18, 21

R. C. Schank and A. Kass. A Goal-based Scenario for High-School Students. *Communications of the ACM*, 39(4):28–29, 1996. 22

R. C. Schank and C. K. Riesbeck, editors. *Inside Computer Understanding: Five Programs Plus Miniatures*. Lawrence Erlbaum Associates, Hillsdale, NJ, Jun 1981a. 23, 101, 110

R. C. Schank and C. K. Riesbeck. Our Approach to Artificial Intelligence. In *Inside Computer Understanding: Five Programs Plus Miniatures* Schank and Riesbeck [1981a], chapter 1, pages 1–9. 17, 23, 24, 27, 29

P. M. Senge. *The Fifth Discipline*. Doubleday, New York, NY, 1990. 10

M. Shanahan. The Frame Problem. In E. N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. 2004. URL `http://plato.stanford.edu/archives/spr2004/entries/frame-problem/`. 97

E. H. Shortliffe. *Computer-based Medical Consultations: MYCIN*. Elsevier, New York, NY, 1976. 11

M. P. Singh. *Multiagent Systems: A Theoretical Framework for Intentions, Know-How and Communication*. Springer-Verlag, Berlin, Germany, 1994. 35

O. A. Soares. A Realidade Construída – I. Master's thesis, Pontifícia Universidade Católica do Paraná, Curitiba, PR, Jul 2001. 29, 67

J. F. Sowa. *Conceptual Structures: Information Processing in Mind and Machine*. Addison-Wesley, Reading, MA, 1984. 7, 22

P. Stone. Layered Learning in Multi-Agent Systems. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, Dec 1998. 46

P. Stone and M. Veloso. Multiagent Systems: A Survey from a Machine Learning Perspective. *Autonomous Robots*, 8(3):345–383, Jul 2000. 38, 40, 42, 43

R. S. Sutton and A. G. Barto. Toward a Modern Theory of Adaptive Networks: Expectation and Prediction. *Psychological Review*, 88(2):135–170, 1981. 8

K. P. Sycara. Multiagent Systems. *AI Magazine*, 19(2), Jun 1998. 42, 45

M. Tambe, W. L. Johnson, R. M. Jones, F. Koss, J. E. Laird, P. S. Rosenbloom, and K. Schwamb. Intelligent Agents for Interactive Simulation Environments. *AI Magazine*, 16(1), 1995. 2, 54, 55, 62

L. G. Valiant. A Theory of the Learnable. *Communications of the ACM*, 27(11):1134–1142, Nov 1984. 11

S. A. Vere and T. W. Bickmore. A Basic Agent. *Computational Intelligence*, 6(1):41–60, Feb 1990. 39, 100

L. S. Vygotsky. O Instrumento e o Símbolo no Desenvolvimento da Criança. In M. Cole, V. John-Steiner, S. Scribner, and E. Souberman, editors, *A Formação Social da Mente – O Desenvolvimento dos Processos Psicológicos Superiores*, pages 33–40. Martins Fontes, São Paulo, SP, 6^{th} edition, 1998. Tradução de J. C. Neto, L. S. M. Barreto e S. C. Afeche. 17

L. S. Vygotsky, A. R. Luria, and A. N. Leontiev. Linguagem, Desenvolvimento e Aprendizagem. In J. Cipolla-Neto, L. S. Menna-Barreto, M. T. F. Rocco, and M. K. de Oliveira, editors, *Linguagem, Desenvolvimento e Aprendizagem*, Educação Crítica. Ícone, São Paulo, SP, 7^{th} edition, 2001. Tradução de M. da P. Villalobos. 18, 106

R. J. Waldinger. Achieving Several Goals Simultaneously. In E. W. Elcock and D. Michie, editors, *Machine Representations of Knowledge*, volume 8 of *Machine Intelligence*, pages 94–136. Ellis Horwood, Chichester, UK, 1977. 76

J. P. Walsh and G. R. Ungson. Organizational Memory. *The Academy of Management Review*, 16(1):57–91, 1991. 10

C. J. C. H. Watkins. *Learning from Delayed Rewards*. PhD thesis, University of Cambridge, Cambridge, UK, 1989. 8, 45

K. M. Wiig. *Knowledge Management Foundations – Thinking about Thinking – How People and Organizations Create, Represent and Use Knowledge*, volume 1 of *Knowledge Management*. Schema Press, Arlington, TX, 1993. 9, 10

K. M. Wiig. *Knowledge Management Methods – Practical Approaches to Managing Knowledge*, volume 3 of *Knowledge Management*. Schema Press, Arlington, TX, 1995. 9, 10, 13

T. Wittig, editor. *ARCHON: An Architecture for Multiagent Systems*. Ellis Horwood, Chichester, UK, 1992. 41

M. J. Wooldridge. Agent-based Software Engineering. In *IEE Proceedings on Software Engineering*, volume 144, pages 26–37, Feb 1997. 31

M. J. Wooldridge. Intelligent Agents. In G. Weiss, editor, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, chapter 1, pages 27–77. MIT Press, Cambridge, MA, 1999. 34

M. J. Wooldridge and P. Ciancarini. Agent-oriented Software Engineering: The State of the Art. In P. Ciancarini and M. Wooldridge, editors, *Proceedings of the* 1^{st} *International Workshop on Agent-oriented Software Engineering*, volume 1957 of *Lecture Notes in Computer Science*, pages 1–28, Berlin, Germany, 2001. Springer-Verlag. 31

M. J. Wooldridge and N. R. Jennings. Intelligent Agents: Theory and Practice. *Knowledge Engineering Review*, 10(2):115–152, Jun 1995. 34, 38, 39

M. J. Wooldridge and N. R. Jennings. Pitfalls of Agent-oriented Development. In *Proceedings of the* 2^{nd} *International Conference on Autonomous Agents*, pages 385–391, Minneapolis, MN, 1998. ACM Press. 36

M. J. Wooldridge and N. R. Jennings. Software Engineering with Agents: Pitfalls and Pratfalls. *IEEE Internet Computing*, 3(3):20–27, May 1999. 36

M. J. Wooldridge, N. R. Jennings, and D. Kinny. A Methodology for Agent-oriented Analysis and Design. In *Proceedings fo the* 3^{rd} *Annual Conference on Autonomous Agents*, pages 69–76, Seattle, WA, 1999. ACM Press. 41

M. J. Wooldridge, N. R. Jennings, and D. Kinny. The Gaia Methodology for Agent-oriented Analysis and Design. *Autonomous Agents and Multi-Agent Systems*, 3(3): 285–312, 2000. 31

L. A. Zadeh. From Computing with Numbers to Computing with Words: From Manipulation of Measurements to Manipulation of Perceptions. *IEEE Transactions on Circuits and Systems*, 45(1):105–119, 1999. 3, 12, 30

Apêndice A

Agente e Ação — O Conceito Original

O termo *agente* deriva do vocábulo em latim *agere*. Segundo o filósofo espanhol Mosterín, que comparou essa derivação e a forma primitiva do termo *fazer*, a palavra *agente* deveria ser reservada para entes executando atos intencionais. Segue um extrato de sua obra; esse trecho é intitulado *Nem todos os atos do homem são atos humanos*.

> "Deixando de lado alguns usos puramente técnicos da palavra *ação* — por exemplo, ação como participação no capital de uma empresa —, o núcleo significativo da palavra assenta na produção ou provocação de um efeito. A palavra *ação* é empregada, às vezes, para falar de seres não humanos — diz-se que a ação das cigarras é benéfica para a agricultura — ou, inclusive, de objetos inanimados — diz-se que a gravitação é uma forma de ação à distância ou que a toda ação exercida sobre um corpo corresponde uma reação igual e de sentido contrário. Porém, sobretudo, empregamos a palavra ação para nos referirmos ao que fazem os seres humanos. Aqui só nos interessa esse tipo de ação, ação humana.
>
> As nossas ações são (algumas das) coisas que fazemos. Na realidade, o verbo *fazer* cobre um campo semântico bem mais amplo que o substantivo *ação*. O latim distingue o *agere* (agir) do *facere* (fazer). Ao substantivo latino *actio*, derivado de *agere*, corresponde o substantivo *ação*. Assim, etimologicamente, *ação* só traz a carga semântica de *agere* enquanto que *fazer* traz tanto a de *agere* quanto a de *facere*.
>
> Tudo o que fazemos é parte da nossa conduta, mas nem tudo o que fazemos constitui uma ação. Enquanto dormimos, fazemos muitas coisas: respiramos, suamos, damos voltas na cama, apertamos a cabeça contra a almofada, sonhamos, talvez ressonemos alto, falemos em voz alta ou andemos sonâmbulos pela casa. Todas essas coisas fazemos inconscientemente, pois estamos adormecidos. Fazemos, mas não nos damos conta disso, não temos consciência de estar fazendo. A essas coisas que

fazemos inconscientemente não chamaremos *ações*. Vamos reservar o termo *ação* para as coisas que fazemos conscientemente, dando-nos conta de que as fazemos.

Há, no entanto, coisas que fazemos conscientemente, dando-nos conta delas, mas sem que à sua realização corresponda uma intenção nossa. Damo-nos conta dos nossos 'tiques' e de muitos dos nossos atos reflexos, contudo os realizamos involuntariamente, constatamo-los como espectadores, não os efetuamos como agentes — a palavra *agente* é outro resíduo do naufrágio do verbo *agere*. Pelo que sentimos depois de comer, damo-nos conta de que estamos fazendo uma boa digestão. Mas fazer a digestão não constitui (normalmente) uma ação. Pelos sorrisos dos que nos observam, damo-nos conta de que estamos sendo ridículos. Mas ser ridículo — praticar atos ridículos — não é uma ação, mas uma reação, algo que nos passa despercebido e que lamentamos — a não ser que o façamos de propósito, como provocação; nesse caso, já seria uma ação. Também não chamamos *ações* a esses aspectos da nossa conduta de que nos damos conta, mas que não efetuamos intencionalmente.

No presente estudo, limitar-nos-emos às ações humanas conscientes e voluntárias, às que daqui em diante chamaremos *ações* (sem mais). Uma ação é uma interferência consciente e voluntária de um ser humano (o agente) no normal decurso das coisas, que sem a sua interferência teriam seguido um caminho distinto do que aquele que, por causa da ação, seguiram. Uma ação consta, então, de um evento que sucede graças à interferência de um agente, agente este que tinha a intenção de interferir para conseguir que tal evento sucedesse".

Mosterín [Mosterín, 1987]

Apêndice B

Passagens sobre Aprendizado — Dois Psicólogos

Adaptações ocorrem como um reflexo das reações da mente a perturbações externas. Eventos inesperados causam perturbações[1] que devem ser trabalhadas pela memória a fim de que ocorrências posteriores de eventos com padrões semelhantes não venham a causar as mesmas perturbações. Essas perturbações são tão somente os estímulos sensoriais e as evoluções em memória por elas causadas, evoluções essas que podem vir a se tornar perturbações novas e dar origem a outras evoluções, e assim por diante. Dessa forma, percebe-se que a memória é um sistema em adaptação constante. "O desenvolvimento da criança é um processo temporal por excelência", advoga Piaget [1983b].

De acordo com Luria [2001], a visão da adaptação como um processo que reage a perturbações também era uma visão compartilhada por Vygotsky[2]. De acordo com sua visão, seres humanos reagem e adaptam-se a estímulos externos segundo um comportamento que evidencia um processo de acomodação às percepções do mundo; para Vigotsky, é dessa forma que se desenvolve a percepção infantil em relação ao mundo exterior.

Na visão de Piaget [1983a], "...o conhecimento não pode ser concebido como algo predeterminado nas estruturas internas do indivíduo, pois elas resultam de uma construção efetiva e contínua, nem nos caracteres pré-existentes no objeto, pois eles só são conhecidos graças à mediação necessária dessas estruturas".

[1] Em um certo sentido, todo evento possui ao menos um aspecto de perturbação, que está embutido em sua própria essência de evento — um acontecimento novo que produz alguma alteração no ambiente.

[2] Ao lado de Leontiev, Luria foi um dos discípulos de Vygotsky, trabalhando ao seu lado por aproximadamente uma década.

Embora divergissem em alguns pontos, o russo Vygotsky e o suíço Piaget trouxeram ambos inúmeras contribuições para a educação. Apesar de terem desenvolvido trabalhos concomitantes, Piaget só veio a tomar conhecimento dos escritos de Vygotsky, incluindo as críticas de Vygotsky a sua obra, muito depois da morte do russo[3], pois os trabalhos de Vygotsky só passaram a ser conhecidos no ocidente a partir de 1958. Segundo análise do próprio Piaget, seus trabalhos posteriores — após a morte de Vygotsky — concordavam muito mais com a visão de Vygotsky do que os primeiros, sobre os quais o russo elaborou suas críticas, mesmo quando o suíço ainda não tinha conhecimento dos trabalhos de Vygotsky. Piaget inclusive reconhece que algumas de suas descobertas deveriam, na verdade, serem remetidas ao russo, já que ele as havia previsto antes.

[3]Vygotsky faleceu muito prematuramente.