Luciano Reis Coutinho

A Modelagem do Aprendiz em Ambientes de Aprendizagem por Computador baseados em Atividades de Resolução de Problemas

Dissertação de Mestrado submetida à Coordenação do Curso de Pós-Graduação em Informática da Universidade Federal da Paraíba - Campus II como parte dos requisitos necessários para obtenção do grau de Mestre.

Evandro de Barros Costa, Doutor
Orientador

Área de Concentração: Inteligência Artificial

Campina Grande – PB
Maio de 1999

A MODELAGEM DO APRENDIZ EM AMBIENTES DE APRENDIZAGEM POR COMPUTADOR BASEADOS EM ATIVIDADES DE RESOLUÇÃO DE PROBLEMAS

LUCIANO REIS COUTINHO

DISSERTAÇÃO APROVADA EM 07.05.1999

PROF. EVANDRO DE BARROS COSTA , D.Sc
Orientador

PROF. ANGELO PERKUSICH, D.Sc
Examinador

PROF. GUILHERME BITTENCOURT, Dr.
Examinador

CAMPINA GRANDE – PB

C871m Coutinho, Luciano Reis
A modelagem do aprendiz em ambientes de aprendizagem por computador baseados em atividades de resolucao de problemas / Luciano Reis Coutinho. - Campina Grande, 1999.
155 f.

Dissertacao (Mestrado em Informatica) - Universidade Federal da Paraiba, Centro de Ciencias e Tecnologia.

1. Inteligencia Artificial 2. Modelagem do Aprendiz 3. Ambientes de Aprendizagem por Computador 4. MATHEMA 5. Dissertacao I. Costa, Evandro de Barros II. Universidade Federal da Paraiba - Campina Grande (PB)

CDU 004.8(043)

Aos meus pais.
À Bel.

Com Reconhecimento e Amor.

Agradecimentos

Toda pesquisa científica é antes de tudo um trabalho de grupo. Sendo assim, quero, com muita satisfação, deixar registrado os meus sinceros agradecimentos a todos que contribuíram direta ou indiretamente para que este trabalho de pesquisa fosse realizado.

Agradeço ao Prof. Doutor Evandro Costa, por sua orientação segura e constante incentivo. Agradeço aos meus colegas e professores do Grupo de IA. Em especial a Josenildo, pelas inúmeras e valiosas discussões sobre os agentes tutores MATHEMA, e ao prof. Doutor Edilson Ferneda por sua atenção e apoio prestados à minha pesquisa. Agradeço também a todos os demais colegas ligados ou não ao mestrado. Em especial a Betânia, Rute e Sandro, que me deram inestimável apoio e estímulo ao longo de meu caminho, e à turma do TIMÃO-L, pelas agradáveis horas de descontração e lazer. Agradeço ainda aos funcionários do DSC e COPIN. Em especial ao pessoal da MINIBLIO, pela presteza e atenção com que atenderam às minhas solicitações de livros, e às secretárias da COPIN, Vera Lúcia e Aninha, pela atenção e boa vontade que dispensaram a mim.

Agradeço à CAPES, pelo suporte financeiro a realização deste trabalho de pesquisa.

Resumo

O objetivo desta dissertação é especificar o processo de modelagem do aprendiz para os agentes tutores da arquitura MATHEMA. A arquitetura MATHEMA é uma estrutura conceitual para o desenvolvimento de ambientes de aprendizagem por computador baseados em atividades de resolução de problemas. A especificação é feita segundo duas linhas de ação. Primeiro, o estabelecimento de direções gerais para a modelagem do aprendiz para os agentes MATHEMA. Tais direções resumem que informações sobre os aprendizes podem ser modeladas; como representar, adquirir e manter estas informações em um modelo do aprendiz; e principalmente, que decisões um agente MATHEMA pode fazer tendo por base um modelo do aprendiz. Segundo, a definição de um Sistema de Modelagem do Aprendiz particular para os agentes MATHEMA. Este Sistema de Modelagem do Aprendiz é especificado por meio da linguagem de especificação formal VDM-SL. Suas características principais são: (1) a representação do modelo do aprendiz feita por sobreposição do *Curriculum* presente no Sistema Tutor dos agentes MATHEMA; (2) a aquisição de informações feita por um processo de análise de passos de resolução de problemas; (3) a manutenção do modelo do aprendiz feita por um processo de manutenção de sobreposição difusa. Por fim, é mostrado como as informações representadas, adquiridas e mantidas pelo Sistema de Modelagem do Aprendiz podem efetivamente ser utilizadas pelo Sistema Tutor dos agentes MATHEMA durante a escolha de problemas a serem passados a um aprendiz.

Abstract

The aim of this dissertation is to specify the learner modeling process for tutor agents of the MATHEMA architecture. MATHEMA architecture is a framework for the development of computer learning environments based on problem solving activities. The specification is done following two lines of action. First, the establishment of general directions for learner modeling in MATHEMA agents. Such directions summarize what informations about the learners can be modeled; how to represent, acquire and maintain these informations in learner model; and mainly, what decisions can be made by MATHEMA agents having a student model. Second, the definition of a particular Learner Modeling System to be used in MATHEMA agents. This Learner Modeling System is specified by means of VDM-SL formal specification language. Its main features are: (1) the learner model representation done by an overlay of MATHEMA agents Tutor System *Curriculum*; (2) the acquisition of informations done by a model tracing process based on problem resolution; (3) the student model maintenance done by a maintenance process on fuzzy overlay models. Finally, it is shown how the Learner Model System represented, acquired and maintained informations can effectively be used by MATHEMA agents Tutor System during selection of problems to be passed to learners.

Conteúdo

Lista de Figuras

Capítulo 1

Introdução

1.1 Contexto

O assunto tratado nesta dissertação se insere na área de pesquisa denominada *Inteligência Artificial na Educação* – a IA-Ed [SE89].

A IA-Ed de forma breve pode ser definida como o uso de técnicas e teorias da *Inteligência Artificial* na concepção de sistemas computacionais "inteligentes" cujo objetivo é promover o processo de *aprendizagem humana.*

1.1.1 Os Ambientes de Aprendizagem Adaptativos

Uma das motivações fundamentais da pesquisa em IA-Ed é o desenvolvimento de meios através dos quais sistemas de computador possam ser concebidos como ambientes onde aprendizes possam ter experiências de aprendizagem individualizada, i.e., experiências que sejam significativas e benéficas a despeito das diferenças individuais, experiências anteriores, ou estados cognitivos dos aprendizes [AS95]. Chamaremos esta motivação de *problema da adaptação* [Cos97], e os ambientes computacionais desenvolvidos a partir do problema da adaptação de *Ambientes de Aprendizagem Adaptativos.*

Um desses meios são os chamados *sistemas de modelagem do aprendiz.*

1.1.2 A Modelagem do Aprendiz

Um *sistema de modelagem do aprendiz* é o componente de um Ambiente de Aprendizagem Adaptativo que o habilita a realizar um processo de modelagem do aprendiz. Ou seja, adquirir e manter um conjunto de informações sobre um dado aprendiz durante o processo de aprendizagem. Informações essas a respeito de alguns aspectos tais como o estado de conhecimento do aprendiz, seus planos, suas aptidões, suas preferências e

motivações. A estrutura na qual um Sistema de Modelagem do Aprendiz representa e mantem informações sobre os aprendizes é chamada de *modelo do aprendiz.*

A modelagem do aprendiz em Ambientes de Aprendizagem Adaptativos é uma área de pesquisa muito vasta. Sua origem remonta a início da década de 1970 quando Carbonell desenvolveu o sistema SCHOLAR, o primeiro ambiente computacional de aprendizagem com um sistema de modelagem do aprendiz [Car70]. De lá para cá muito foi pesquisado e atualmente há inúmeras técnicas e recursos computacionais para a aquisição, representação e manutenção de modelos do aprendiz.

1.1.3 A arquitetura MATHEMA

A *arquitetura MATHEMA* é uma estrutura conceitual para o desenvolvimento de ambientes de aprendizagem por computador baseados em atividades de resolução de problemas [Cos97]. Como uma estrutura conceitual, a arquitetura MATHEMA tem como proposta principal prover uma arquitetura padrão e um conjunto de princípios e ferramentas que orientem o desenvolvimento de ambientes computacionais onde estudantes possam aprender a partir de atividades de resolução de problemas.

Um ambiente de aprendizagem construído segundo a arquitetura MATHEMA será formado, dentre outros elementos, por uma *Sociedade de Agentes Tutores Artificiais.* Ou seja, uma coleção de *agentes tutores*: cada agente tutor sendo um módulo de *software* cujo objetivo é interagir com um aprendiz a fim de orientá-lo e ajudá-lo a aprender parte de um domínio de conhecimento.

1.2 Objetivos

No estado atual de desenvolvimento da arquitetura MATHEMA a definição da estrutura interna dos agentes tutores prevê a utilização de um Sistema de Modelagem do Aprendiz como meio de adaptar as ações dos agentes tutores às necessidades e características de cada aprendiz. No entanto, os componentes deste Sistema de Modelagem do Aprendiz ainda não foram especificados. Ainda não foi dito explicitamente quais informações sobre os aprendizes podem ser modeladas; como representar, adquirir e manter estas informações em um modelo do aprendiz; e, principalmente que decisões um agente tutor MATHEMA pode tomar tendo por base um modelo do aprendiz. Isso, em suma, é o objetivo geral de nosso trabalho.

Dividimos esse objetivo geral em dois outros um pouco mais específicos que melhor explicam nossas pretensões ao longo do texto:

(1) *Estabelecer direções gerais para a modelagem do aprendiz na arquitetura MATHEMA.* Uma vez que a modelagem do aprendiz é uma área de pesquisa muito vasta com inúmeras possibilidades e recursos computacionais já desenvolvidos, o objetivo aqui é identificar dentre as idéias correntes quais as que se aplicam à modelagem do aprendiz a ser realizada por um agente tutor MATHEMA.

(2) *Propor um Sistema de Modelagem do Aprendiz para os agentes tutores MATHEMA.* Uma vez tendo estabelecido direções gerais para a modelagem do aprendiz nos agentes tutores MATHEMA, o objetivo aqui é definir os elementos arquiteturais de um sistema de modelagem do aprendiz particular que possa ser utilizado em ambientes de aprendizagem construídos segundo a arquitetura MATHEMA.

Na consecução de nossos objetivos iremos assumir que cada agente tutor da Sociedade de Agentes Tutores Artificiais realiza a modelagem do aprendiz independentemente de outros agentes tutores. Esta é uma simplificação que fazemos para tornar nosso trabalho realizável dentro das restrições de espaço e tempo impostas por um curso de Mestrado. Conforme veremos nos próximos capítulos da dissertação, há muito o que fazer e que, uma vez feito, é o início para que possamos pensar na modelagem do aprendiz de forma cooperativa pelos agentes tutores MATHEMA.

Com relação ao segundo item acima, o Sistema de Modelagem do Aprendiz que iremos propor será definido por meio de uma linguagem de especificação formal – a linguagem VDM-SL [LHB+96, ABH+95]. A especificação do Sistema de Modelagem do Aprendiz bem como as razões para uma abordagem formal encontram-se no Capítulo 5.

1.3 Relevância

A principal contribuição de nosso trabalho é enriquecer e detalhar a arquitetura MATHEMA no tocante ao processo de modelagem do aprendiz. Como já dissemos, a arquitetura MATHEMA tem como proposta principal prover uma arquitetura padrão e um conjunto de princípios e ferramentas que orientem o desenvolvimento de ambientes de aprendizagem por computador. Dessa forma, nosso trabalho contribui para:

- Enriquecer os princípios da arquitetura MATHEMA: notemos que as direções gerais para a modelagem do aprendiz na arquitetura MATHEMA (item (1) acima) têm como principal objetivo servir como um conjunto de princípios e orientações que irão guiar o projeto e implementação de Sistemas de Modelagem do Aprendiz em ambientes de aprendizagem MATHEMA.

- Detalhar a arquitetura MATHEMA: notemos que o Sistema de Modelagem do Aprendiz que iremos propor (item (2) acima) objetiva ser uma visão detalhada de um componente previsto na definição original da estrutura interna dos agentes tutores MATHEMA.

De forma mais geral, nosso trabalho pode ser visto como uma contribuição para a área de modelagem do aprendiz. Para estabelecer direções gerais para a modelagem do aprendiz na arquitetura MATHEMA, iremos fazer um amplo levantamento das idéias e abordagens existentes para a modelagem do aprendiz. Este levantamento, além de servir de guia para a modelagem do aprendiz na arquitetura MATHEMA, pode ser utilizado por outros pesquisadores, trabalhado em outros projetos, para decidir como realizar o processo de modelagem do aprendiz.

1.4 Organização do texto

O texto está organizado em cinco partes.

A primeira parte é esta introdução onde motivamos e apresentamos ao leitor os objetivos que perseguimos ao longo de nossa pesquisa.

A segunda parte é composta de dois capítulos: os Capítulos 2 e 3. No Capítulo 2 exploramos a idéia de ambientes de aprendizagem por computador e, neste contexto, apresentamos os elementos principais da arquitetura MATHEMA. No Capítulo 3 revisamos várias idéias e abordagens para a modelagem do aprendiz em ambientes de aprendizagem por computador. Ao final deste capítulo teremos formado uma estrutura de trabalho a ser utilizada nos capítulos seguintes durante a especificação da modelagem do aprendiz na arquitetura MATHEMA.

A terceira parte é formada por três capítulos: os Capítulo 4, 5 e 6. É a parte central da dissertação. No Capítulo 4 estabelecemos direções gerais para a modelagem do aprendiz nos agentes MATHEMA. As direções apresentadas são em essência um resumo da revisão do Capítulo 3 tendo em vista os elementos da arquitetura dos agentes tutores MATHEMA discutidos no Capítulo 2. No Capítulo 5, seguindo algumas das direções apontadas no Capítulo 4, apresentamos uma proposta de Sistema de Modelagem do Aprendiz para os agentes tutores MATHEMA. O Sistema de Modelagem do Aprendiz proposto é especificado por meio da linguagem VDM-SL. No Capítulo 6 mostramos como o Modelo do Aprendiz do Sistema de Modelagem do Aprendiz especificado no

Capítulo 5 pode ser utilizado pelos agentes MATHEMA. A utilização básica é a escolha de problemas adequados a serem passados a um dado aprendiz.

A quarta parte é formada pelo Capítulo 7. Neste capítulo apresentamos as conclusões que chegamos ao final da dissertação.

A quinta e última parte é formada por dois Apêndices e a Bibliografia. No Apêndice A apresentamos um resumo da sintaxe da linguagem de especificação formal VDM-SL. Este resumo é necessário pois não assuminos que o leitor tenha conhecimento prévio da linguagem VDM-SL. No Apêndice B apresentamos demonstrações de algumas proposições estabelecidas durante a especificação formal do Sistema de Modelagem do Aprendiz apresentada nos Capítulo 5 e 6.

Capítulo 2

A Arquitetura MATHEMA

2.1 Introdução

Neste capítulo revisamos a noção de *ambiente de aprendizagem por computador* e apresentamos a *arquitetura MATHEMA* – um modelo conceitual para o desenvolvimento de ambientes de aprendizagem por computador baseados em atividades de resolução de problemas.

O capítulo está organizado da seguinte forma. Iniciamos, na seção 2.2, falando sobre os ambientes de aprendizagem por computador. Mostramos uma classificação geral de tais ambientes e introduzimos alguns conceitos básicos tais como tutor artificial, modelo de domínio e modelo do aprendiz. A seguir, nas seções 2.3 - 2.5, apresentamos a arquitetura MATHEMA. Mostramos os componentes básicos de um ambiente de aprendizagem por computador construído segundo a arquitetura MATHEMA, a estruturação e organização do domínio de aprendizagem, e a estrutura geral dos *agentes tutores artificiais* MATHEMA. Ao final, na seção 2.6, situamos o trabalho de pesquisa veiculado nesta dissertação no âmbito da arquitetura MATHEMA.

2.2 A Aprendizagem por Computador

Os **ambientes de aprendizagem por computador**, de forma geral, podem ser definidos como sistemas de computador cujo objetivo é auxiliar ou promover a aprendizagem humana.

A aprendizagem humana é um fenômeno complexo, não bem definido. *O que é aprender? Como se aprende?* Eis duas questões difíceis, controversas, e muito importantes pois influenciam de forma direta a concepção de qualquer sistema de computador

cujo objetivo é auxiliar ou promover a aprendizagem humana. A busca de respostas a essas questões básicas tem levado os pesquisadores e projetistas de ambientes de aprendizagem por computador a abraçar determinadas *teorias psicológicas* e *filosofias educacionais* relativas ao intricado fenômeno da aprendizagem humana.

A teoria Comportamentalista

A primeira, historicamente, foi a teoria *comportamentalista* (*behaviorist theory*) de Skinner. A teoria comportamentalista esteve em voga nas décadas de 50-60. Esta teoria relacionava conhecimento com comportamento observado e aprendizagem com um processo de reforço de comportamentos desejados através de estímulos e respostas. Tal teoria levou a construção dos primeiros sistemas de computador para auxiliar a aprendizagem humana, os sistemas de *Instrução Assistida por Computador* – os IAC [PPS87]. Nestes sistemas a aprendizagem era promovida através do *ensino direto intercalado com breves sessões de exercícios e prática*: o material a ser ensinado era estruturado em torno de um *curriculum* e o aprendiz era conduzido através de tópicos seguindo um dentre alguns caminhos de ensino pré-estabelecidos.

As teorias Cognitivas

Nas décadas de 70 e 80 as idéias comportamentalistas incorporadas nos IAC foram gradualmente substituídas por teorias *cognitivas de aprendizagem*. Ao contrário da psicologia comportamentalista, que relacionava aprendizado com comportamentos externos observáveis, as teorias cognitivas de aprendizagem passaram a advogar a idéia de que conhecimento e aprendizado têm relação direta com estruturas e processos mentais "internos", não observáveis. Tais estruturas e processos sendo considerados a causa dos comportamentos observados. Essas novas idéias levaram a concepção dos *Sistemas Tutores Inteligentes* – os STI [Wen87].

Os STI, baseando-se nas idéias cognitivas, tinham como principal característica a compilação e representação explícita do conhecimento a ser aprendido, ou seja das estruturas e processos mentais que o aluno[1] deveria adquirir. Esta representação de conhecimento chamava-se *modelo do domínio*. Dado um modelo do domínio, a aprendizagem nos STI em geral ocorria através de ensino interativo intercalado com atividades de resolução de problemas. Essas atividades de aprendizagem, ao contrário do que ocorria com os IAC, eram planejadas durante a interação do sistema com os apren-

[1] Ao longo do texto usaremos os termos *aprendiz*, *aluno* e *estudante* como sinônimos para nos referirmos aos usuários de um ambiente de aprendizagem por computador. Daremos, no entanto, preferência ao termo *aprendiz* ao falarmos de *modelo do aprendiz* e *modelagem do aprendiz*.

dizes. O planejamento era realizado tendo como guia dois outros modelos, um *modelo pedagógico* e um *modelo do aprendiz*. No modelo pedagógico estavam codificadas as estratégias instrucionais, i.e., detalhes a respeito dos modos básicos de interação com o aprendiz suportados pelo sistema. No modelo do aprendiz estavam armazenadas *informações úteis sobre o aprendiz*, em especial, o seu *estado de conhecimento*. Dispondo de um modelo pedagógico e um modelo do aprendiz os STI eram capazes de (a) analizar o estado de conhecimento do aprendiz; (b) descrever o estado de conhecimento desejável; e (c) apresentar material e problemas para promover a transição no estado de conhecimento do aprendiz [SP95].

Juntas, a teoria comportamentalista e as teorias cognitivas usadas na construção dos IAC e STI, apresentavam em comum o fato de verem a aprendizagem humana como um processo que ocorre primariamente através de transmissão, ensino ou instrução de um corpo de conhecimento pré-estabelecido e aceito como "correto". Esta concepção por vezes tem sido chamada de *filosofia instrucionista* da aprendizagem ou *instrutivismo*[2]; [Rie92] apud [Was96]. Opondo-se ao instrutivismo, ou antes, tirando o ensino ou instrução do foco das atenções na concepção de ambientes de aprendizagem por computador, encontra-se a *filosofia construtivista* da aprendizagem.

A filosofia Construtivista

A filosofia construtivista de aprendizagem não é um ponto de vista unificado sobre o que é aprender e como se aprende. Segundo [Win93] apud [Was96], sob o nome *construtivismo* encontra-se uma ampla gama de idéias, pontos de vista e posições teóricas tendo como ponto de interseção a idéia de que a aprendizagem é um processo de *construção* de significados (conhecimentos) onde o aprendiz deve ser um agente ativo em todo o processo. Essas idéias, empregadas na construção de ambientes de aprendizagem por computador, fizeram surgir duas categorias de sistemas, os *Ambientes de Aprendizagem por Descoberta/Exploração* – AAD [LD93a, Rie94, Wil96]; e os *Ambientes de Aprendizagem Interativos* – AAI [Gia92, JW92].

Os AAD não são uma proposta nova. Surgiram com o nome de *micromundos* entre as décadas de 70-80 [Pap80, Tho87]. Os micromundos propunham uma abordagem radicalmente oposta à idéia de aprendizagem por ensino incorporada nos sistemas de IAC e STI. Em um micromundo não havia conhecimento "correto" a ser transmitido, comunicado ao aprendiz. Ao aprendiz era dado um ambiente e um conjunto de ferramentas com as quais poderia explorar o ambiente e construir seu próprio con-

[2]Tradução livre do termo inglês *instructivism*

hecimento. Na década de 90, com o aparecimento de novas tecnolologias tais como terminais gráficos de alta resolução, interfaces multimídia, hipermídia e equipamentos de realidade virtual os AAD ganharam nomes e roupagem novos. Surgiram então os *mundos de exploração virtuais* [Wil92], os *livros eletrônicos* [Bar92], os *ambientes de aprendizagem por imersão* [SP95], e outros.

As novas tecnologias, no entanto, não trouxeram apenas novo vigor às idéia de aprendizagem por descoberta e exploração. Ressaltaram um sério problema com os AAD. Dar ao aprendiz um rico ambiente e ferramentes nem sempre é a receita certa para tornar o processo de aprendizagem efetivo. Os alunos em geral necessitam de certo grau de supervisão ou orientação. Dessas conclusões surgiram os AAI. Ou melhor, dessas conclusões em conjunto com o relativo fracasso, em fins da década de 80, da pesquisa em STI em cumprir o que por duas décadas foi prometido: promover e melhorar a qualidade da aprendizagem por computador através de ambientes de aprendizagem por ensino[3]. Assim, os AAI afiguram-se como uma versão melhorada dos AAD e/ou uma evolução natural dos STI. *O melhor de dois mundos.*

A filosofia Sócio-Cultural

Há ainda um terceiro mundo. A interpretação sócio-cultural da aprendizagem humana. Esta abordagem enfatiza o papel do relacionamento social e da cultura como fatores decisivos no aprendizado humano. A aprendizagem antes analisada apenas sob um ponto de vista individual deve agora ser revista, reavaliada a partir de uma perspectiva coletiva; esta é a opinião de alguns pesquisadores influenciados pela abordagem sócio-cultural. "Em certo sentido [aprender] é uma experiência social: alguma pessoa – colega, tutor, manual do autor – está envolvido tanto quanto o aprendiz."[JM93, p.20] apud [Was96].

Essas idéias em contato com as idéias de aprendizagem por ensino e aprendizagem por descoberta já discutidas, deram origem, entre outros desdobramentos[4], ao conceito de *Ambientes de Aprendizagem Colaborativos* – os AAC [Cha95, DBBO94]. Há duas variantes principais de AAC. Na primeira o computador é visto como um colabo-

[3] A seguinte observação de John Self é ilustrativa: *"Ao fim da década de 80, a pesquisa sobre sistemas tutores inteligentes desenvolveu uma reputação de ter falhado em distribuir o que prometeu ou, mais precisamente, o que foi prometido a seu favor (o acronimo STI foi algumas vezes interpretado como 'sistemas tutores invisíveis')"* [Sel95a, sec. 3.1]

[4] Wasson [Was96] cita, além da aprendizagem colaborativa, mais duas abordagem para a construção de ambientes de aprendizagem por computador que sofrem fortes influências da interpretação sócio-cultural: *aprendizado cognitivo* (*cognitive apprenticeship*: [LL89]), *aprendizagem situada* (*situated learning*: [Cla95]).

rador, um *parceiro virtual* [CB90]. Alguém com quem o aprendiz irá interagir e juntos aprender. Na segunda variante o computador é visto como um meio, um suporte, um ambiente onde aprendizes e professores possam se comunicar, colaborar, cooperar e juntos adquirir conhecimentos [LF98].

Em suma

Os ambientes de aprendizagem por computador têm sido construídos tendo como fundamento três grandes correntes teórico-filosóficas a respeito do complexo fenômeno da aprendizagem humana: a filosofia instrucionista (aprendizagem por ensino, instrução), a filosofia construtivista (aprendizagem por descoberta, exploração) e a filosofia sócio-cultural (aprendizagem em contextos sociais, colaborativa). Estas três linhas de pensamento combinadas dão origem a uma ampla gama de sistemas retratados de forma esquemática na Figura 2.1.

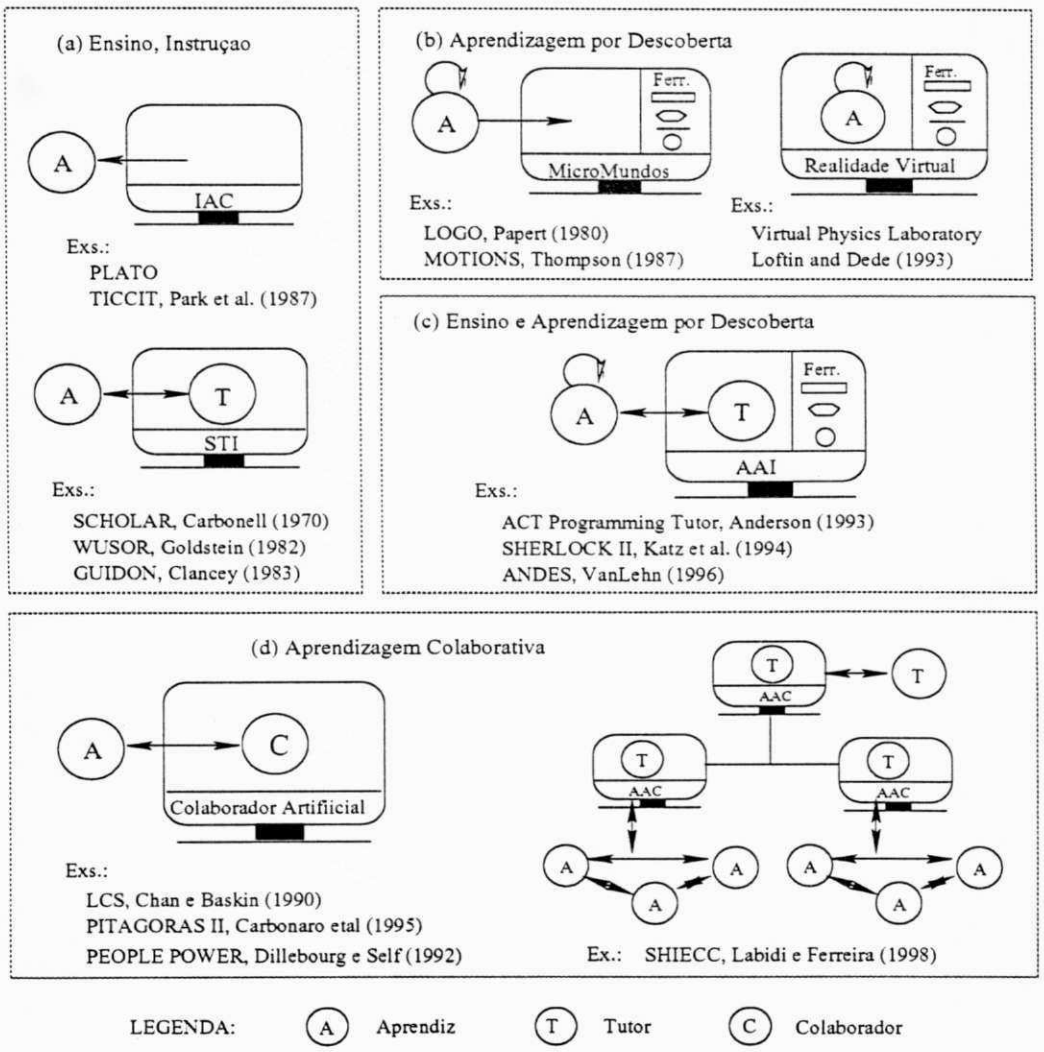


Figura 2.1: Os Ambientes de Aprendizagem por Computador

2.2.1 Os Ambientes de Aprendizagem Adaptativos

Um *ambiente de aprendizagem adaptativo* é um ambiente de aprendizagem por computador onde o *aprendiz* recebe instrução, ajuda ou supervisão individualizada, ou seja, de acordo com suas necessidades ou características individuais. Por esse critério, identificamos, entre os ambientes de aprendizagem por computador apresentados, duas categorias principais de ambientes adaptativos: os *Sistemas Tutores Inteligentes* (STI) e os *Ambientes de Aprendizagem Interativos* (AAI).

Os STI, como discutimos, surgiram na mesma linha de pensamento dos antigos sistemas IAC – a aprendizagem por instrução. No entanto, em relação aos IAC que apresentavam uma dinâmica de ensino rígida (pré-programada), os STI inovaram adotando uma postura de ensino mais flexível e adaptável as necessidades e características dos aprendizes.

Os AAI surgiram como uma evolução dos STI e ao mesmo tempo como um melhoramento dos Ambientes de Aprendizagem por Descoberta (AAD). Em relação aos STI, os AAI inovaram provendo aos aprendizes ambientes computacionais onde estudantes passaram a aprender não apenas por ensino mas também por descoberta e exploração. Em relação aos AAD, os AAI inovaram provendo ambientes computacionais de aprendizagem por descoberta mais individualizados e adaptados aos aprendizes.

O tutor artificial

Em comum, os STI e AAI incorporam a idéia de um *tutor artificial.* Chamamos de tutor artificial o componente presente em ambientes de aprendizagem por computador responsável por orientar e supervisionar o processo de aprendizagem. Retratamos o tutor artificial na Figura 2.1 através de um círculo nomeado T. Um tutor artificial que orienta e supervisiona levando em consideração as necessidades e peculiaridades dos aprendizes chamamos de *tutor adaptativo.*

Um dos resultados principais da pesquisa em STI é a definição de uma arquitetura básica para a construção de tutores artificiais. Esta arquitetura, já comentada anteriormente, é formada por três modelos – o *modelo do domínio*, o *modelo pedagógico* e o *modelo do aprendiz.* Estes modelos constituem respectivamente o conhecimento do tutor articial sobre o que está sendo estudado (domínio de aprendizagem), como interagir e orientar os aprendizes (estratégias pedagógicas), e quem são os aprendizes (suas necessidades e características). *Destes três modelos o modelo do aprendiz é o componente fundamental em um tutor adaptativo*[5].

[5]Pelo menos no âmbito desta dissertação. Alguns pesquisadores argumentam que na construção

2.3 A arquitetura MATHEMA

A arquitetura MATHEMA é uma estrutura conceitual para o desenvolvimento de ambientes de aprendizagem por computador baseados em atividades de resolução de problemas. Como uma estrutura conceitual, a arquitetura MATHEMA tem como proposta principal prover uma *arquitetura padrão* – composta, dentre outros elementos, por uma *Sociedade de Agentes Tutores Artificiais* – e um conjunto de *princípios e ferramentas* que orientem o desenvolvimento de Ambientes Interativos de Aprendizagem.

2.3.1 Os Ambientes de Aprendizagem MATHEMA

Um ambiente de aprendizagem por computador construído a partir da arquitetura MATHEMA terá uma estrutura formada por três elementos básicos [Cos97, pp.48-50]. São eles:

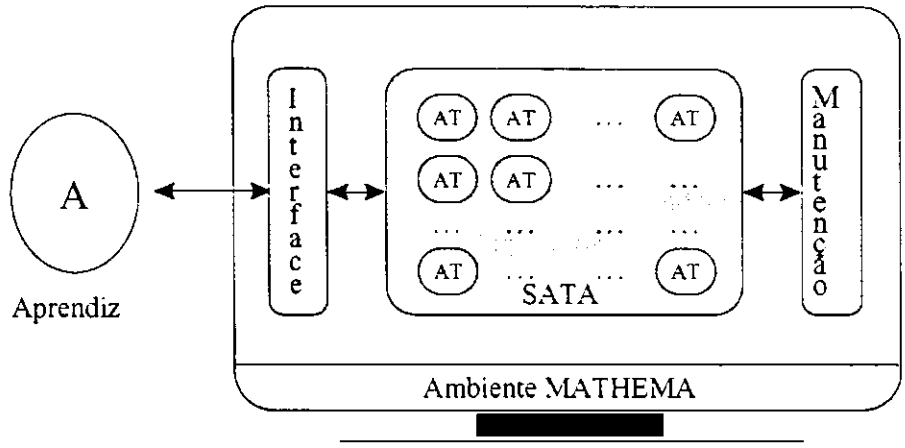


Figura 2.2: O ambiente computacional MATHEMA

- a *Interface* : "local", ambiente computacional onde o aprendiz irá realizar atividades de resolução de problemas e através da qual entrará em contato com *agentes tutores artificiais*:
- a *SATA*: *Sociedade de Agentes Tutores Artificiais.*
- o *Ambiente de Manutenção*: ferramenta utilizada pelos projetistas do ambiente de aprendizagem para incluir, retirar e/ou alterar agentes tutores da SATA.

de tutores adaptativos: *"o que é necessário são melhores modelos cognitivos ou descrições da aprendizagem e ensino que possam ser codificadas em tutores baseados em conhecimento. Trabalhos sobre a modelagem de bons professores e conhecimento especializado podem compensar modelos do aprendiz mais simples."* [Woo92] apud [Mur97a]

2.3.2 A SATA

Dos três componentes de um ambiente de aprendizagem MATHEMA, a SATA desempenha um papel central. Em linhas gerais, a SATA pode ser conceituada como um *sistema multiagentes*. Quer dizer, uma coleção de tutores artificiais chamados de *agentes tutores* MATHEMA; cada agente tutor sendo um módulo de *software* autônomo[6], orientado por objetivos e capaz de comunicação e cooperação com outros agentes tutores [Cos97, cap.5]. O papel da SATA é interagir com um aprendiz a fim de orientá-lo e ajudá-lo a aprender determinado *domínio de aprendizagem.*

Na arquitetura MATHEMA a aprendizagem é assumida "como sendo favorecida e decorrente de atividades provenientes do processo de resolução de problemas"[Cos97, p.47]. Desta forma, toda interação entre um aprendiz e a SATA tem como início a *resolução de problemas.* A interação de um aprendiz com a SATA é feita *um a um* – em dado instante um aprendiz interage com um único agente tutor – sendo cada agente tutor um especialista em uma parte específica do domínio de aprendizagem. No Capítulo 4 discutiremos em mais detalhes a interação dos agentes tutores com aprendizes em situações de resolução de problemas.

A construção da SATA

A SATA é construída tendo como guia dois princípios ou passos básicos:

1. A decomposição e organização do domínio de aprendizagem alvo em subdomínios de aprendizagem utilizando uma abordagem multidimensional;
2. A definição e construção dos agentes tutores MATHEMA a partir dos subdomínios de aprendizagem identificados no passo anterior.

A seguir discutimos esses dois passos.

2.4 O modelo de domínio multidimensional

A construção de um ambiente de aprendizagem MATHEMA inicia com a visualização do domínio de aprendizagem utilizando uma abordagem multidimensional. Nesta abordagem o conhecimento a ser aprendido é estruturado e organizado sob dois pontos de vista: a *visão externa* e a *visão interna* [Cos97, pp.37-44].

[6]Ou seja, que tem objetivos próprios, existência independente da existência de outros agentes tutores, e é capaz de agir no sentido de atingir seus objetivos através de seus próprios meios, tendo total controle sobre seu estado interno e ações.

2.4.1 Visão Externa

Na visão externa o conhecimento a ser aprendido é dividido em *subdomínios de aprendizagem.* A divisão é orientada pela combinação de duas dimensões principais[7] : a dimensão de *contexto* e a dimensão de *profundidade.* Com relação à primeira dimensão, o domínio de aprendizagem é classificado segundo uma série de contextos distintos, onde cada contexto corresponde a uma abordagem diferente na qual o domínio de aprendizagem pode ser estudado. Com relação à segunda dimensão, o conhecimento em cada contexto identificado é visto em diversas profundidades. Cada qual correspondendo a um nível de dificuldade ou complexidade no qual o conhecimento de cada contexto pode ser apresentado. Um subdomínio de aprendizagem d_{ij} de um domínio de aprendizagem D é então definido combinando um contexto i, e uma profundidade j do domínio de aprendizagem D. Ver Figura 2.3.

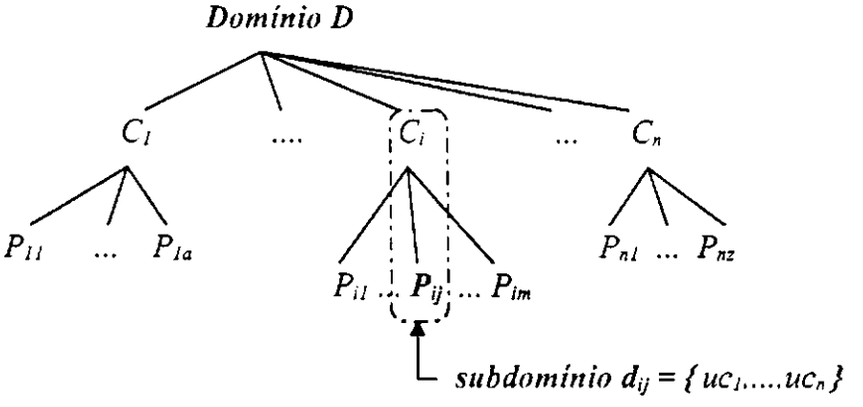


Figura 2.3: Visão multidimensional de um domínio de aprendizagem D

Exemplo

Vejamos um exemplo. O domínio de aprendizagem é *Estudo da Reta.* Em livros didáticos do 2º grau, tais como [Buc92], o estudo da reta é feito em dois contextos distintos. O primeiro consiste em estudar as características e propriedades da reta no contexto da *Geometria de Posição.* O segundo consiste em estudar a reta no contexto da *Geometria Analítica.* No contexto da Geometria de Posição identificamos apenas uma profundidade que chamaremos de estudo da reta no Espaço Euclidiano. No contexto da Geometria Analítica identificamos duas profundidades. O estudo da reta no Plano Cartesiano e, o estudo da reta no Espaço Euclidiano. Um subdomínio do domínio

[7] Há uma terceira dimensão, a *lateralidade.* Em nossa apresentação ela foi omitida por não ser necessária à explicação de nossos objetivos.

de aprendizagem Estudo da Reta é então o *Estudo da Reta no contexto da Geometria Analítica restrito ao (profundidade) Plano Cartesiano.* Ver Figura 2.4.

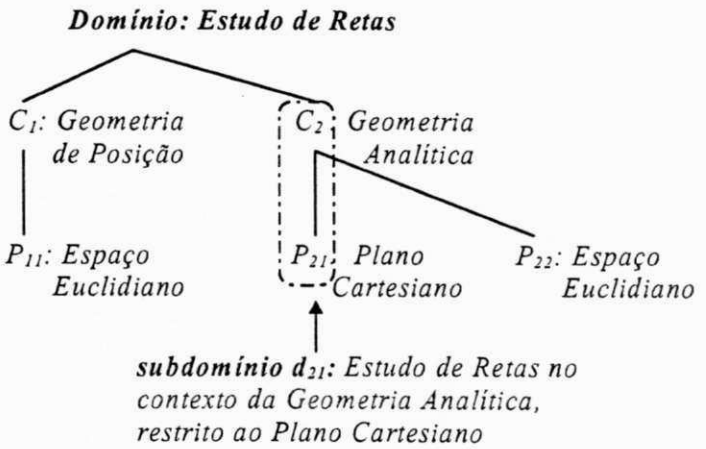


Figura 2.4: Visão multidimensional do domínio de aprendizagem Estudo de Retas

2.4.2 Visão Interna

Na visão interna, cada subdomínio d_{ij} identificado na visão externa é internamente organizado formando uma *estrutura pedagógica.* A estrutura pedagógica é composta por dois elementos principais: o *curriculum* e o *domínio de problemas.* Ver Figura 2.5.

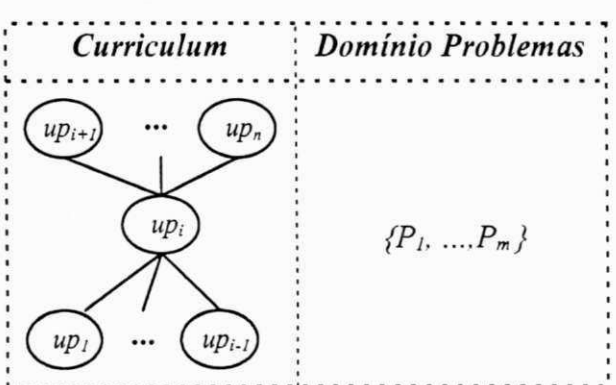


Figura 2.5: Estrutura pedagógica de um subdomínio d_{ij}.

Curriculum

O *curriculum* é uma organização do conhecimento presente em cada subdomínio d_{ij} que visa estabelecer possíveis *seqüências* nas quais o conteúdo do subdomínio pode ser

aprendido. O primeiro passo na definição de um *curriculum* é particionar o subdomínio em *unidades pedagógicas*. Uma unidade pedagógica é um conjunto de *unidades de conhecimento* relativas ao subdomínio que, segundo critérios educacionais, formam uma unidade de aprendizagem dentro do subdomínio. Uma unidade de conhecimento são basicamente *conceitos* e *habilidades* que um aluno irá aprender. O segundo passo é estabelecer um ordenamento entre as unidades pedagógicas identificadas segundo critérios de pré-requisitos e/ou níveis de dificuldade. Na Figura 2.5 mostramos um fragmento de um *curriculum* representado através de um *diagrama de Hasse*[8]. Os pontos nomeados por $up_{1...n}$ representam algumas das unidades pedagógicas nas quais o subdomínio foi particionado. A ordem implícita no diagrama nos diz que a unidade pedagógica up_i tem como pré-requisitos todas as unidades pedagógicas $up_{1...i-1}$; da mesma forma, up_i é um pré-requisito para os conhecimentos presentes nas unidades pedagógicas $up_{i+1...n}$; as unidades pedagógicas no mesmo nível não mantém relação de pré-requisito.

Por fim – tendo definido um *curriculum* – *seqüências de aprendizagem* para as unidades pedagógicas do *curriculum* podem ser definidas. Um seqüência de aprendizagem é uma alguma seqüência na qual as unidades pedagógicas podem ser dispostas (para estudo) respeitando a ordem de pré-requisitos/dificuldades imposta pelo *curriculum*.

Domínio de Problemas

O domínio de problemas é uma uma catalogação dos problemas relativos a um subdomínio d_{ij} que serão utilizados durante o estudo do subdomínio em questão. Um problema relativo a um subdomínio d_{ij} é qualquer problema que envolva conhecimento presente em uma ou mais unidades pedagógicas do *curriculum* estabelecido para d_{ij}. Na definição de um domínio de problemas duas informações básicas sobre os problemas são explicitamente representadas: o *enunciado* do problema e as *unidades de conhecimento* necessárias a sua resolução.

Exemplo

Continuemos o exemplo Estudo de Retas. Um possível particionamento para o conhecimento relativo ao subdomínio Estudo de Retas no contexto da Geometria Analítica

[8]Os diagramas de Hasse são usados comumente em matemática para representar conjuntos ordenados [DP90, p.7]. Em um diagrama de Hasse uma ligação entre dois pontos significa que o ponto de nível mais baixo tem precedência sobre o ponto de nível mais alto. Dois pontos de mesmo nível não são comparáveis.

restrito ao Plano Cartesiano consiste em dividir o subdomínio em cinco unidades pedagógicas [Buc92, cap.19]: *Coeficiente Angular*, *Equação da Reta*, *Posições Relativas entre Retas*, *Ângulos entre Retas* e *Distâncias entre Retas*.

Na unidade pedagógica Coeficiente Angular uma unidade de conhecimento (uc) é o conceito de *coeficiente angular calculado a partir de dois pontos*:

uc: Sejam $A(x_1, y_1)$ e $B(x_2, y_2)$ dois pontos do plano cartesiano. Sendo $x_1 \neq x_2$, o coeficiente angular m do segmento de reta $\overline{AB}$ é dado por:

$$m = \frac{y_2 - y_1}{x_2 - x_1}$$

Outros exemplos de unidade de conhecimento são os conceitos de *equação da reta dado um ponto e o coeficiente angular* pertencente à unidade pedagógica Equação da Reta e *retas perpendiculares* pertencente à unidade pedagógica Posições Relativas entre Retas:

uc: Sejam dados um ponto $A(x, y)$ e o coeficiente angular m de uma reta. A equação dessa reta é dada pela fórmula:

$$y - y_1 = m(x - x_1)$$

uc: Duas retas distintas são perpendiculares se, e somente se, o produto de seus coeficientes angulares for -1:

$$r \perp s \leftrightarrow m_r \cdot m_s = -1$$

Identificadas, as cinco unidades pedagógicas podem ser ordenadas conforme vemos no diagrama da Figura 2.6. No diagrama, a unidade Coeficiente Angular é pré-requisito para todas as outras; a unidade Equação da Reta é pré-requisito para as unidades Posições Relativas entre Retas, Ângulos entre Retas e Distância entre Retas; estas últimas não mantêm relação de dependência entre si.

Uma seqüência de aprendizagem ao longo do *curriculum* para o subdomínio Estudo de Retas no contexto da Geometria Analítica restrito ao Plano Cartesiano pode ser definida da seguinte forma. Primeiro o estudo da unidade pedagógica Coeficiente Angular, depois o estudo da Equação da Reta, depois Posições Relativas entre Retas, depois Ângulos entre Retas e por fim, Distâncias entre Retas. Notemos que nesta seqüência a ordem de pré-requisitos existentes entre as unidades pedagógicas é respeitada. Qualquer seqüência de unidades pedagógicas onde o estudo da Equação da Reta é feito antes do estudo do Coeficiente Angular não é uma seqüência de estudo válida pois a ordem de pré-requisitos estabelecida pelo *curriculum* não é preservada.

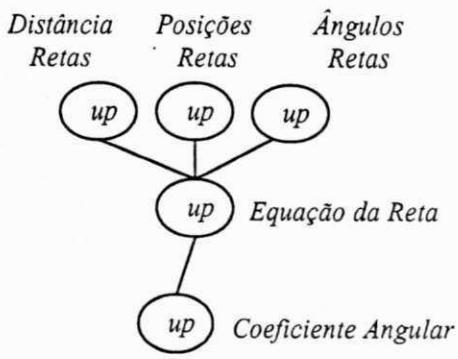


Figura 2.6: *Curriculum* para o subdomínio Estudo de Retas no contexto da Geometria Analítica restrito ao (profundidade) Plano Cartesiano

Um problema relativo ao subdomínio Estudo de Retas tem o seguinte enunciado: *Calcule o coeficiente angular da reta que passa pelos pontos* $A(3;2)$ *e* $B(1;4)$. Neste exemplo, o problema envolve a unidade de conhecimento coeficiente angular; deve-se ter em mente que dados dois pontos o coeficiente angular é calculado por $m = \frac{y_2 - y_1}{x_2 - x_1}$.

Um outro problema tem o enunciado: *Escreva a equação da reta que passa pelo ponto* $A(3;2)$ *e é perpendicular à reta* $y = -2x + 1$. Este problema envolve várias unidades de conhecimento: o conceito de equação (reduzida) de reta, a habilidade de identificar o coeficiente angular em uma equação reduzida, o conhecimento de que quando uma reta é perpendicular a outra o coeficiente angular de uma é o simétrico do inverso do coeficiente angular da outra e, por fim, o conhecimento de que dado um ponto e o coeficiente angular aplicamos a fórmula $y - y_1 = m(x - x_1)$ para escrever a equação da reta.

2.5 Os agentes MATHEMA

Uma vez tendo dividido e organizado o domínio de aprendizagem alvo através das visões externa e interna, o próximo passo no projeto de um ambiente de aprendizagem MATHEMA é a definição e criação dos agentes tutores que formarão a SATA.

2.5.1 Definição

Em linhas gerais, a definição dos agentes tutores MATHEMA pode ser dita da seguinte forma. Associe a cada subdomínio d_{ij} da visão externa do domínio de aprendizagem D um agente tutor AT_{ij} e, para cada agente tutor AT_{ij} defina como modelo de domínio o conhecimento do subdomínio d_{ij} organizado através da estrutura pedagógica.

2.5.2 Criação

Após a definição, a criação dos agentes tutores MATHEMA é feita por intermédio do ambiente de manutenção. O ambiente de manutenção recebe como entrada a definição (especificação) de um agente tutor e gera como saída o agente tutor (o software) AT_{ij} que integrará a SATA [Sil99].

Essencialmente, a arquitetura dos agentes tutores criados pelo ambiente de manutenção será formada por três sistemas principais [Cos97, cap.5]:

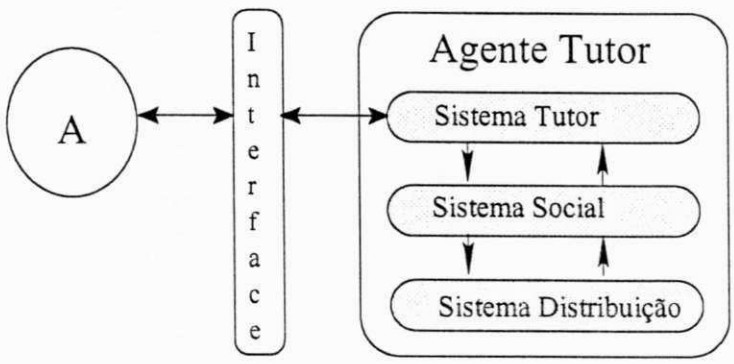


Figura 2.7: Arquitetura de um Agente Tutor MATHEMA

- *Sistema Tutor* : responsável pela interação direta com o aprendiz;
- *Sistema Social* : responsável pela viabilização do comportamento cooperativo entre os agentes tutores;
- *Sistema de Distribuição* : responsável pelo gerenciamento e distribuição das mensagens enviadas e recebidas pelo agente tutor através do meio de comunicação.

2.6 A Modelagem do Aprendiz

Dos três sistemas componentes de um agente MATHEMA, o Sistema Tutor é o alvo do trabalho de pesquisa apresentado nesta dissertação. Com veremos no Capítulo 4, o Sistema Tutor, em essência, é um tutor artificial; o componente de um agente MATHEMA responsável por orientar e supervisionar o aprendiz durante atividades de resolução de problemas. Nosso trabalho consiste em especificar a *modelagem do aprendiz* a ser realizada pelo Sistema Tutor dos agentes MATHEMA.

Capítulo 3

A Modelagem do Aprendiz

> *"Learner models can be seen as an explicit representation in the system of some characteristics of a particular learner, which will enable the system to adapt the knowledge communication to that particular learner."*[1]
>
> [PSH94]

3.1 Introdução

A *modelagem do aprendiz* é o processo através do qual um ambiente de aprendizagem por computador (ou seu projetista) adquire e mantém um *modelo do aprendiz.*

A *modelagem do aprendiz* pode ser estudada a partir de quatro questões tidas como fundamentais na área de modelagem de usuários de sistemas de computador [Fin89] :

1. *Quem modelar*: aprendizes individuais ou classes de aprendizes?
2. *O que modelar*: conhecimentos, planos, aptidões, etc.?
3. *Para que modelar*: que usos fazer do modelo?
4. *Como modelar*: como representar, adquirir e manter modelo?

Segundo Karen Jones [Jon89, p.341], qualquer empreendimento específico de modelagem do aprendiz que pretenda ser realista deve apresentar respostas claras a estas quatro questões. As três primeiras questões especificam as características principais de um *modelo do aprendiz*; a quarta estabelece os mecanismos computacionais utilizados para realizar a *modelagem do aprendiz.*

[1] "Modelos do aprendiz podem ser vistos como uma representação explícita no sistema de algumas características de um aprendiz particular, que habilitará o sistema a adaptar a comunicação de conhecimento ao aprendiz particular."

O nosso objetivo, neste capítulo, é apresentar uma revisão de várias idéias e abordagens existentes para a *modelagem do aprendiz* tendo como guia as quatro questões acima. Em nossa apresentação não almejamos completude; as idéias e abordagens existentes são muitas. O que apresentamos é um apanhado das idéias e abordagens a que tivemos acesso e que julgamos importantes ou relevantes aos objetivos a que nos propomos nesta dissertação. Assim, ao final esperamos ter formado uma estrutura de trabalho a ser utilizada nos próximos capítulos durante a especificação da *modelagem do aprendiz* para os ambientes de aprendizagem MATHEMA.

O capítulo está organizado em quatro seções principais. Na seção 3.2 discutimos algumas idéias relativas a modelos do aprendiz dando respostas às perguntas *quem modelar?*, *o que modelar?* e *para que modelar?* Nas seções seguintes buscamos responder a pergunta *como modelar?*, ou seja, mostrar que métodos e técnicas computacionais estão disponíveis para a implementação de *sistemas de modelagem do aprendiz.* Na seção 3.3 revisamos vários métodos para *representação* de *modelos do aprendiz.* Na seção 3.4 falamos sobre as principais estratégias e técnicas utilizados na aquisição de informações para *modelos do aprendiz.* Por fim, na seção 3.5 revisamos abordagens para a *manutenção* de *modelos do aprendiz* tendo em vista dois problemas principais: a *incerteza* inerente ao processo de modelagem e *manutenção da consistência* dos modelos.

3.2 O Modelo do Aprendiz

A idéia de construir e utilizar modelos do aprendiz em ambientes de aprendizagem por computador tornou-se um lugar comum. No entanto, o que se entende por *modelo do aprendiz* tem variado consideravelmente entre os pesquisadores. Uma primeira definição que damos para o termo modelo do aprendiz é a seguinte:

> *O* ***modelo do aprendiz*** *é uma representação computacional de alguns aspectos de um* aprendiz, *i.e., o usuário de um ambiente de aprendizagem por computador, que é usada pelo ambiente (como fonte de conhecimento) para adaptar-se ao aprendiz e tornar o processo de aprendizagem mais efetivo.*

Essa definição é muito ampla e vaga. Alguém pode perguntar: (1) Quem realmente é o aprendiz? (2) Que aspectos do aprendiz podem ser representados? (3) Como o ambiente de aprendizagem pode usar um modelo do aprendiz para adaptar-se ao aprendiz e tornar o processo de aprendizagem mais efetivo? Nesta seção iremos discutir como essas questões encontram respostas nos trabalhos de pesquisa já realizados. Ao

final consideraremos uma questão crucial: (4) Como um modelo do aprendiz pode ser efetivamente construído? [2]

Antes, fazemos uma importante distinção quanto à forma na qual um modelo do aprendiz pode estar codificado em um ambiente de aprendizagem. Quanto a forma de representação, alguns pesquisadores dividem os modelos do aprendiz em *modelos implícitos* e *modelos explícitos* [McC92]. Um modelo implícito é caracterizado como um conjunto de informações sobre possíveis aprendizes refletidas nas decisões de projeto de um ambiente de aprendizagem. Como exemplo Holt *et al.* [HDJG94] cita as metáforas e ícones utilizadas nas interfaces com o aprendiz; em geral estas interfaces veiculam de forma implícita uma idéia (um modelo) que o projetista faz dos aprendizes que irão utilizar o ambiente. Um modelo explícito, por outro lado, consiste em informações sobre os aprendizes representadas de forma explícita como um módulo identificável na arquitetura do ambiente (ver Figura 3.3) e que é consultado quando o ambiente necessita saber de algo sobre os aprendizes. Usando as técnicas e terminologia da Inteligência Artificial, um modelo explícito em geral assume a forma de uma *base de conhecimento*. Apresentaremos mais detalhes a este respeito na seção 3.3.

Observação. *Os modelos do aprendiz que consideramos neste capítulo, bem como em toda a dissertação, são modelos do aprendiz explícitos.*

3.2.1 Quem é o aprendiz sendo modelado?

Com relação ao aprendiz sendo modelado dois pontos merecem consideração [Fin89]: o *grau de especialização* e a *duração temporal* do modelo.

Quanto ao grau de especialização os modelos podem ser *individuais* ou *genéricos*. Nos modelos individuais o aprendiz sendo modelado é um único indivíduo. Ou seja, há uma representação separada para cada aprendiz usuário do ambiente de aprendizagem. Nos modelos genéricos o aprendiz sendo modelado corresponde a uma classe de

[2] Essas quatro questões correspondem respectivamente a quatro questões fundamentais na definição e construção de qualquer *modelo* entendido como *uma representação de aspectos de uma realidade com um fim específico*:

1. Qual a realidade a ser modelada?
2. Que aspectos desta realidades serão representados no modelo?
3. Qual o propósito do modelo?
4. Que recursos serão utilizados na construção do modelo?

indivíduos. Ou seja, há uma única representação para uma classe ou tipo de aprendizes.

Quanto à duração temporal os modelos podem ser *persistentes* ou *temporários*. Nos modelos persistentes a representação do aprendiz é feita por grande período tempo (várias seções de aprendizagem). Nos modelos temporários a representação do aprendiz tem um curto período de duração (uma única seção de aprendizagem).

3.2.2 Que aspectos do aprendiz podem ser modelados?

Há três limites para o conteúdo de um modelo do aprendiz. O primeiro, a imaginação do pesquisador; o segundo, as necessidades informacionais do ambiente de aprendizagem; o terceiro, os recursos computacionais disponíveis para a realização do modelo[3]. No que segue mostramos apenas o que os pesquisadores *têm imaginado* ser interessante representar em um modelo do aprendiz.

A construção de modelos do aprendiz tem levado os pesquisadores a explorar um vasto espaço de informações. Reunindo as observações de alguns pesquisadores [HDJG94, Ver94, Sel94a, Fin89] quanto ao conteúdo de um modelo do aprendiz, chegamos à conclusão de que os modelos de aprendiz já propostos têm explorado três grandes categorias de informações. São elas: o *Conhecimento* dos aprendizes, suas *Intenções* e alguns *Atributos* individuais. Na Figura 3.1 mostramos mais detalhes.

Conhecimentos

O *conhecimento* de um aprendiz engloba três subcategorias: (1) o conhecimento sobre o domínio de aprendizagem, ou seja, o que o aprendiz já assimilou da área sendo estudada; (2) conhecimentos gerais, ou seja, o que o aprendiz sabe além do que está sendo estudado; e (3) meta-conhecimentos, ou seja, o que o aprendiz sabe sobre como utilizar o conhecimento que tem (conhecimento reflexo), ou ainda, o que o aprendiz sabe sobre o que já sabe ou não sabe (conhecimento introspectivo)[Sel94a, Sel95b].

Intenções

As *intenções* consistem nos objetivos, planos e estratégias de um aprendiz durante o processo de aprendizagem. Um objetivo é um estado de coisas que o aprendiz quer

[3]Em geral isso é válido para qualquer modelo. Inicialmente podemos imaginar muitos aspectos de uma realidade que podem ser retratados em um modelo. O propósito do modelo irá restringir esse espaço de possibilidades aos aspectos essenciais. Por fim, os meios que dispomos para codificar o modelo, em especial os meios computacionais, tendem a restringir ainda mais o que iremos ou não representar no modelo.

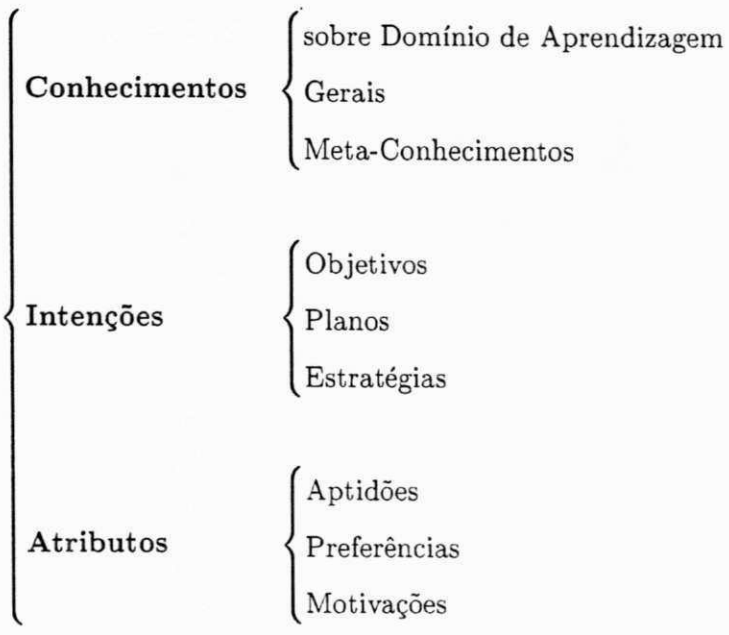


Figura 3.1: O possível conteúdo de um modelo do aprendiz.

atingir; os planos e estratégias são seqüências de ações ou eventos que o aprendiz espera resultem na realização de seus objetivos. Em ambientes de aprendizagem por resolução de problemas os objetivos são respostas corretas para problemas; os planos e estratégias consistem em uma estruturação de alguns conhecimentos que aplicados a um problema dão origem a um conjunto de ações cujo resultado final é uma resposta para o problema.

Atributos

Os *atributos* estão relacionados com algumas características individuais do aprendiz. Estas características são divididas em três subcategorias. (1) As *aptidões* do aprendiz, ou seja, as tendências e capacidades do aprendiz; e.g., sua tendência em refletir antes de responder a perguntas, sua capacidade de assimilar novos conhecimentos, sua capacidade de reter novos conhecimentos, sua capacidade de aprender por associação, e outras [Shu95]. (2) As *preferências* do aprendiz; e.g., o tipo de interação que o aprendiz prefere (ter o controle ou ser controlado), os tipos de problemas que o aprendiz prefere resolver (fáceis, difíceis, médios, abstratos, aplicados, etc.), e outras. (3) As *motivações* do aprendiz ao longo do processo de aprendizagem [DS92a].

3.2.3 Para que modelar esses aspectos?

Um modelo do aprendiz não precisa ser detalhado e sim útil. De nada adianta ter representado todo o conhecimento do aluno, seus objetivos, planos, suas aptidões, preferências e motivações se não houver meios de um ambiente de aprendizagem utilizar essas informações para adaptar-se ao aprendiz e tornar o processo de aprendizagem mais efetivo. Vejamos então como as informações apresentadas na seção anterior têm sido ou podem ser utilizadas com este fim.

Conhecimentos

O *conhecimento* de um aprendiz, em especial o conhecimento sobre o domínio de aprendizagem, tem sido de longe o principal tipo de informação explorado na construção de modelos do aprendiz. Tanto que, na maioria dos projetos de pesquisa já divulgados, o termo modelo do aprendiz é empregado como sinônimo de representação dos conhecimentos de um aprendiz sobre o domínio de aprendizagem [Rag96, DS92b]. Este fato é perfeitamente compreensível se pensarmos no objetivo básico de um ambiente de aprendizagem por computador que é fazer um aprendiz adquirir conhecimentos. Assim, dispor de informações sobre o estado de conhecimento do aprendiz pode ser e tem sido útil para tornar o processo de aprendizagem por computador mais efetivo [Sel88, Sel94b, HDJG94].

Os usos de informações sobre o estado de conhecimento de um aprendiz podem ser agrupados em três categorias principais: uso *corretivo*, uso *elaborativo*, e uso *avaliativo*.

Uso corretivo. O uso corretivo consiste em identificar falhas no conhecimento do aprendiz, chamadas de *mal-entendidos*, e a partir desta informação iniciar um processo de *remediação* ou *correção* dos mal-entendidos.

O processo de remediação pode ser feito de várias formas [Sel94a]:

- *remediação direta*: se o sistema descobre que o aprendiz aprendeu determinado conhecimento de forma errada então pode decidir re-apresentar o assunto ao aprendiz;
- *remediação por emenda*: se o sistema isola um mal-entendido então com esta informação pode mostrar ao estudante seu erro, mostrar o correto e dar uma justificativa;
- *remediação por contra-exemplos*: se o sistema descobre que há falhas no conhecimento do aprendiz pode então decidir apresentar alguns fatos que contradigam

as crenças do aprendiz dando oportunidade para que este reflita sobre o que antes acreditava ser correto.

O uso corretivo foi bastante enfatizado na década de 80 durante a construção dos Sistemas Tutores Inteligentes (STI)[4] :

> "Em instrução [nos STI], nós monitoramos o comportamento do estudante (um sistema cognitivo), procuramos por discrepâncias em relação à especificação ideal (modelo de resolução de problema alvo), trilhamos o caminho das discrepâncias aos erros no presumido modelo de mundo ou procedimento de inferência do estudante, e 'reparamos' o estudante por instrução." [Cla88, p.60]

Atualmente é muito criticado e desaconselhado. A crítica mais séria quanto a esse uso de informações sobre o estado de conhecimento do aprendiz é que em muitas situações de aprendizagem algumas crenças e comportamentos do aprendiz que desviem do padrão estabelecido como correto não podem e nem devem ser identificados como mal-entendidos. Por exemplo, uma resposta "errada" do aluno a um problema pode indicar que há uma falha no conhecimento do aprendiz; no entanto, pode indicar também que o aluno estava distraído, pode indicar que ele ao escrever sua resposta tenha cometido um deslize, ou ainda, e o que é pior, pode indicar que o aluno sabe mais que o sistema e encontrou uma resposta alternativa.

Um outro problema com o uso corretivo é que ele requer mecanismos de representação de mal-entendidos e diagnóstico de erros elaborados o que torna o processo de criar o modelo do aprendiz uma tarefa difícil computacionalmente [Sel90]. Discutiremos mais a este repeito nas seções 3.3.1 e 3.4.3.

Uso elaborativo. O uso elaborativo consiste em empregar as informações sobre o conhecimento do aprendizes para ajudá-lo a ampliar e melhorar a qualidade de seu conhecimento. Na prática isto corresponde a *decidir sobre que tópicos de um curriculum ou atividades* o aluno está apto a estudar ou realizar; *gerar ou selecionar problemas, exercícios* de acordo com o nível de conhecimento do aprendiz; *dar dicas, explicações, orientações* levando em conta o que o aluno já sabe; e outros.

O uso elaborativo não requer que informações sobre mal-entendidos estejam representadas no modelo do aprendiz e é atualmente o principal uso de um modelo do

[4] Falamos dos STI na seção 2.2. Nos STI a proposta educacional é comunicar um corpo de conhecimento tido como correto através de ensino direto ou instrução. Neste caso o processo de aprendizagem é efetivo quando o STI consegue transmitir fielmente tudo o que sabe ao aprendiz.

aprendiz em Ambientes de Aprendizagem Interativos (AAI)[5].

> "[Os AAI] necessitam tomar decisões baseadas em um modelo do estudante para ajudar os estudantes. Entretanto, estes modelos do estudante não são do tipo desenvolvido para sistemas tutores, ou seja, os que se concentram no (mau-)entendimento do estudante em um domínio de conhecimento específico. Os modelos do estudante dos AAI (...) pretendem suportar o controle conjunto do diálogo (e não fazer com que o sistema sozinho tome todas as decisões instrucionais)."[Sel94b, p.8]

Uso avaliativo. O uso consiste em utilizar as informações sobre o conhecimento do aprendiz para gerar avaliações sobre o desempenho do aprendiz. Em sua forma mais simples esta avaliação pode tomar a forma de um número sumarizando o desempenho do aprendiz. Em uma forma mais elaborada a avaliação pode incluir descrições sobre que tópicos de um *curriculum* o aluno se saiu bem, que tópicos foi mal, o que precisa rever, e outros.

Planos e objetivos

As informações sobre os planos e objetivos de um aprendiz têm sido utilizadas de duas formas básicas [Woo88, Gen82]. A primeira, *ajudar o ambiente de aprendizagem durante a etapa de aquisição de novas informações sobre o conhecimento do aprendiz.* Este uso comentaremos na seção 3.4. A segunda, *ajudar o aprendiz a atingir seus objetivos.* Em um ambiente de aprendizagem por resolução de problemas isto pode significar *oferecer dicas, dar conselhos, explicações* e/ou *mostrar caminhos alternativos durante a resolução de um dado problema.*

Um exemplo ilustrativo é o trecho de interação entre um Consultor, o sistema MACSYMA *Advisor*[Gen82], e um estudante de nome Carleton mostrado na Figura 3.2. No diálogo (adaptado de [Gen82, p.140]) o estudante diz ao Consultor que tentou simplificar uma expressão matemática usando comandos do sistema MACSYMA e encontrou um resultado não esperado. Após analisar a estratégia de simplificação adotada pelo estudando, seu plano, o Consultor conclui que um comando, o `Coeff(exp,var,pow)`,

[5] Falamos dos AAI na seção 2.2. Nos AAI a proposta educacional é deixar o aprendiz construir e elaborar seu conhecimento fornecendo um ambiente e assistência individualizados. Neste caso a aprendizagem é efetiva quando além de transmitir algum conhecimento "correto", o ambiente desperta no aprendiz habilidades metacognitivas tais como aprender a aprender, aprender a refletir sobre seu aprendizado, aprender a planejar e monitorar seu comportamento em situações de resolução de problemas, e outras.

Consultor: Fale!
Carleton: Eu estava tentando simplificar $x^2 - z(x + xy - yz)$ para x e obtive 0.
Consultor(após ver estratégia do estudante): Você espera que `Coeff` retorne o coeficiente de $x^2 - z(x + xy - yz)$
Carleton: Sim , não é?
Consultor: `Coeff(exp,var,pow)` retorna ... Talvez você deva usar `Ratcoeff`.
Carleton: Ok, obrigado. Bye.

Figura 3.2: Usos de informações sobre planos para ajudar um aprendiz.

foi utilizado de forma indevida. Ao final, o Consultar dá uma dica, usar o comando `Ratcoeff`, para que o estudante possa completar seu objetivo.

Uma terceira forma de utilizar informações sobre planos e objetivos de um aprendiz é avaliar o progresso do processo de aprendizagem. "Modelando os planos de um estudante sobre um número de episódios de resolução de problemas, um sistema pode desenvolver um entendimento do progresso da aprendizagem de um estudante."[GK95].

Atributos

As informações sobre atributos individuais, com relação ao estado de conhecimento do aprendiz, seus planos e objetivos, quase não têm sido empregadas na construção de modelos do aprendiz. Uma das explicações é o fato de haver pouco consenso em como estas informações podem ser efetivamente utilizadas. A outra explicação, é o fato de que tais informações são em geral de difícil tratamento computacional.

Entre os poucos trabalhos de pesquisa realizados está o sistema *Stat Lady*, que se utiliza de uma abordagem de modelagem do estudante chamada *Smart* [Shu95]. Nesta abordagem, algumas informações sobre as aptidões dos estudantes são consideradas:

> "A idéia básica é que os resultados da aprendizagem podem refletir diferenças em aptidões que surgem, tais como habilidade associativa, capacidade de memorização, ou reflexividade. (...)
>
> (...) Pesquisa anteriores nesta área têm demonstrado que vários testes de habilidades cognitivas (e.g., ...) podem acuradamente *predizer o desempenho do treinamento* (e.g., ...) e *ajudar atribuir estudantes a ambientes de treinamento que se adequem às suas aptidões* (e.g., ...)."[Shu95, p.7]

Um outro trabalho de pesquisa que considera o uso de informações sobre aptidões dos aprendizes é o sistema MFD [BSW97]. Neste sistema informações sobre as capaci-

dades de *aquisição* e *retenção* de novos assuntos pelos aprendizes são representadas e utilizadas para *controlar a dificuldade dos problemas* a serem passados.

Além de aptidões, algumas informações sobre preferências, representadas através de estereótipos (ver seção 3.3.3), têm sido empregadas para *tomada de decisões estratégicas por parte do ambiente de aprendizagem* [Sel88, pp.80-82]. Com relação ao uso de informações sobre motivações o artigo [DS92a] apresenta um trabalho de pesquisa nesta direção.

3.2.4 Como realizar a modelagem do aprendiz?

Como já dissemos há três limites para o conteúdo de um modelo do aprendiz: primeiro os aspectos sobre o aprendiz que *imaginamos* poder representar (seção 3.2.2); segundo, as informações que o ambiente de aprendizagem *realmente* necessita ou pode vir a utilizar para promover o processo de aprendizagem (seção 3.2.3); e terceiro, as informações que os recursos computacionais de que dispomos nos permitem representar, adquirir e manter. Este último é o tema a ser abordado com algum detalhe no restante deste capítulo.

A Modelagem do Aprendiz

A construção de um modelo do aprendiz é um processo que envolve três tarefas básicas: (1) a especificação de uma estrutura representacional para o modelo do aprendiz; (2) a aquisição das informações sobre o aprendiz; e (3) a incorporação e manutenção das informações no modelo do aprendiz. A tarefa (1) é realizada uma única vez pelo projetista do modelo do aprendiz; as tarefas (2) e (3) podem ser realizada inúmeras vezes por meio do processo de *modelagem do aprendiz*:

> *A **modelagem do aprendiz** é o processo de aquisição e manutenção de informações em um modelo do aprendiz.*

Dependendo do tipo de modelo envolvido (quem, o que, para que), a modelagem do aprendiz pode ser realizada de duas formas e em dois momentos [Pai95]: (1) Pelo projetista *antes* (*off-line*) de qualquer interação entre o ambiente de aprendizagem e o aprendiz. Esta é a forma de modelagem utilizada na construção de modelos de aprendiz implícitos ou modelos explícitos tendo em vista classes de usuários. (2) Pelo ambiente de aprendizagem *durante* (*on-line*) a sua interação com o aprendiz. Esta é a forma de modelagem utilizada na construção de modelos do aprendiz explícitos tendo em vista principalmente o estado cognitivo de aprendizes individuais.

O Sistema de Modelagem do Aprendiz

Na modelagem do aprendiz realizada durante a interação entre o ambiente de aprendizagem e o aprendiz, a aquisição e a manutenção do modelo do aprendiz ficam a cargo de um *sistema de modelagem do aprendiz*:

> *Chamamos de* ***sistema de modelagem do aprendiz*** *o componente de um ambiente de aprendizagem por computador responsável por realizar o processo de modelagem do aprendiz durante a interação do ambiente com o aprendiz.*

Um sistema de modelagem do aprendiz em geral é formado por três módulos principais [PS95]: um módulo de *aquisição*, um módulo de *manutenção* e um *modelo do aprendiz*. Na Figura 3.3 mostramos estes módulos e os relacionamentos existentes entre eles. O módulo de aquisição é o módulo encarregado de interagir com o aprendiz, seja

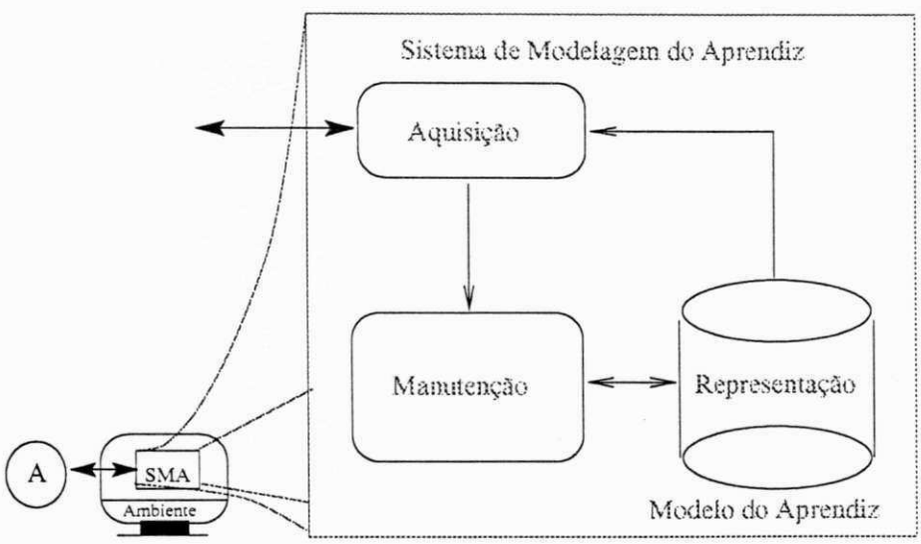


Figura 3.3: A estrutura de um sistema de modelagem do aprendiz

de forma direta (fazendo questões) ou de forma indireta (observando seu comportamento), e com isto coletar ou inferir informações sobre seu estado de conhecimento, seus planos, aptidões, etc., durante o processo de aprendizagem. Como veremos, as informação obtidas pelo módulo de aquisição em sua maioria serão incertas, incompletas e até mesmo inconsistentes. O módulo de manutenção é encarregado de receber as informações (incertas, incompletas, inconsistentes) sobre o aprendiz provindas do módulo de aquisição e atualizar o modelo do aprendiz. O modelo do aprendiz, como já dissemos, é uma representação computacional (explícita) das informações adquiridas sobre o aluno.

3.3 A Representação do Modelo

O modelo do aprendiz é uma *representação computacional* (explícita) de alguns aspectos de um aprendiz usuário de um ambiente de aprendizagem por computador. Os aspectos que se têm imaginado poder representar são divididos em três categorias. Ver Figura 3.4.

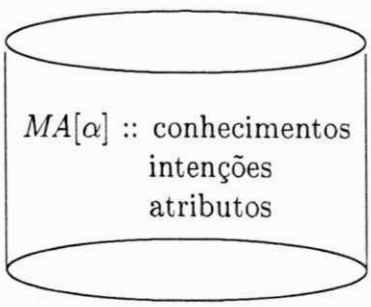


Figura 3.4: A representação do modelo do aprendiz.

Na prática, o número de aspectos representados tem sido reduzido. *Poucos trabalhos têm ido além da representação do conhecimento dos aprendizes em relação ao domínio de aprendizagem.* Segundo Holt *et al.*, os "trabalhos têm sido limitados pela falta de mecanismos padronizados para facilmente determinar o estado do aprendiz relativo a características individuais além do conhecimento especializado do domínio e mal-entendidos comuns"[HDJG94, p.5]. O que apresentamos nesta seção é um reflexo desta situação.

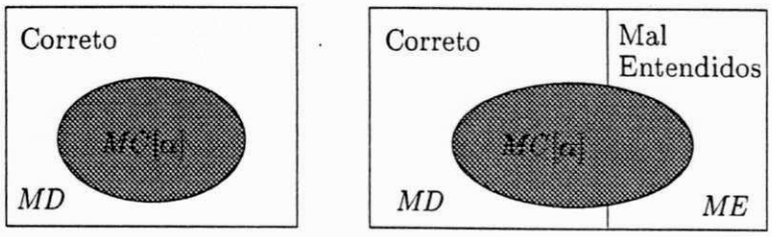
3.3.1 Modelo Cognitivo

Chamamos de *modelo cognitivo* a representação computacional do conhecimento de um aprendiz em relação um domínio de aprendizagem incluída em um modelo do aprendiz.

O modelo cognitivo, em geral, é construído como uma *base de conhecimento.* De forma rápida, uma base de conhecimento é um conjunto de sentenças de uma linguagem simbólica, dita *linguagem de representação de conhecimento*, que buscam capturar o nosso conhecimento sobre fatos, objetos, relacionamentos entre os objetos e a forma como raciocinamos com estes fatos em um dado domínio [RN95].

Utilizaremos a seguinte notação para expressar a idéia de um modelo cognitivo como uma base de conhecimento:

$$MC[\alpha] = \{\varphi \in \mathcal{L} \mid RC(\varphi, \alpha, D)\}$$



a) representação por sobreposição

b) representação de perturbações

Figura 3.5: Métodos de representação do $MC[\alpha]$ tendo como referência o MD

Onde $MC[\alpha]$ denota o modelo cognitivo de um aprendiz de nome α; $\mathcal{L}$ a linguagem de representação de conhecimento utilizada na codificação de $MC[\alpha]$ e, φ tal que $RC(\varphi, \alpha, D)$ as sentenças de $\mathcal{L}$ que representam o conhecimento de α sobre o domínio de aprendizagem D. As *redes semânticas* [Car70] e o formalismo de *regras de produção* [ACKP95], como veremos a seguir, são alguns exemplos de linguagens para representação de conhecimento já utilizadas na codificação de modelos cognitivos.

Na grande maioria dos ambientes de aprendizagem por computador, o modelo cognitivo é representado tendo como referência uma outra base de conhecimento, o *modelo do domínio*. No modelo do domínio está representado o conhecimento que o sistema espera comunicar aos seus aprendizes usuários. Para fins explicativos, diremos que o modelo de domínio é o modelo cognitivo de um *aprendiz ideal*[RAF85]. Ou seja, a representação do conhecimento que se espera de um aprendiz muito bom na área sendo estudada[6]. Em símbolos:

$$MD = \{\varphi \in \mathcal{L} \mid RC(\varphi, \textit{aprendiz-ideal}, D)\}$$

Existem dois métodos básicos comumente usados na representação de um $MC[\alpha]$ tendo como referência o MD. São elas a *representação por sobreposição* e a *representação de perturbações* [Rag96, Pai95, HDJG94, Kas89]. Em ambos o $MC[\alpha]$ é representado por comparação com o MD; ver Figura 3.5.

[6] Ressaltamos que, ao contrário de [RAF85], não dizemos que esta representação deva ter alguma validade psicológica espelhando estruturas mentais que o aluno irá adquirir após interagir com o sistema. Ver ainda subseção *Teorias Cognitivas* na seção 2.2

Representação por sobreposição

No método de representação por sobreposição[7] (Figura 3.5 (a)) o conhecimento do aprendiz é visto como um subconjunto próprio de *MD*:

$$MC[\alpha] \subset MD \qquad (\dagger)$$

Esta inclusão tem uma série de implicações.

Em nossa notação implica que a linguagem de representação de conhecimento usada na codificação do $MC[\alpha]$ deve ser um subconjunto da linguagem usada na codificação do *MD*. Neste caso dizemos que $MC[\alpha]$ herda a linguagem de representação de *MD*.

Quanto aos aspectos práticos implica que a representação por sobreposição é um método bastante simples e fácil de implementar: em geral, herda a linguagem de representação[8] e a estrutura do conhecimento previamente codificado no sistema. Ainda, é um método que se tem mostrado bastante útil em ambientes de aprendizagem cujo objetivo é comunicar um corpo de conhecimento já consolidado e organizado através de um *curriculum*.

Muitos sistemas já tentaram esta abordagem de representação por sobreposição com algum sucesso. Um exemplo típico é o pioneiro sistema SCHOLAR desenvolvido por Carbonell no início de 1970 [Car70]. Neste sistema o conhecimento a ser ensinado era geografia da América do Sul. A linguagem de representação escolhida por Carbonell para codificar o domínio de conhecimento, herdado pelo modelo do aprendiz, era o formalismo das *redes semânticas* [Wen87, p.32]. Outros exemplos bastante conhecidos são os sistemas WUSOR [CG77] e GUIDON [Cla83] ambos utilizando um formalismo de *regras de produção* como linguagem de representação de conhecimento.

Por fim, quanto aos aspectos educacionais implica que a representação por sobreposição, quando usada de forma muita rígida quanto aos seus pressupostos, é um método que deixa a desejar em alguns pontos. Um dos pressupostos por trás da rep-

[7] Representação por sobreposição é uma tradução livre de *overlay model*, nome originalmente dado a esta técnica por Carr e Goldstein na década de 70 [CG77].

[8] A *herança de linguagem* é a técnica mais básica de implementar a idéia de sobreposição. Contudo, nem sempre é o caso. A representação pode adotar uma linguagem diferente e mais simples devido ao fato que muitos detalhes presentes no *MD* poderem ser irrelevantes e desnecessários para o $MC[\alpha]$. A linguagem pode ainda ser mais rica tendo que capturar informações extras e descrições necessárias ao $MC[\alpha]$ e que não pertencem ao *MD*.

resentação por sobreposição é que quaisquer "*erros*"[9] cometidos pelo aprendiz devem ser atribuídos única e exclusivamente a sua falta de conhecimento em relação ao *MD*. Assume-se que o aprendiz não irá pensar diferente do *MD*. Em alguns casos, no entanto, o aluno pode vir a "errar" não pela falta de conhecimento mais por ter conhecimentos que não estão previstos no *MD*.

Representação de perturbações

A representação de pertubações[10] é uma extensão do método de sobreposição que procura representar conhecimentos do aprendiz que não pertençam ao *MD* assumindo que estes consistem em mal-entendidos, ou seja, *perturbações* no conhecimento aprendido que levam o aprendiz a agir de forma "errada". Em nossa notação escrevemos:

$$MC[\alpha] - MD = P[\alpha] \qquad (\ddagger)$$

Onde $P[\alpha]$ denota a representação dos mal-entendidos ou perturbações no conhecimento do aprendiz α.

Através da representação de pertubações, os "erros" do aprendiz passam a ser explicados tanto pela falta de conhecimento quanto pela influência de mal-entendidos, ou ambos.

Uma técnica comumente utilizada para implementar a idéia de perturbação é a criação de *catálogos de mal-entendidos*. O catálogo de mal-entendidos é uma coleção de mal-entendidos mais comuns em um domínio de aprendizagem identificados pelo projetista e representado no sistema. Um exemplo de um catálogo de mal-entendidos relativos ao domínio de álgebra nos é dado pelo trabalho [Mar95, pp.235-236], parte do qual listamos na Figura 3.6.

Usando um catálogo de mal-entendidos *ME*, os mal-entendidos do aprendiz são diagnosticados e representados como um subconjunto do catálogo: $P[\alpha] \subset ME$. Juntando esta relação com a relação ($\ddagger$) anterior temos que:

$$MC[\alpha] \subset MD \cup ME$$

Ou seja, com o uso de um catálogo de mal-entendidos o modelo do aprendiz $MC[\alpha]$ é representado por sobreposição do conhecimento do sistema *MD* estendido com os

[9] Chamamos de "erros" discrepâncias entre o *comportamento* do aluno e o *comportamento* esperado pelo sistema através de *MD*. Geralmente "erros" ocorrem em situações de resolução de problemas onde *comportamento* significa a solução de um problema e *MD* é um modelo de resolução de problemas.

[10] Tradução livre de *perturbation model*.

8. ...
9. $\frac{x+y}{x+z} = \frac{y}{z}$
10. $\frac{1}{x-y} = \frac{-1}{x+y}$
11. $\frac{x}{y} + \frac{r}{s} = \frac{x+r}{y+s}$
12. $x\left(\frac{a}{b}\right) = \frac{ax}{bx}$
13. $\frac{xa+xb}{x+xd} = \frac{a+b}{d}$
14. ...

Figura 3.6: Catálogo de mal-entendidos em álgebra.

mal-entendidos mais comuns *ME* identificados em um domínio de aprendizagem; ver Figura 3.5 (b).

A representação de perturbações tem sido utilizada principalmente na construção de modelos do aprendiz com fins de diagnóstico e remediação de erros. Um dos exemplos mais conhecidos são os três sistemas desenvolvidos durante o projeto BUGGY [Bur82]. O objetivo do projeto era desenvolver programas que diagnosticassem causas de erros comuns cometidos por crianças ao realizarem subtrações. Os programas construiam um *modelo de diagnóstico* dos alunos representado através de uma *rede de procedimentos* [Wen87, p.155]. Na rede de procedimentos do sistema havia tanto procedimentos corretos de subtração quanto procedimentos com algumas falhas chamados de "*bugs*". A rede inteira formava um reticulado de habilidades de subtração que pretendia representar todas as formas possíveis de um aprendiz tentar resolver um problema de subtração. Um modelo de diagnóstico seria uma parte desta rede.

Outros exemplos do uso da representação de perturbação são os sistemas LMS [SS81], GREATERP e o Geometry Tutor [ACKP95], todos utilizando regras de produção como linguagem de representação para o domínio de aprendizagem e catálogo de mal-entendidos.

Por envolver a representação de mal-entendidos, a representação de perturbações não é um método tão simples de implementar. Um ponto bastante discutido na literatura é como catalogar e representar os mal-entendidos. Nos programas BUGGY o *catálogo de "bugs"* era predefinido pelos projetistas. Outra abordagem é ter um mecan-

ismo de geração deste conhecimento defeituoso a partir da base de conhecimento do sistema [VL87]. As duas abordagens apresentam problemas sérios. No primeiro caso é difícil e demorado coletar informações sobre os mal-entendidos mais comuns cometidos por aprendizes de uma área de conhecimento. No segundo caso é difícil justificar psicologicamente a geração de conhecimento defeituoso.

Representação por sobreposição com incerteza

A representação por sobreposição e a representação de pertubações são considerados os métodos *básicos* ou *padrões* de representação na área de modelagem do estudante. Além destes métodos existe uma série de outros mais específicos e menos conhecidos que são considerados como *variações* ou *extensões* dos métodos básicos; em especial da idéia de sobreposição[11]. Duas destas variações, extensões são os chamados *modelo de aprendiz limitado*[12] [EC88] e o *modelo do aprendiz baseado em restrições*[13] [Ohl94]. Uma outra extensão é o que chamamos de *representação por sobreposição com incerteza.*

Para compreendermos a representação por sobreposição com incerteza devemos examinar a idéia de sobreposição utilizando o conceito de *função característica.* Em teoria dos conjuntos, como sabemos, a função característica de um conjunto $A \subseteq U$ é definida como:

$$\chi_A : U \rightarrow \{0, 1\}$$

$$\chi_A(u) = \begin{cases} 1 \text{ se, somente se } u \in A \\ 0 \text{ se, somente se } u \notin A \end{cases}$$

Assim definida, a função característica formaliza o processo por meio do qual elementos de um conjunto U são determinados *com certeza. precisão* serem ou não membros de um subconjunto A de U.

No âmbito da representação por sobreposição vemos que a função característica formaliza o processo de representar o modelo cognitivo por sobreposição. Chamaremos $\chi_{MC[\alpha]}$ de *função de representação* de $MC[\alpha]$. Assim definida, a função de representação reflete a idéia de que a modelagem do aprendiz, cujo objetivo último é estabelecer $\chi_{MC[\alpha]}$, é capaz de determinar *com certeza, precisão* que conhecimentos fazem ou não

[11] Vale notar que a técnica mais comum usada para implementar a idéia de perturbação, o catálogo de mal-entendidos, é em essência uma extensão da representação por sobreposição.

[12] Tradução livre de *bounded user model*

[13] Tradução livre de *constraint-based model*

parte do estado de conhecimento do aluno. Como será discutido na seção 3.5, isso não é verdade.

A representação por sobreposição com incerteza, partindo da idéia de função de representação, estende a representação por sobreposição incorporando elementos que expressem a incerteza inerente ao processo de modelagem do aprendiz. Isto é feito generalizando a noção de função de representação através do que chamaremos de *função de representação de $MC[\alpha]$ com incerteza.* Em símbolos:

$$I_{MC[\alpha]} : MD \rightarrow \mathcal{L}^I$$

Onde $\mathcal{L}^I$ é dita *linguagem de repesentação de incerteza.* Notemos que sendo $\mathcal{L}^I = \{0, 1\}$ temos a representação por sobreposição padrão.

Adiaremos para a seção 3.5 uma discussão sobre linguagens de representação de incerteza e o significado atribuído aos $I_{MC[\alpha]}(\varphi) \in \mathcal{L}^I$. Lá veremos duas linguagens principais, a linguagem da *teoria dos conjuntos difusos* e a linguagem da *teoria de probabilidades*, apropriadas para dois tipos distintos de incerteza, respectivamente, *incertezas lingüísticas* e *incertezas estocásticas.*

Teorias Lógicas

Como dissemos de início, na grande maioria de ambientes de aprendizagem onde é realizada a modelagem do aprendiz, o $MC[\alpha]$ é representado tendo como referência o MD. No entanto, alguns ambientes, notadamente ambientes de aprendizagem por descoberta (seção 2.2), não representam conhecimento "correto" e "completo" a ser transmitido ao aluno; quer dizer, não possuem MD. Neste caso há algumas abordagens alternativas para o problema de representar o que aluno sabe no decorrer do processo de aprendizagem.

Uma das abordagens utilizadas é construir uma *teoria lógica* que explique o comportamento observado dos aprendizes. Por teoria lógica queremos dizer um conjunto de sentenças de uma linguagem lógica (e.g. lógica de proposições, predicados, lógicas modais não monotônicas) inferidas a partir do comportamento observado do aprendiz e que representam o seu estado de conhecimento. Construir tais teorias observando e buscando por regularidades no comportamento observado dos aprendizes pode ser conseguido usando técnicas que em geral são chamadas de *aprendizagem de máquina.* Falaremos a este respeito na seção 3.4.2.

A construção de teorias lógicas para representar o estado cognitivo do aprendiz tem como vantagem não assumir que o aprendiz deve se enquadrar a um MD pré-estabelecido, como nas técnicas de sobreposição e representação de perturbação. Um

dos problemas básicos, no entanto, é a própria construção e a manutenção da teoria, ou seja, a manutenção da consistência lógica do modelo cognitivo gerado. Sobre o problema da consistência em modelos cognitivos como teorias lógicas falaremos na seção 3.5.3.

3.3.2 Representação de Planos

Em um modelo do aprendiz, além do conhecimento do aluno em relação ao domínio de aprendizagem, podem estar representados os planos, estratégias e objetivos dos aprendizes durante atividades de resolução de problemas. Ao contrário da representação do estado de conhecimento dos aprendizes, não existem métodos padrões para representação de planos. O que há são linguagens particulares para representação de planos já utilizadas em alguns ambientes de aprendizagem.

MACSYMA Advisor: Planos como Grafos de Dependências

Uma dessas representações de planos é a utilizada no sistema MACSYMA Advisor [Gen82]. Neste sistema os planos são vistos como grafos de dependências que explicam como as ações do aprendiz atingem seus objetivos em termos de seu conhecimento sobre a área do problema. Na Figura 3.7 mostramos um plano para computar uma raiz de uma equação do segundo grau usando fórmula de Baskara adaptado de [Gen82, p.145].

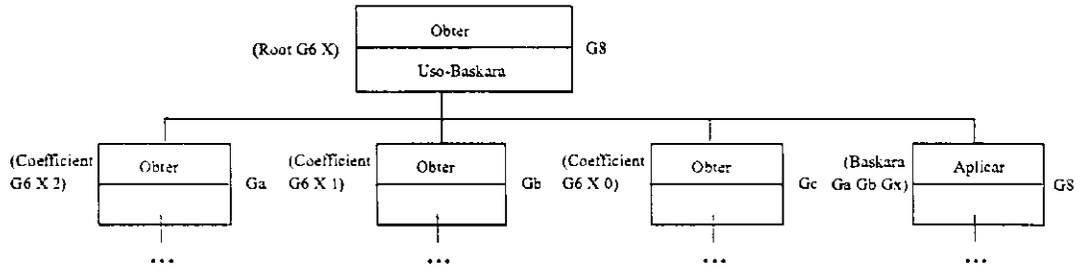


Figura 3.7: Planos como grafos de dependências.

Na figura cada caixa representa um plano formado possivelmente por alguns subplanos. A esquerda de cada plano está o objetivo a ser atingindo com o plano. A direta está o resultado após o plano ser executado. Assim o plano geral consiste em achar uma raiz de uma equaçao de nome `G6` : `(Root G6 X)`. O resultado será chamado de `G8`. O plano é dividido em quatro subplanos. Obter coeficiente do termo de segundo grau: `(Coefficient G6 X 2)`. O resultado será chamado de `Ga`. Obter coeficiente do termo primeiro grau: `(Coefficient G6 X 1)`. O resultado será chamado de `Gb`. Obter coeficiente do termo de grau zero: `(Coefficient G6 X 0)`. O resultado será

chamado de Gc. Por fim a fórmula de Baskara é aplicada : (Baskara Ga Gb Gc). O resultado é chamado de G8.

FITS-2: Hierarquias de Planos

Uma exceção às representações de planos desenvolvidas em ambientes de aprendizagem particulares é a representação utilizada no sistema FITS-2 [Woo88] que segundo o autor pretende ser a base de uma ferramenta para a construção de STI. Neste sistema um plano do estudante consiste em uma estrutura em forma de grafo onde os nós são chamados de *elementos de expansão* e correspondem a subplanos ou *planos filhos* parametrizados e os arcos correspondem a um ordenamento necessário entre os elementos de expansão e quaisquer possíveis *restrições* em seus parâmetros. Um plano em conjunto com seus planos filhos formam uma *hierarquia de planos.* Na Figura 3.8 mostramos a estrutura de um plano reconhecido pelo sistema FITS-2 durante o ensino dos comandos do sistema operacional UNIX adaptado de [Woo88, pp.219-220].

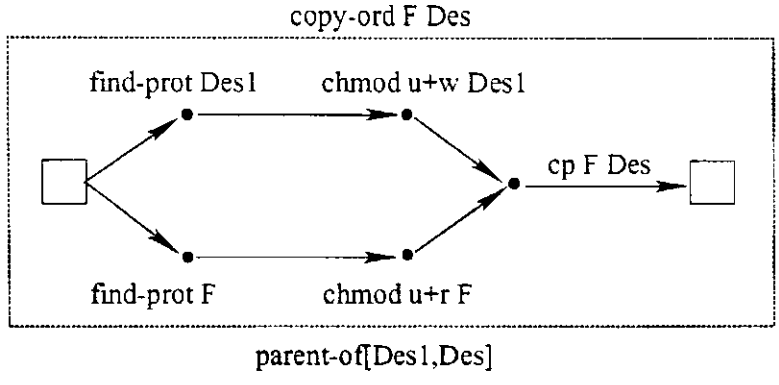


Figura 3.8: FITS-2: Hierarquias de planos.

No diagrama um plano chamado copy-ord é representado. Este plano, com os argumentos F e Des, representa o processo de cópia de um arquivo F para o diretório Des do sistema de arquivos do UNIX. Há cinco elementos de expansão: find-prot Des1, find-prot F, chmod u+r F, chmod u+w Des1 e cp F Des. O subplano find-prot representa o ato de determinar a proteção de um arquivo. Os subplanos chmod u+r e cp são comandos do sistema operacional UNIX. Há ainda uma restrição envolvendo os parâmetros dos elementos de expansão: parent-of(Des1,Des). Esta restrição faz com que o diretório Des esteja localizado no diretório Des1.

Uma vez que praticamente não existem técnicas gerais, a representação de planos é muito dependente da técnica utilizada para realizar o reconhecimento dos planos de aprendizes durante suas interações com um ambiente de aprendizagem por computador.

Algumas dessas técnicas serão discutidas na seção 3.4.2.

Uma outra observação importante diz respeito à duração temporal da representação dos planos de um aprendiz. No geral, os planos do aprendiz são representados e mantidos por pouco tempo. A idéia de modelar os planos de um aprendiz, cuja utilidade foi discutida na seção 3.2.3, consiste mais em *reconhecer* os planos do aprendiz do que em manter um registro explícito e permanentemente dos planos em um modelo do aprendiz. Assim, é mais adequado falarmos de representações temporárias de planos utilizadas pelo ambiente de aprendizagem para interpretar o comportamento do aprendiz.

3.3.3 Estereótipos

Um *estereótipo* representa uma coleção de características e atributos que geralmente se aplicam a uma classe de pessoas [Ric89]. Um exemplo típico de estereótipo é o termo *novato*; quando um professor considera um aluno novato, imediatamente atribui a este aluno um conjunto de características tais como: seu conhecimento é praticamente nulo; irá ter dificuldades em resolver problemas difíceis; irá necessitar de dicas durante a resolução de problemas. Estas atribuições que em geral mostram-se válidas, mas que vez ou outra são equivocadas, irão guiar o comportamento do professor frente ao aluno.

Na modelagem do aprendiz, a idéia de estereótipo tem sido utilizada tanto como método de representação de informações quanto como método de aquisição de novas informações para um modelo do aprendiz[14] [Pai95, PS94, Bea94].

Como método de representação, uma das principais vantagens dos estereótipos é a possibilidade de representar de forma estruturada outras informações além do estado cognitivo dos aprendizes tais como preferências, motivações e aptidões.

Os estereótipos geralmente são organizados em hierarquias. A Figura 3.9 mostra um exemplo de hierarquia de estereótipo. Em uma hierarquia de estereótipos todas as características atribuídas a um estereótipo de um nível superior são propagadas a estereótipos de nível inferior ligados por um caminho ao longo da hierarquia. Dessa forma, na Figura 3.9, um aprendiz I-1 tem todas as características de um aluno Intermediário mais alguns atributos específicos de seu estereótipo.

[14]Em realidade a idéia de estereótipos surgiu e tem sido mais explorada na área de pesquisa geral conhecida por *modelagem de usuário*; ver [Ric79, Ric89]. A pesquisa em modelagem do usuário se preocupa não apenas com a modelagem de usuários de ambientes de aprendizagem mas com a modelagem de usuários dos sistemas de computador em geral.

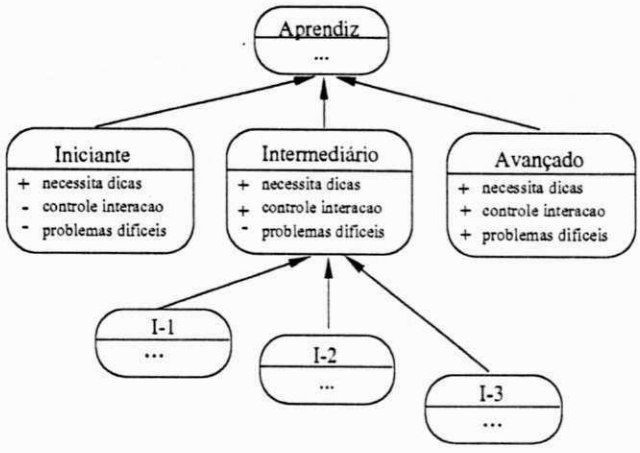


Figura 3.9: Uma hierarquia de estereótipos.

O uso mais difundido de estereótipos, no entanto, é como um método de coletar e inferir novas informações para modelos de apredizes usuários.

3.4 A Aquisição do Modelo

A *aquisição* durante a modelagem do estudante é o processo através do qual o sistema obtém informações que darão origem a novos conhecimentos sobre o aprendiz, seu estado de conhecimento, planos, etc., a serem incorporados ao atual modelo do aprendiz. Ver Figura 3.10.

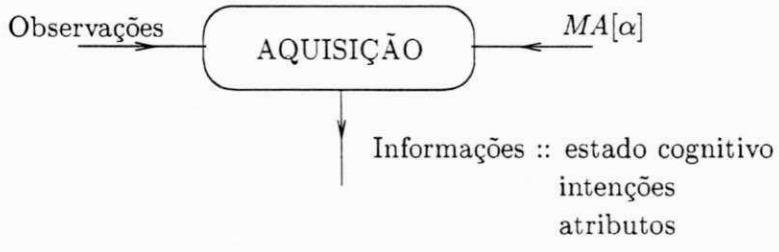


Figura 3.10: A aquisição do modelo do aprendiz.

A aquisição pode ser realizada de forma *direta* ou de forma *indireta* [PS95]. Na aquisição direta as informações são obtidas mediante um diálogo entre o sistema e o aprendiz no qual o sistema solicita explicitamente o que necessita. A aquisição direta gera novos fatos sobre o aprendiz independentemente do atual estado do modelo do aprendiz. Na aquisição indireta as informações sobre o aprendiz são inferidas a partir de seu *comportamento observado*, ou seja, suas ações realizadas na interface do sistema. A aquisição indireta gera *hipóteses*, ou seja informações incertas, sobre o aprendiz que dependem do atual estado do modelo do aprendiz.

3.4.1 Aquisição Direta

Em geral a aquisição direta é usada para inquirir o aprendiz sobre seu conhecimento inicial, suas motivações, preferências, aptidões, objetivos e outras informações gerais durante a *inicialização* do modelo do aprendiz.

Classificação em estereótipos. A implementação mais simples da idéia de aquisição direta consiste em exibir um formulário ao aprendiz onde o sistema irá solicitar cada unidade de informações que necessita saber. Em geral esse é um processo demorado para o aprendiz. Um processo mais interessante seria exibir um formulário com poucas perguntas e com o auxílio de estereótipos inferir uma série de outras informações a serem incluídas no modelo do aprendiz. Chamamos este metódo de aquisição de *classificação em estereótipos* [Pai95].

A classificação em estereótipo é o método empregado pelo sistema ANATOM-TUTOR um sitema tutor para ensino de anatomia [Bea94].

A classificação em estereótipos é um método bastante eficiente de inicializar um modelo do aprendiz. Sua principal desvantagem, no entanto, é tornar a aquisição direta sujeita a *incertezas*. As informações presentes em um estereótipo aplicam-se a maioria dos indivíduos de uma classe, não a todos.

O uso de estereótipos não está restrito a uma forma de aquisição direta usada apenas para inicializar o modelo do aprendiz. Podem ainda ser empregados com certas restrições como um método de aquisição indireta a partir do qual o modelo do aprendiz será constantemente atualizado. A principal restrição está no fato de que a aquisição por estereótipos não fornece uma análise detalhada do aprendiz [Sel94a, p.303]. Além disso a aquisição por estereótipos pode levar a introdução de *inconsistências* no modelo do aprendiz. Este assunto é discutido na seção 3.5.3.

3.4.2 Aquisição Indireta

Em geral a aquisição indireta é usada para inferir o estado de conhecimento e os planos de um aprendiz ao longo do processo de aprendizagem. No primeiro caso a aquisição tem sido chamada de *diagnóstico cognitivo* [Rag96],[DS92b],[Vas91], [Wen87, cap.17]; no segundo *reconhecimento de planos* [AZN98, GK95, Woo88].

Diagnóstico cognitivo

O diagnóstico cognitivo é utilizado principalmente em ambientes de aprendizagem baseada em resolução de problemas. Nestes ambientes o comportamento observado consiste nos passos tomados pelo aprendiz durante a resolução de um problema que foram "escritos" na interface do sistema[15]. As informações obtidas em geral dizem respeito a que conhecimentos foram utilizados pelo aluno ao resolver um problema (o que o aluno sabe) e quais as causas de possíveis erros (falta de conhecimento, mal-entendidos).

A grande maioria das técnicas de diagnóstico cognitivo reportadas na literatura especializada pode ser distribuída em quatro categorias [Rag96], [Wen87, cap.17]. São elas: *reconstrução de passos*, *análise de passos*, *reconhecedores de itens pedagógicos*, e *aprendizagem de máquina*. Esta divisão leva em conta três fatores principais: a *estratégia básica de diagnóstico*, a *forma de codificação do conhecimento aprendido* e a *granularidade do comportamento observado*.

Quanto à estratégia básica de diagnóstico há duas possibilidades: o *diagnóstico dirigido por modelos* e o *diagnóstico dirigido por dados de observação*. No primeiro parte-se de um modelo de domínio pré-definido (resolução de problemas, itens pedagógicos, catálogo de mal-entendidos) e tenta-se identificar conhecimentos (pertencentes ao modelo de domínio) que expliquem o comportamento observado do aprendiz. Na segunda estratégia parte-se do comportamento observado e tenta-se inferir ou induzir conhecimentos (não necessariamente pertencentes a um modelo de domínio) que expliquem as ações tomadas pelo aprendiz. A estratégia básica de diagnóstico é o que distingue as técnicas chamadas de aprendizagem de máquina das outras. Naquela a estratégia é dirigida por dados de observação; nestas a estratégia é dirigida por modelos.

Quanto à forma de codificação do conhecimento aprendido há também duas possibilidades: *conhecimento de resolução de problemas* e *conhecimentos pedagógicos*. No primeiro caso, uma representação do conhecimento que o aprendiz utiliza para resolver problemas é adquirida; esta representação é dita conter a *funcionalidade* do conhecimento do aprendiz, quer dizer, pode ser utilizada pelo sistema para simular o comportamento do aprendiz. No segundo caso, uma *descrição* do conhecimento do aprendiz em termos de *itens pedagógicos* (e.g., elementos de um *curriculum*) é adquirida. A forma de codificação é o que distingue as técnicas chamadas reconhecedores de itens pedagógicos das outras. Naquela adquirem-se conhecimentos pedagógico; nestas adquirem-se conhecimentos de resolução de problemas.

[15] Passos "não-escritos" na interface do sistema são ditos *comportamento não observado*.

Quanto à granularidade, o comportamento observado pode ser completo ou parcial. No caso completo todas as ações tomadas pelo aprendiz são "escritas" na interface do sistema. No caso parcial existem comportamentos observados e comportamentos não observados. A granularidade de comportamento é o que distingue as técnicas chamadas de análise de passos das outras. Naquela o comportamento observado é assumido como completo; nestas o comportamento observado é parcial.

Na Figura 3.11 sumarizamos essas observações. As setas representam a estratégia básica de diagnóstico; MP e MRP abreviam respectivamente modelo pedagógico e modelo de resolução de problemas e F e G abreviam respectivamente granularidade fina e granularidade grossa do comportamento observado.

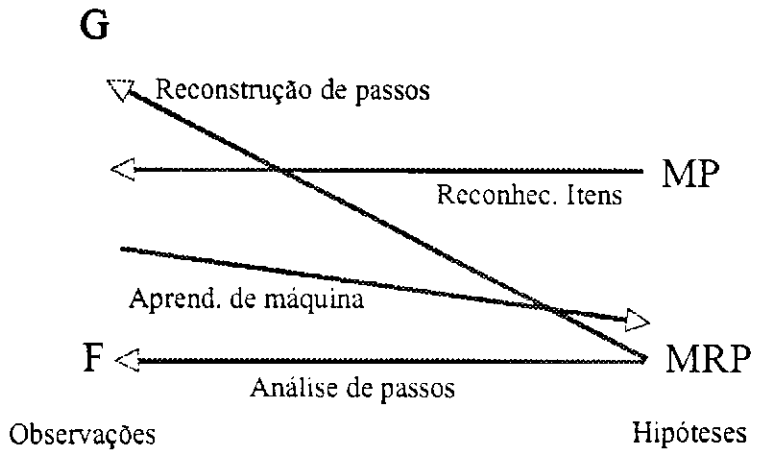


Figura 3.11: O diagnóstico cognitivo.

Reconstrução. Em um caso extremo, dado um problema o aluno devolve ao sistema apenas a resposta sem passos intermediários. Neste caso o processo de inferir o conhecimento utilizado pelo aluno, de forma correta ou incorreta, refazendo seus possíveis passos ao resolver o problema, tendo como referência um modelo de resolução de problemas (modelo de domínio, possivelmente estendido com um catálogo de mal-entendidos) é dito reconstrução dos passos de resolução.

A reconstrução de passos é um problema difícil computacionalmente. Um cálculo rápido ressalta as dificuldades. Suponhamos que um modelo de resolução de problema contenha r regras[16] passíveis de serem aplicadas na resolução de um problema. Dado que um problema é resolvido corretamente pelo aluno em p passos (aplicações de regras) então o sistema terá de reconstruir em meio a r^p possíveis combinações de regras o caminho seguido pelo aluno ao resolver o problema. Pior, dado que o aluno tenha errado o problema e que a causa seja um ou mais mal-entendidos; havendo m maneiras

[16] Por exemplo, uma base de conhecimentos formada por regras de produção.

nas quais cada uma das r regras possa ser mal entendida pelo aluno o espaço de busca sobe para $(m(n+1))^p$ [Sel90].

Uma solução para a explosão combinatória do processo de reconstrução – como para todo problema de busca – é o uso de heurísticas. Uma outra solução utilizada no sistema LMS [Sle82] é sistematizar o processo de reconstrução; este sistema utiliza um algoritmo de reconstrução onde o espaço de busca é reduzido a $p(n+1)$ caminhos possíveis. Uma terceira solução é fazer com que o aprendiz explicite alguns passos intermediários.

Além do sistema LMS outros sistemas que implementam a estratégia de diagnóstico por reconstrução são o sistema PIXIE [Wen87, cap.9] e os sistemas BUGGY [Bur82].

Análise de passos. Em outro ponto extremo, dado um problema o aluno devolve ao sistema todos os passos básicos que fez ao resolver o problema. Neste caso o sistema é capaz de inferir o conhecimento utilizado pelo aluno assumindo que cada passo de resolução corresponde à aplicação de uma unidade de conhecimento (ou mal-entendido) de um modelo de resolução de problemas ideal (modelo de domínio, possivelmente estendido com um catálogo de mal-entendidos). Esta estratégia de diagnóstico é dita análise dos passos de resolução[17].

A análise de passos é também uma forma de conter a explosão combinatória da estratégia de reconstrução. No entanto, o aluno é obrigado solucionar problemas explicitando todos os passos; dependendo da granularidade de cada passo esta estratégia pode forçar o aprendiz a um processo de resolução de problemas tedioso e desajeitado.

Um exemplo típico de sistema que utiliza a estratégia de análise de passos é o *Lisp tutor* [RAF85]. Um outro exemplo é o sistema WUSOR [Wen87, cap.7].

Reconhecedores de itens pedagógicos. Um item pedagógico pode ser definido como qualquer elemento de um *curriculum* (modelo de domínio) cuja participação em decisões durante a resolução de problemas pode ser reconhecida e discutida [Wen87, p.380]. Assim, dada a resolução de um problema uma terceira estratégia de diagnóstico é identificar que itens pedagógicos foram ou não utilizados pelo aluno através de reconhecedores de itens pedagógicos.

Um exemplo já clássico desta abordagem de diagnóstico é o sistema WEST [Wen87, pp.127-133]. Neste sistema - um jogo educacional para exercitar habilidades aritméticas - quando e a vez do aprendiz jogar o sistema gera uma lista das possíveis jogadas. Caso a jogada do aprendiz não seja a melhor prevista pelo sistema o processo de diagnóstico inicia. Por meio de reconhecedores de itens pedagógicos as decisões do aprendiz são

[17] Tradução livre do termo inglês *Model Tracing*

analisadas; o mesmo acontecendo com as jogadas do sistema que são melhores que a jogada do aprendiz. Ao final o sistema dispõe de uma lista dos itens (habilidades aritméticas) que o aprendiz não levou em consideração ou falhou ao usar em sua jogada.

Uma das vantagens do uso de reconhecedores de itens pedagógicos sobre as estratégias anteriores é o fato de que não há necessidade de reconstruir ou analisar exatamente como o conhecimento utilizado pelo aprendiz o levou a responder um problema. No entanto, a construção de reconhecedores, ou seja, regras que mapeiem ações e respostas do aprendiz a itens pedagógicos pode não ser uma tarefa trivial [Wen87, p.380]. Segundo Ragnemalm "um estudo de como reconhecedores são construídos seria uma contribuição para a área" [Rag96, p.111].

Além do WEST outros sistemas que utilizam essa abordagem são o sistema BIP [Wen87, pp.108-111] e o sistema SCENT [GMM89].

De forma integrada, as três estratégias de diagnóstico acima apresentadas podem ser incorporadas no seguinte esquema geral de diagnóstico cognitivo:

1. Verificar a resposta do problema tendo como referência um modelo de domínio pré-definido (resolução de problemas, pedagógico). Se correta atribuir ao aprendiz o conhecimento (pertencente ao modelo de domínio) necessário a resolução do problema. Senão,

2. Isolar erros:

   (a) reconstruindo os passos da resolução;

   (b) ou, analisando os passos da resolução;

   (c) ou, reconhecendo itens pedagógicos;

3. Explicar erros:

   (a) falta de conhecimento em relação ao modelo de domínio; (sobreposição)

   (b) mal-entendidos pertencentes a um catálogo de mal-entendidos; (perturbações)

Neste esquema, apesar de separados, os três passos podem ser, e na prática são, realizados concomitantemente. Notemos ainda que no último passo são geradas hipóteses que tentam explicar possíveis erros do aprendiz; estas explicações dependem do tipo de representação adotada no modelo do aprendiz.

Aprendizagem de máquina. O termo *aprendizagem de máquina* de forma geral é usado para referenciar "uma área de pesquisa na Inteligência Artificial preocupada em desenvolver teorias computacionais de processos de aprendizagem e construir máquinas que aprendem" [GS88, p.179]. De forma específica, empregamos o termo para designar estratégias de diagnóstico cognitivo que utilizam algoritmos da área de aprendizagem de máquina para inferir o estado de conhecimento de um aprendiz tendo como referência básica o seu comportamento observado ao resolver problemas.

Uma dessas estratégias é a utilizada no sistema ACM [LOS84], [Wen87, 210-218] chamada de *indução de condições*. Em linhas gerais a indução de condições utiliza técnicas de aprendizagem de máquina conhecidas como *aprendizagem por exemplos* ou *aprendizagem indutiva* [GS88]. Dado um número considerável de caminhos de solução de problemas, os exemplos, a aprendizagem indutiva consegue gerar regras que explicam o comportamento observado do aprendiz.

Outras técnicas de aprendizagem de máquina já utilizadas em estratégias de diagnóstico cognitivo são a *revisão de teorias* (*theory revision*) [BM92] e *focalização* (*focussing*) Gilmore:Self:88. Ver também [SS96].

Diferentemente das estratégias anteriores, as estratégias baseadas em aprendizagem de máquina podem inferir conhecimentos do aprendiz não presentes em um modelo de domínio pré-estabelecido. No entanto, um possível problema é a instabilidade do modelo cognitivo gerado [Rag96, p.102]. Quando um aluno aprende algo novo seu comportamento é instável, sendo os novos conhecimentos às vezes aplicados de forma correta e às vezes não. Disso podem surgir contradições, inconsistências no modelo cognitivo sendo inferido. Na seção 3.5.3 falaremos sobre a manutenção de consistência em modelos cognitivos[18].

Abordagens híbridas. As quatro abordagens para diagnóstico cognitivo apresentadas não são incompatíveis e podem ser combinadas. Em realidade, na grande maioria de sistemas já desenvolvidos o que existe é um misto das abordagens que descrevemos.

Um exemplo é o sistema GUIDON [Cla83]. Neste sistema a abordagem de diagnóstico utiliza idéias das estratégias de reconstrução e reconhecimento de itens

[18] Um outro problema com o uso de técnicas de aprendizagem de máquina para inferir o estado de conhecimento de aprendiz apontado por [GS88] é a validade psicológica do modelo cognitivo gerado. Os autores, após analisarem alguns métodos de aprendizagem de máquina, concluem: "... para a modelagem do estudante nós devemos buscar uma base psicológica mais forte do que aquela que é comum no trabalho em aprendizagem de máquina, freqüentemente motivada mais por critérios de performance. Nenhuma das técnicas de aprendizagem de máquina existentes parece adequada para manter modelos do estudante dinâmicos."(p.195).

pedagógicos. Um outro exemplo é o próprio sistema ACM discutido anteriormente. Nele é utilizada a idéia de reconstrução de passos de resolução para produzir os exemplos nos quais são aplicadas as técnicas de aprendizagem de máquina por indução.

Reconhecimento de planos

Reconhecimento de planos é uma ampla área de pesquisa enfocando três tipos de problemas: (1) inferência de planos durante interações cooperativas, (2) entendimento de estórias e (3) reconhecimento de planos de um agente que não está ciente de que seus planos estão sendo inferidos [AZN98]. Nos dois primeiros casos, o processo de reconhecimento de planos conta com a ajuda *direta* do usuário do sistema que tenta comunicar seus planos ao sistema. No terceiro caso, o processo é feito de forma *indireta* a partir de informações incompletas.

Em modelagem do aprendiz a pesquisa em reconhecimento de planos tem enfocado principalmente o terceiro problema. A grande maioria das técnicas desenvolvidas são chamadas de *reconhecimento baseado em bibliotecas de planos.*

Biblioteca de planos. Um dos trabalhos pioneiros na modelagem de planos foi o sistema MACSYMA *Advisor* desenvolvido por Genesereth [Gen82][19]. De forma geral, o reconhecimento de planos no sistema MACSYMA *Advisor* é visto como um processo complexo de análise sintática onde as operações mentais do aprendiz ao decidir que passos realizar durante a resolução de um problema – seus planos – formam a gramática e a seqüência dos passos realizados são as palavras. Desta forma, utilizando uma gramática padrão – uma biblioteca de planos – sistema tenta realizar uma análise sintática de baixo para cima (*bottom-up*) tendo como insumo os passos do aprendiz ao resolver um problema [Gen82, pp.147-150].

Um outro exemplo de reconhecimento de planos baseado em biblioteca de planos é o sistema FITS-2 [Woo88]. No sistema FITS-2 a biblioteca de planos é organizada em torno de hierarquias de planos. Ver seção 3.3.2. Dados alguns passos de resolução de problema, o sistema tenta por em correspondência estes passos com algum plano da biblioteca de planos. Achando um plano o sistema tenta descobrir um plano mais geral que possivelmente o aluno está seguindo. Isto é conseguido procurando em uma hierarquia de planos um *plano pai* para o atual plano reconhecido. Uma vez identificado

[19] Já falamos sobre o sistema MACSYMA *Advisor* em duas ocasiões. Na seção 3.2.3 ilustrando como a modelagem de planos pode ser útil e na seção 3.3.2 mostrando como planos podem ser representados. Agora analisamos a técnica de reconhecimento utilizada por esse sistema.

não pode ser satisfatoriamente atingida devido a várias dificuldades com a abordagem de representação de mal-entendidos;

- tornar o $MA[\alpha]$ aberto ao estudante [PSH95]. Isto ajuda o estudante a refletir sobre o seu progresso além de poder opinar sobre o conteúdo do modelo;
- trabalhar em sistemas que adotam uma postura mais colaborativa O sistema não precisa "saber"tudo sobre o domínio; mas, deve ser capaz de poder ajudar o estudante a descobrir as respostas certas. Desta forma o $MA[\alpha]$ não necessita ser tão detalhado e fiel.

Uma outra forma de pensar nas limitações do processo de aquisição (e representação) é imaginá-las como fontes de incerteza e inconsistências a serem tratadas pelo módulo de manutenção.

3.5 O Tratamento de Incertezas e Inconsistências

A modelagem do estudante, como viemos considerando, apresenta-se como um processo *dinâmico* e *incerto*. É dinâmico pois tenta caracterizar algo (o estado de conhecimento do aprendiz, suas intenções, etc.) que está em constante estado de mudança. É incerto pois o máximo que irá conseguir é uma caracterização *imprecisa*, *incompleta* e as vezes *inconsistente*. Isto devido a uma série de fatores dentre os quais identificamos: ruídos no comportamento observado, limitações no processo de aquisição, limitações na linguagem de representação.

Os ruídos no comportamento observado têm como causas principais *inconsistências* no comportamento do aprendiz e *restrições* no canal de comunicação entre aprendiz e sistema. As inconsistências são um problema sério; ao resolver um problema o aprendiz pode cometer "deslizes", ou seja, errar por distração, cansaço, e outros fatores e não por falta de conhecimento ou mal-entendidos; ainda, pode acertar um problema por sorte, "chutes", ou outros fatores e não por conhecimento no assunto sendo aprendido. As restrições no canal de comunicação referem-se as dificuldades em fazer o sistema "perceber" o comportamento do aprendiz de forma completa e acurada. Como a aquisição e posterior atualização do modelo do aprendiz dependem em grande parte da percepção do comportamento do aprendiz e sua consistência com o conhecimento adquirido, os ruídos no comportamento observado são os principais fatores de incerteza durante a modelagem do aprendiz.

As limitações no processo de aquisição e na linguagem de representação dizem respeito a problemas computacionais inerentes ao processo de adquirir e representar

conhecimentos. Adquirir conhecimentos detalhados sobre o aprendiz em geral é um problema computacionalmente intratável [Sel90]. Representar conhecimentos sobre o aprendiz é quando muito uma aproximação; é incompleto e algumas vezes inconsistente. Estas limitações além de levar à incerteza quanto ao modelo do aprendiz produzido, têm levado muitos pesquisadores a desferir pesadas críticas, ou mesmo rejeitar por completo, a modelagem do estudante. Na seção anterior vimos alguns métodos de contornar algumas das principais limitações relativas a aquisição; nesta seção veremos como tornar ainda mais robusto o processo de modelagem através do tratamento de incerteza.

Chamemos de *crença* a idéia de conhecimento dinâmico e incerto. O modelo do aprendiz é então melhor caracterizado como uma *base de crenças* sujeita a constantes mudanças ao longo do tempo.

3.5.1 A Manutenção

A *manutenção* durante o processo de modelagem do estudante sob um ponto de vista geral tem como objetivo gerenciar as mudanças feitas em um modelo do aprendiz. Em grande parte, sua função é integrar as informações provindas da aquisição ao conhecimento já presente no modelo do aprendiz. Ver Figura 3.12.

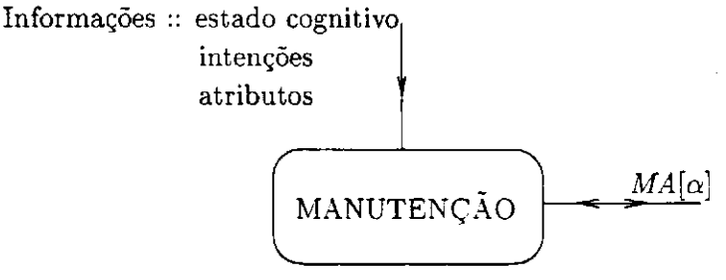


Figura 3.12: A manutenção do modelo do aprendiz.

Sendo o modelo do aprendiz uma base de crenças a manutenção terá de lidar com dois possíveis problemas: a *incerteza inerente ao processo de modelagem* e a *dinâmica de aprendizagem*. O tratamento de incerteza, como veremos, tem sido o principal propósito da grande maioria das técnicas computacionais desenvolvidas para implementar o processo de manutenção. A manutenção da dinâmica de aprendizagem só recentemente tem sido explorada por alguns pesquisadores.

No que segue revisamos algumas técnicas para implementação do processo de manutenção reportadas na literatura especializada. As técnicas são divididas em duas

grandes categorias: *técnicas numéricas* e *técnicas simbólicas.*

3.5.2 Manutenção Numérica

As técnicas de manutenção numérica têm como objetivo único lidar com a incerteza inerente ao processo de modelagem do estudante. Como o nome sugere, tais técnicas operam utilizando valores numéricos para quantificar a incerteza presente. Tradicionalmente em IA, três teorias são empregadas no desenvolvimento de técnicas computacionais para o tratamento numérico de incerteza em bases de conhecimento [BK86]. São elas: a *teoria matemática da evidência de Dempster e Shafer* [Sha76], a *teoria de conjuntos difusos* (ou lógica *fuzzy*) [KF88, Zad88] e a *teoria de probabilidades.*

Na área de modelagem do estudante a teoria da evidência de Dempster e Shafer pouco tem sido usada: de nosso conhecimento apenas o trabalho de Tokuda e Fukuda [TF93] emprega conceitos desta teoria para lidar com incerteza durante a modelagem do estudante. Sendo assim, não entraremos em detalhes sobre a teoria da evidência de Dempster e Shafer. Falaremos das teorias de conjunto difuso e probabilidades e como alguns de seus conceitos têm sido usados no desenvolvimento de técnicas para tratamento de incerteza durante o pocesso de manutenção.

Teoria de conjuntos difusos

A teoria de conjuntos difusos têm como idéia fundamental a atribuição de *graus de pertinência*, valores no intervalo $[0, 1]$, a elementos de um conjunto. Matematicamente a atribuição de graus de pertinência funciona como uma generalização da noção de função característica de um conjunto $A \subseteq U$ (ver seção 3.3.1 e [KF88, cap.1]). Em símbolos:

$$\mu_A : U \rightarrow [0, 1]$$

Intuitivamente, quanto mais próximo de 1 for $\mu_A(u)$, mais u é dito pertencer a A. A é dito ser um *conjunto difuso*, U o *universo de discurso* e μ_A a *função de pertinência* de A.

A teoria de conjuntos difusos é especialmente concebida para lidar com situações e problemas que envolvem *incertezas lingüísticas*, ou seja, noções, conceitos ou termos vagos, não bem definidos. Um exemplo típico que ilustra claramente *duas formas básicas de usar conjuntos difusos no tratamento de incertezas lingüísticas* é a nossa apreensão e emprego intuitivo da noção de *velho*. Imaginemos duas situações. Na primeira uma pessoa diz sua idade, digamos 70 anos, e alguém exclama: nossa como *é* "velha" esta pessoa! Na segunda situação uma pessoa desconhecida aparentando "idade

avançada" chega e alguém fica imaginando: qual será a *possível* idade desta pessoa? Em ambas as situações podemos reformular os problemas básicos representando as noções de "velho" e "avançado" como conjuntos difusos tendo como universo de discurso um dado conjunto de idades. No primeiro caso utilizamos o conjunto difuso "velho", os graus de pertinência, para dizer até que ponto alguém irá chamar de velha uma pessoa com x anos. Formalizamos assim *a internalização de uma idéia vaga e o pertencimento de elementos a esta idéia.* No segundo caso utilizamos o conjunto difuso "avançado" para resumir observações sobre a aparência de uma pessoa e assim imaginar, através de graus de pertinência, sua possível idade. Formalizamos assim *a chegada ou aquisição de informações vagas, incompletas e o que delas é possível concluir.*

No contexto de tratamento de incerteza durante o processo de manutenção estes dois usos básicos dão origem a duas categorias de técnicas de manutenção: as técnicas de manutenção baseadas em *sobreposição difusa* e as técnicas de manutenção baseadas em *variáveis lingüísticas.*

Sobreposição difusa. Sobreposição difusa é o nome que damos à representação do modelo cognitivo $MC[\alpha]$ como um conjunto difuso tendo como universo de discurso o modelo de domínio MD. Em símbolos:

$$I_{MC[\alpha]} : MD \to [0,1]$$

Nesta abordagem, $I_{MC[\alpha]}$ é intuitivamente vista como uma idéia vaga, imprecisa que o sistema faz sobre o conhecimento do aprendiz. Dado um conceito $\varphi \in MD$, $I_{MC[\alpha]}(\varphi)$ nos diz até que ponto o sistema "acha" que φ faz parte do estado cognitivo do aluno α.

A manutenção baseada em sobreposição difusa consiste então em mecanismos para inicializar e atualizar os valores $I_{MC[\alpha]}(\varphi) \in [0,1]$. Um exemplo deste tipo de mecanismo é a seguinte função de manutenção proposta por Gisolfi, Dattalo e Balzano [GDB92]:

$$Manutenção : (k, w) \mapsto \begin{cases} w^{1/2^k} & 0 < k \leq i \\ w^{2^{|k|}} & j \leq k < 0, w \in (0,t) \\ w^{|k|} & j \leq k < 0, w \in [t,1) \end{cases}$$

Esta função funciona a partir das seguintes suposições. Ao aluno é passado um problema envolvendo um conceito $\varphi \in MD$. Dada a resposta do aluno ao problema o processo de diagnóstico irá gerar uma hipótese numérica sobre sua correção: atribuirá um inteiro $0 < k \leq i$ caso a resposta esteja correta; atribuirá um inteiro $j \leq k < 0$

caso a resposta seja considerada errada. Resposta com maiores valores absolutos são tidas como mais corretas ou erradas do que respostas com menores valores. Os inteiros i, j são, respectivamente, o máximo e mínimo valores atribuídos a respostas corretas ou erradas.

Com base no valor k a manutenção irá incrementar ou decrementar o valor w associado ao conceito φ. Para isto usará respectivamente a função *raiz n-ésima* $w^{1/2^n}$ e as funções *potência n-ésima* $w^{2^k}, w^{|k|}$, cujo comportamento, segundo os autores, é tal que pode ser verdadeiramente visto como a curva de aprendizadem do estudante [GDB92, p.331].

O incremento é feito de forma única; o decremento leva a duas opções dependentes de um parâmetro $0 \leq t \leq 1$ previamente estabelecido. Quando o aprendiz é dito ainda não saber o conceito φ, ou seja, quando $w \in (0, t)$, o decremento é realizado pela função w^{2^k}; quando o aprendiz é dito já ter conhecimento sobre φ, ou seja, quando $w \in [t, 1)$, o decremento é realizado pela função $w^{|k|}$. Notemos que na segunda opção o decremento é menor que na primeira. Isto representa o fato de que quando o aluno é dito saber um conceito, eventuais diagnósticos negativos devem ser vistos mais como causados por ruídos no comportamentamento do aprendiz e limitações computacionais do que propriamente pelo desaprender por parte do aluno.

Quanto a escolha t, os autores observam que no intervalo $[0.7, 0.9]$ irá existir um valor a partir do qual as derivadas das funções de incremento e decrementos aproximam-se de zero. Isto modela o fato de que a partir de um patamar de conhecimento poucas ou quase nenhuma variação deve ocorrer.

Variáveis lingüísticas. De forma breve, uma variável lingüística é uma variável cujos valores são conjuntos difusos representando palavras ou sentenças de uma linguagem natural ou artificial. No exemplo acima "idade" é uma variável lingüística e "avançada" um de seus valores; um outro valor poderia ser "de estudante do 2^0 grau".

Variáveis lingüísticas podem ser utilizadas para representar incertezas em um modelo cognitivo da seguinte forma:

$$I_{MC[\alpha]} : MD \rightarrow \{\mu_A \mid \mu_A : U \rightarrow [0, 1]\}$$

Ou seja, cada conceito $\varphi \in MD$ é visto como uma variável lingüística. Nesta abordagem cada conjunto difuso $I_{MC[\alpha]}(\varphi)$ resume *possibilidades* de conhecimento de um aprendiz α sobre um dado conceito φ. Possibilidades estas estimadas a partir de informações vagas, imprecisas ou incompletas sobre o próprio conceito φ e sobre seu uso pelo aprendiz como percebido pelo processo de aquisição.

A manutenção baseada em variáveis lingüísticas consiste então em mecanismos para inicializar e atualizar os conjuntos difusos $I_{MC[\alpha]}(\varphi)$. Um exemplo deste tipo de mecanismo são as *regras difusas se-então* utilizadas no sistema KNOME [Jam96, pp.227-232].

No sistema KNOME os conjuntos difusos $I_{MC[\alpha]}(\varphi)$ têm como universo de discurso um conjunto com quatro elementos: U = { NOVATO , INICIANTE , INTERMEDIÁRIO , ESPECIALISTA }. Cada elemento representando um nível de assimilação de um dado conceito pelo aprendiz. Associados aos quatro níveis de assimilação podem estar 9 valores: $\{0.0, 0.125, 0.25, 0.375, 0.5, 0.625, 0.75, 0.875, 1.0\}$. Cada valor representando uma possibilidade numérica sobre o conhecimento do aprendiz em relação a um dado conceito φ. O valor 0.0 representa impossibilidade; 0.5 indica incerteza e 1.0 total possibilidade. Os valores iniciais são $I_{MC[\alpha]}(\varphi)$(NOVATO) = 0.5 , $I_{MC[\alpha]}(\varphi)$(INICIANTE) = 0.625 , $I_{MC[\alpha]}(\varphi)$(INTERMEDIÁRIO) = 0.5 e $I_{MC[\alpha]}(\varphi)$(ESPECIALISTA) = 0.5.

Uma regra difusa se-então utilizada pelo sistema KNOME tem a seguinte forma:

- **SE** o conceito φ é SIMPLES e o aprendiz usou corretamente φ
  ENTÃO agora é MAIS PROVÁVEL que o aprendiz seja ESPECIALISTA no conceito φ

Onde SIMPLES é o grau de dificuldade de um dado conceito φ e MAIS PROVÁVEL é um fator de incremento que deve ser adicionado ao atual valor associado ao nível ESPECIALISTA; ou seja, $I_{MC[\alpha]}(\varphi)$(ESPECIALISTA)+ = MAIS PROVÁVEL[22]. No total existem três classes nas quais o conceitos se enquadram e nove fatores de incremento dando origem a um total de 12 regras de atualização para os conjuntos difusos $I_{MC[\alpha]}(\varphi)$ quando um aprendiz utiliza um dado conceito φ corretamente. No quadro da Figura 3.13 resumimos estas 12 regras mostrando os fatores de incremeto em sua forma numérica. Notemos que MAIS PROVÁVEL corresponde a um valor de incremento 0.25; outros valores são: POUCO PROVÁVEL = 0.125, POUCO IMPROVÁVEL = − 0.125, IMPOSSÍVEL = − 0.5.

A manutenção baseada em variáveis difusas é apenas parte do tratamento de incertezas no sistema KNOME. Aliado à este método, o sistema ainda emprega um esquema de manutenção baseada em sobreposição difusa. Este também atualizado por regras difusas se-então [Jam96, pp.227-232].

[22]Onde $x+ = a$ quer significar $x := x + a$.

SE	o aprendiz usou corretamente φ e o conceito φ	é	SIMPLES	MÉDIO	COMPLEXO
ENTÃO	$I_{MC[\alpha]}(\varphi)$(ESPECIALISTA)	$+=$	+0.25	+0.25	+0.125
	$I_{MC[\alpha]}(\varphi)$(INTERMEDIÁRIO)	$+=$	+0.25	+0.125	− 0.125
	$I_{MC[\alpha]}(\varphi)$(INICIANTE)	$+=$	+0.125	− 0.125	− 0.5
	$I_{MC[\alpha]}(\varphi)$(NOVATO)	$+=$	− 0.125	− 0.5	− 0.5

Figura 3.13: Regras difusas se-então utilizada pelo sistema KNOME.

Além do sistema KNOME e da função de manutenção proposta por Gisolfi, *et al.*, [GDB92], dois outros trabalhos de pesquisa que têm utilizado a teoria de conjuntos difusos no tratamento de incertezas em modelos do aprendiz são o sistemas SYPROS e IFTRED [Jam96].

Teoria de probabilidades

A teoria de probablidades é um modelo matemático para lidar com *experimentos aleatórios*. Experimentos onde ocorrem *eventos* que independem das condições iniciais nas quais o experimento foi realizado. A idéia central da teoria de probabilidades é a noção de *probabilidade* associada a um evento de um experimento aleatório. Um número real entre 0 e 1 que indica a crença de uma pessoa na ocorrência de um dado evento em um experimento aleatório[23].

Outras idéias importantes são o conceito de *probabilidade condicional*, *variáveis aleatórias* e *distribuição de probabilidades*. Estes conceitos, em conjunto com a *regra de Bayes*, formam os recursos básicos da teoria de probabilidades para representação de incertezas e raciocínio a partir de crenças em bases de conhecimento [Ste95].

As idéias da teoria de probabilidades têm sido muito empregadas no tratamento de incertezas na modelagem do aprendiz. Em especial na etapa de manutenção do modelo do aprendiz. No que segue consideramos duas classes de técnicas que utilizam conceitos da teoria de probabilidades: as técnicas de manutenção baseadas em *variáveis aleatórias independentes* e as técnicas de manutenção baseadas em *redes bayesianas*.

Variáveis aleatórias independentes. Tendo em vista a incerteza inerente ao processo de modelagem do aprendiz podemos considerar as informações obtidas pelo processo de aquisição como o *espaço amostral* de um experimento aleatório. Assim, uma

[23] Definição subjetiva de probabilidade. Ver em [Dan97] outras definições.

maneira de representar incertezas em um modelo cognitivo $MC[\alpha] \subset MD$ é imaginar cada $\varphi \in MC[\alpha]$ como uma variável aleatória (independente) e armazenar explicitamente uma distribuição de probabilidade para cada φ. Uma variável aleatória cujo domínio é o espaço amostral do processo de aquisição, e a imagem é um conjunto $\{na_1, ..., na_n\}$; cada valor na_i representando um nível de assimilação de φ pelo aprendiz α; onde índices maiores indicam um maior nível de assimilação. Em símbolos:

$$I_{MC[\alpha]} : MD \rightarrow \{\langle p_1, ..., p_n \rangle \mid p_i \in [0,1], \sum_{1 \leq i \leq n} p_i = 1\}$$

Onde $I_{MC[\alpha]}(\varphi).p_i = P[\varphi = na_i]$: ou seja a probabilidade de φ encontrar-se no nível de assimilação i.

A manutenção baseada em variáveis aleatórias (independentes) consiste então em mecanismos para inicializar e atualizar as distribuições de probabilidades $I_{MC[\alpha]}(\varphi)$. Um exemplo deste tipo de mecanismo é o modelo de aprendizagem simples baseado em dois estados utilizado nos Tutores de Programação ACT (APT) [CA95, CR92].

Nos sistemas APT o conhecimento a ser aprendido, $\varphi \in MD$, é codificado por meio de regras de produção. Assume-se que cada regra no modelo cognitivo $\varphi \in MC[\alpha]$ (variável aleatória φ) em dado momento ou está no estado *não-aprendido* ($\varphi = na_1$) ou no estado *aprendido* ($\varphi = na_2$). Uma regra pode mudar do estado não-aprendido para o estado aprendido em cada oportunidade que o aprendiz tem de aplicar a regra ($P[\varphi = na_2 \mid \varphi = na_1] = \mathbf{p(T)} > 0$). Não há esquecimento; regras não podem mudar do estado aprendido para o estado não-aprendido ($P[\varphi = na_1 \mid \varphi = na_2] = 0$). Assume-se ainda que cada regra no modelo cognitivo em dado momento ou é aplicada de forma *correta* ou é aplicada de forma *errada* pelo aluno, informações obtidas pelo processo de aquisição. Se uma regra está no estado aprendido é possível que o aprendiz venha a cometer um *deslize* e aplicar a regra de forma errada ($P[errada \mid \varphi = na_2] = \mathbf{p(S)} > 0$). Se uma regra está no estado não-aprendido é possível que o aprendiz faça um "chute" e venha a aplicar a regra corretamente ($P[correta \mid \varphi = na_1] = \mathbf{p(G)} > 0$).

A idéia então é manter uma estimativa da probabilidade de o aprendiz ter assimilado uma dada regra condicionada a sua performance ao aplicar a regra. Isto é feito por meio da seguinte equação:

$$p(L_k) = p(L_{k-1} \mid E_k) + (1 - p(L_{k-1} \mid E_k)) \cdot p(T)$$

onde, $p(L_k) = P[\varphi = na_2 \mid E_k, ..., E_1]$, ou seja a probabilidade de uma regra c ter sido assimilada após o sistema ter coletado k evidências a respeito de sua utilização, onde cada evidência pode ser ou $E_i = correta$ ou $E_i = errada$.

Assim, a probabilidade de que uma regra esteja no estado aprendida após a *k-ésima* oportunidade que um aprendiz tem de aplicar a regra, $p(L_k)$, é a soma de duas probabilidades: (1) a probabilidade de que a regra já estava no estado aprendida condicionada a atual evidência, $p(L_{k-1} \mid E_k)$; e (2) a probabilidade de que a regra venha a mudar para o estado aprendido caso já não esteja nele, $(1 - p(L_{k-1} \mid E_k)) \cdot p(T)$.

As probabilidades $p(L_{k-1} \mid E_k)$ são calculadas da seguinte forma:

$$p(L_{k-1} \mid E_k) = \begin{cases} \dfrac{(1-p(S)) \cdot p(L_{k-1})}{(1-p(S)) \cdot p(L_{k-1}) + p(G) \cdot p(\neg L_{k-1})} & \text{quando } E_k = \textit{correta} \\ \dfrac{p(S) \cdot p(L_{k-1})}{p(S) \cdot p(L_{k-1}) + (1-p(G)) \cdot p(\neg L_{k-1})} & \text{quando } E_k = \textit{errada} \end{cases}$$

onde, $p(\neg L_{k-1}) = P[\varphi = na_1 \mid E_{k-1}, ..., E_1]$, ou seja a probabilidade de uma regra φ não ter sido assimilida após $k-1$ evidências a respeito de sua utilização.

Essa última equação é obtida por meio da regra de Bayes[24] escrevendo $p(E_k \mid L_{k-1})$ e $p(E_k \mid \neg L_{k-1})$ a partir dos parâmetros de performance $p(S)$ e $p(G)$ discutidos acima: ver [Rey96, pp. 599-601] para mais detalhes. Estes, em conjunto com os parâmetros de aprendizagem $p(T)$, também discutido acima, e $p(L_0)$, a probabilidade de que uma regra esteja no estado aprendida antes de quaisquer aplicações, formam os quatro parâmetros do processo de manutenção dos sitemas APT. Em geral, os valores desses parâmetros devem ser estabelecidos empiricamente, podendo variar para cada regra $\varphi \in MC[\alpha]$. No entanto, Corbert e Anderson apresentam um estudo onde esses valores foram mantidos constantes e iguais a $p(L_0) = 0.5$, $p(T) = 0.4$, $p(G) = p(S) = 0.2$, para 21 regras em um modelo cognitivo [CR92].

Um outro exemplo de manutenção baseada em variáveis aleatórias (independentes) são os *vetores de crenças* utilizados inicialmente no sistema SHERLOCK II [KLEG94], e posteriormente nos sistemas ATS/ML-Modeler [GDS95] e MFD [BSW97].

Um vetor de crença é uma distribuição de probabilidade de uma variável aleatória $\varphi \in MC[\alpha]$ que pode assumir n níveis de assimilação[25]; ou seja, $I_{MC[\alpha]}(\varphi) = \langle p_1, ..., p_n \rangle$. Ao contrário do mecanismo discutido acima, que utiliza equações derivadas a partir da teoria de probabilidades, os valores em um vetor de crença são atualizados por meio

[24] Regra de Bayes: $P(L \mid E) = \dfrac{P(E \mid L) \cdot P(L)}{P(E \mid L) \cdot P(L) + P(E \mid \neg L) \cdot P(\neg L)}$

[25] No trabalho original de [KLEG94] são utilizados 5 níveis; nos trabalhos [GDS95, BSW97] são empregados 7 níveis.

de regras heurísticas de *incremento* e *decremento* de crenças:

$$\text{incremento}: p_i = \begin{cases} p_1 - p_1 \cdot c & i = 1 \\ p_i - p_i \cdot c + p_{i-1} \cdot c & 2 \leq i \leq n-1 \\ p_n + p_{n-1} \cdot c & i = n \end{cases}$$

$$\text{decremento}: p_i = \begin{cases} p_1 + p_2 \cdot c & i = 1 \\ p_i - p_i \cdot c + p_{i+1} \cdot c & 2 \leq i \leq n-1 \\ p_n - p_n \cdot c & i = n \end{cases}$$

onde c é uma constante que controla a taxa de atualização de p_i. As regras de incremento são utilizadas quando há evidências de que o aprendiz aplicou algum conhecimento $\varphi \in MD$ corretamente; as regras de decremento, quando há evidência do contrário. Estas regras intuitivamente deslocam para direita, até um valor ideal $\langle 0, 0, ..., 0, 1 \rangle$, ou para esquerda, até um valor mínimo $\langle 1, 0, ..., 0, 0 \rangle$, as probabilidades em um vetor de crenças. Segundo Beck, Stern e Woolf [BSW97, p.279], "não há nenhum entendimento formal de como este mecanismo funciona, assim ele deve ser validado empiricamente".

Um terceiro mecanismo de manutenção baseada em variáveis aleatórias (independentes) é o utilizado no sistema *Desktop Associate* [Mur97b]. Em linhas gerais esse mecanismo une o melhor dos dois métodos anteriores. Trabalha com vetores de crenças n-dimensionais atualizando-os por meio de equações derivadas a partir da teoria de probabilidades.

Os mecanismos de manutenção com tratamento de incerteza que apresentamos até o momento têm assumido implicitamente que para cada $\varphi \in MD$ a inicialização e atualização das estimativas $I_{MC[\alpha]}(\varphi)$ podem ser feitas *independentemente* de outras estimativas $I_{MC[\alpha]}(\phi)$, para algum outro $\phi \in MD$. Este, no entanto, nem sempre é o caso.

Em um MD, quer dizer, na organização de um corpo de conhecimento a ser aprendido freqüentemente surge a necessidade de estabelecermos relacionamentos entre os elementos $\varphi \in MD$. Um destes é o relacionamento de pré-requisitos. Ou seja, um ou mais φ cujo aprendizado não é possível sem o anterior aprendizado de outros $\varphi_1, ..., \varphi_n \in MD$. Ao considerarmos relacionamentos de pré-requisitos, ou outros relacionamentos entre elementos de MD, a atualização de valores $I_{MC[\alpha]}(\varphi)$ passa a influenciar ou ser influenciada pela modificação de $I_{MC[\alpha]}(\varphi_1), ..., I_{MC[\alpha]}(\varphi_n)$.

No âmbito da teoria de probabilidades, uma maneira de lidar com essa situação é representar e manter o modelo do aprendiz por meio de *redes bayesianas*

Redes bayesianas. De forma breve uma rede bayesiana BN é um grafo dirigido acíclico no qual: (1) nós $n \in N$ correspondem a variáveis aleatórias; (2) arcos $a \in A$ dirigidos de um nó n_a a um nó n_b correspondem à influência direta que n_a tem sobre n_b e (3) cada nó tem associadas distribuições de probabilidades condicionais $d \in D$ que quantificam os efeitos que os nós pais têm sobre o nó; os nós pais de um nó são todos os nós que têm arcos dirigidos para o nó[26]. Em símbolos: $BN = \langle N, A, D \rangle$.

Uma maneira de representar um modelo cognitivo $MC[\alpha] \subset MD$ por meio de redes bayesianas é definir uma rede $\langle MD, A, P \rangle$ e, como antes, fazer:

$$I_{MC[\alpha]} : MD \rightarrow \{\langle p_1, ..., p_n \rangle \mid p_i \in [0,1], \sum_{1 \leq i \leq n} p_i = 1\}$$

Onde $I_{MC[\alpha]}(\varphi).p_i = P[\varphi = na_i]$: ou seja a probabilidade de φ encontrar-se em um nível de assimilação i.

A manutenção baseada em redes bayesianas consiste então em obter evidências a partir do processo de aquisição, atualizar algumas distribuições de probabilidades $I_{MC[\alpha]}(\varphi)$ e, utilizando alguns algoritmos padrões[27] [Nea90, Pea88], propagar estas mudanças ao longo da rede $\langle MD, A, P \rangle$. Um exemplo de sistema que utiliza representação e manutenção baseadas em redes bayesianas é o sistema PITAGORA 2.0 [CMRS95].

No sistema PITAGORA 2.0 um item de conhecimento é definido como um conjunto unitário $i = \{s_i\}$ contendo um procedimento de resolução de problemas na área da Geometria Euclidiana. O conhecimento a ser aprendido por um aluno é então $\bigcup_i \{s_i\}$. Um arco $a \in A$ existe de um nó s_i para um nó s_j se, e somente se, s_i é uma subcadeia (subprocedimento) de s_j. Por fim, define-se uma rede bayesiana $\langle \bigcup_i \{s_i\}, A, P \rangle$ e para cada nó s_i da rede associa-se uma variável aleatória com três valores correspondentes a nível baixo, médio ou alto de assimilação do procedimeto s_i.

As redes bayesianas são atualmente o método mais pesquisado para tratamento numérico de incertezas tanto em modelos do aprendiz como em bases de conhecimentos

[26] Para uma exposição introdutória das Redes Bayesianas ver [Cha91]. Outras referências mais técnicas são [RN95, cap.15], [Ste95, cap.6], [Nea90, Pea88].

[27] O mais conhecido é um algoritmo baseado em passagem de mensagens que dissemina mudanças nas distribuições de probabilidades dos nós $n \in N$ de uma rede $\langle N, A, P \rangle$. Este algoritmo inicia no nó que primeiro teve suas probabilidades mudadas e prossegue propagando estas mudanças pelos seus ancestrais e descendentes na rede, mediado pelas distribuições de probabilidades condicionais $p \in P$ [Ste95, p.499].

em geral. Muitos trabalhos têm sido conduzidas nesta direções. Alguns dos trabalhos mais relevantes são: o sistema EPI-UMOD [Jam96], os sitemas OLAE e POLA [MV95, CV96] e os artigos [Vil92] e [Rey96].

Limitação da Manutenção Numérica

Conforme dissemos de início, as técnicas de manutenção numérica têm como objetivo único lidar com a incerteza inerente ao processo de modelagem do estudante utilizando para isto valores numéricos para quantificar a incerteza presente. Nas técnicas que discutimos, em especial as baseadas na teoria de probabilidades, muitos destes valores numéricos devem ser determinados *a priori*; formam os parâmetros das técnicas.

Uma pergunta crucial é: como obter esses valores? Para esta pergunta não existem respostas diretas. Os pesquisadores quando muito dão algumas sugestões. No mais fica a cargo de quem se dispõe a utilizar as técnicas um bom trabalho de avaliar e estabelecer através de estudos empíricos valores adequados para os parâmetros.

3.5.3 Manutenção Simbólica

A manutenção, durante o processo de modelagem do aprendiz, sob um ponto de vista amplo, tem como objetivo gerenciar as mudanças feitas em um modelo do aprendiz. Estas mudanças segundo Paiva, Self e Hartley [PSH94] podem ser de dois tipos[28] :

- *Mudanças do Sistema*: são mudanças que devem ser feitas para integrar informações (incertas) provindas da aquisição às informações já presentes no modelo do aprendiz;
- *Mudanças do Aprendiz*: são mudanças que devem ser feitas e explicitamente representadas para seguir a dinâmica de aprendizagem de um aluno.

As técnicas de manutenção numérica revisadas na seção anterior lidam apenas com o primeiro tipo de mudanças. As técnicas de manutenção simbólica, que agora passamos a discutir, tentam gerenciar esses dois tipos de mudanças de forma integrada.

Na manutenção numérica, seção anterior, a representação do modelo do aprendiz considerada foi a representação por sobreposição com incerteza $I_{MC[\alpha]}$; ver seção 3.3.1. Na manutenção simbólica a representação do modelo do aprendiz considerada é a representação por teorias lógicas $Th[\alpha]$; ver seção 3.3.1. Em símbolos:

$$Th[\alpha] = \{\varphi \in \mathcal{L} \mid D \vdash_{\alpha} \varphi\}$$

[28] Também chamadas, respectivamente, de *mudanças não-monotônicas* e *mudanças temporais* [Pai95].

onde $\mathcal{L}$ denota uma linguagem lógica (e.g. lógica predicados, lógicas modais não monotônicas, lógica temporal) e $D \models_\alpha \varphi$ representa o fato de que o aprendiz α crê que φ seja uma sentença verdadeira (conhecimento) em um domínio de aprendizagem D.

Enquanto a manutenção numérica têm como objetivo lidar com a incerteza inerente ao processo de modelagem do aprendiz as técnicas simbólicas de manutenção lidam com *inconsistências* no modelo do aprendiz; inconsistências que surgem tanto pela incerteza inerente ao processo de modalagem do aprendiz (mudanças do sistema) ou pelo simples fato de o aluno aprender e desaprender (mudanças do aprendiz).

Sistemas de Manutenção de Verdade

Uma das técnicas utilizadas em sistemas de modelagem do aprendiz para gerenciar as mudanças e manter a consistência de um modelo $Th[\alpha]$ são os chamados *sistemas de manutenção de verdade*[29]. Alguns sistemas de manutenção de verdade já desenvolvidos para modelos do aprendiz são reportados nos seguintes trabalhos de pesquisa: [HMGN91, KIM92, GT94, PS94]; *apud* [Pai95].

Um sistema de manutenção de verdade para modelos do aprendiz $Th[\alpha]$ consiste em mecanismos para revisar fatos φ no momento em que surgem contradições (inconsistências) no modelo: $Th[\alpha] = \{..., \varphi, \neg\varphi, ...\}$. Para tanto o sistema de manutenção de verdade mantém um histórico de todas as justificativas para todos os fatos incluídos no modelo do aprendiz. Além disso o sistema de manutenção deve ser capaz de decidir como revisar fatos contraditórios; deve decidir que fatos manter e que fatos eliminar.

Um exemplo é o sistema de manutenção de verdade implementado no sistema TAGUS [PS95]. Neste sistema as justificativas para a inclusão de novos fatos φ no modelo do aprendiz $Th[\alpha]$ são mantidas por um subsistema chamado AMMS [PS94]. O AMMS utiliza uma rede de justificativas para manter as dependências entre fatos no modelo. A rede de justificativas é definida como um par $\langle N, J \rangle$, onde N é o conjunto de nós da rede e J é o conjunto de justificativas. Há quatro tipos de nós na rede: suposições, suposições justificadas, "endosso" e premissas. Cada fato $\varphi \in Th[\alpha]$ tem associado um nó $n_\varphi \in N$ na rede.

Quando o processo de aquisição deriva um novo fato φ sobre o aprendiz, o AMMS irá criar uma nova justificativa que será incluída no conjunto J da rede de justificativas. Uma justificativa é

$$n_{\varphi_1}, n_{\varphi_2}, ..., n_{\varphi_k} \rightarrow n_\varphi$$

[29] Do inglês: *Truth Maintenance Systems*.

onde $n_{\varphi_1}, n_{\varphi_2}, ..., n_{\varphi_k}$ são os nós antecedentes e n_φ é o conseqüente.

Quando uma contradição é encontrada no modelo do aprendiz, o AMMS, por meio da rede de justificativas, produz um conjunto de *ambientes* (nós causas) que suportam a contradição. A partir destes ambientes o sistema de manutenção irá decidir se a contradição é uma inconsitência do sistema ou se é uma insconsistência do aprendiz. Por exemplo, se tanto φ quanto $\neg\varphi$ estão em $Th[\alpha]$, duas coisas podem ter acontecido: ou o aprendiz tem realmente crenças contraditórias ou o processo de aquisição, utilizando regras de aquisição limitadas (suposições), inferiu um fato que não é verdadeiro. Então, como o sistema de manutenção pode decidir que fatos reter em $Th[\alpha]$?

Esta decisão é feita pelo uso de uma *função de confiança*[30] sobre os ambientes que suportam a contradição. A confiança do sistema de manutenção em um fato no modelo do aprendiz depende de como o fato foi inferido. Altos valores de confiança indicam que o aprendiz tem crenças contraditórias. Baixos valores de confiança indicam que o processo de aquisição inferiu fatos contraditórios.

Se o sistema de modelagem do aprendiz inferiu fatos contraditórios sobre o aprendiz, é dever do AMMS modificar a rede de justificativas, retirando nós relacionados a fatos poucos confiáveis, até tornar o modelo do aprendiz novamente consistente. Se o aprendiz tem crenças contraditórias, suas causas (ambientes) são passadas a um módulo de planejamento pedagógico que irá criar uma intervenção de acordo com a contradição. Neste ínterim a contradição é mantida no modelo do aprendiz. Posteriormente, ações do aprendiz que indicam mudanças no estado de crenças do aprendiz irão iniciar o processo de eliminação da inconsistência do modelo do aprendiz.

Modelos do Aprendiz Temporais

Uma outra técnica atualmente sendo pesquisada para gerenciar as mudanças e manter a consistência em modelos do aprendiz é a representação explícita de conhecimentos temporais [GT96]. A idéia básica da proposta é lidar com a incerteza e a dinâmica inerentes ao processo de modelagem do aprendiz representando o estado de conhecimento dos alunos por meio de crenças referentes a conceitos aprendidos por eles e a situações temporais nas quais eles mostram saber ou ignorar os conceitos. Estas crenças são chamadas de fórmulas temporais. Em símbolos:

- $B_\alpha(t, \varphi)$: no tempo t o aprendiz α crê no conceito φ, onde φ refere-se a um conceito correto ou incorreto de um domínio de aprendizagem D.

- $B_\alpha(t_1, t_2, \varphi)$: no intervalo de tempo $[t_1, t_2]$ o aprendiz α crê no conceito φ.

[30] Tradução livre de *trust function*.

Um modelo do aprendiz temporal $TTh[\alpha]$ consiste então em uma coleção de fórmulas temporais, em conjunto com *restrições temporais*. Estas últimas tendo a forma de inequações envolvendos variáveis de fórmulas temporais: $a \leq t \leq b$; $a_1 \leq t_1 \leq b_1$; $a_2 \leq t_2 \leq b_2$.

Tendo em mãos um modelo do aprendiz temporal $TTh[\alpha]$ o processo de manutenção consiste em criar e incluir novas fórmulas e restrições temporais em $TTh[\alpha]$ toda vez que novas informações sobre um aprendiz α são adquiridas e, lidar com possíveis inconsistências por meio de distinções feitas a partir das restrições temporais associadas a cada fórmula em $TTh[\alpha]$ [GT97].

Capítulo 4

Direções Gerais para a Modelagem do Aprendiz nos Agentes MATHEMA

"(...), como um modelo conceitual, o MATHEMA propõe-se principalmente a prover princípios e uma arquitetura alternativa necessários para orientar o desenvolvimento de sistemas IA-ED particulares."

[Cos97, cap. 4]

4.1 Introdução

O objetivo desta dissertação é especificar a modelagem do aprendiz para a arquitetura MATHEMA. No capítulo introdutório dividimos este objetivo geral em dois outros mais específicos: (1) identificar de forma geral quais as possibilidades e recursos computacionais disponíveis para a realização da modelagem do aprendiz pelos agentes tutores MATHEMA; (2) definir um sistema de modelagem do aprendiz particular que possa ser utilizado em ambientes de aprendizagem construídos segundo a arquitetura MATHEMA.

Neste capítulo iremos apresentar os resultados que alcançamos ao perseguirmos o objetivo (1). A apresentação será baseada nos elementos da arquitetura MATHEMA mostrados no Capítulo 2 e no estudo geral sobre modelagem do estudante feito no Capítulo 3.

O capítulo está organizado em três seções principais. Na seção 4.2 mostramos mais alguns detalhes dos agentes tutores MATHEMA. Falamos sobre a funcionalidade geral e estrutura interna do Sistema Tutor dos agentes MATHEMA. Na seção 4.3, levando

em consideração as características do Sistema Tutor, identificamos quem modelar, que informações são de interesse modelar e para que os agentes tutores MATHEMA podem utilizar as informações de um modelo do aprendiz. Na seção 4.4 mostramos que recursos computacionais podem ser utilizados no projeto e implementação de Sistemas de Modelagem do Aprendiz para o Sistema Tutor de um agente MATHEMA.

4.2 O Sistema Tutor

O MATHEMA, conforme apresentamos no Capítulo 2, é uma arquitetura multi-agente para a concepção de ambientes de aprendizagem por computador. Nesta arquitetura cada agente, dito agente tutor MATHEMA, é responsável por uma visão específica (contexto, profundidade) do conhecimento a ser ensinado aos aprendizes usuários do sistema (seção 2.5.1). Um agente tutor MATHEMA é composto de três subsistemas: o *Sistema Tutor*, o Sistema Social e o Sistema de Distribuição (seção 2.5.2).

4.2.1 A Interação com o Aprendiz

O Sistema Tutor é o componente de um agente MATHEMA responsável por interagir diretamente com o aprendiz. Seu papel é fazer com que o aprendiz adquira conhecimentos relativo à parte do domínio de aprendizagem (um par contexto, profundidade organizado através de uma estrutura pedagógica) tomando por base um modelo de ensino e aprendizagem *"calcado em resolução de problemas"* [Cos97, p.104].

O Modelo de Ensino-Aprendizagem

O modelo de ensino e aprendizagem adotado pelo Sistema Tutor contempla a aprendizagem por resolução de problemas a partir de duas modalidades. Na primeira modalidade, o Sistema Tutor escolhe, *mediante certos critérios*, um problema e o apresenta ao aprendiz. O aprendiz tenta então resolvê-lo; caso não consiga ou tenha dificuldades o Sistema Tutor pode intervir (sendo solicitado ou por conta própria) e dar ajuda ao aprendiz. A ajuda pode ser uma dica, uma explicação ou mesmo a solução de parte do problema. Neste caso o aprendiz é dito *aprender fazendo* dispondo de orientação quando necessário. Esta modalidade enfatiza as características exploratórias do aprendiz em situações de resolução de problemas.

Na segunda modalidade o aprendiz coloca um problema para o Sistema Tutor. O Sistema Tutor resolve o problema apresentando ao longo do processo de resolução uma série de explicações. Neste caso o aprendiz é dito *aprender por instrução direta*. Esta modalidade enfatiza o papel do *ensino* no processo de aprendizagem.

O Modelo de Interação

A partir do modelo de ensino-aprendizagem adotado pelo Sistema Tutor, grande parte da interação entre aprendiz e Sistema Tutor, ou antes, o *conteúdo* da interação pode ser resumido no esquema apresentado na Figura 4.1. Nesse esquema a interação entre Sistema Tutor ocorre através de de troca de mensagens: o Sistema Tutor envia uma mensagem de conteúdo $[T]$ e recebe do aprendiz uma mensagem de conteúdo $[A]$.

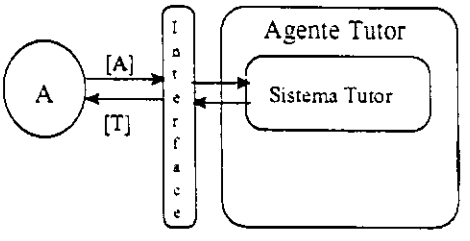


Figura 4.1: A interação Sistema Tutor - Aprendiz

As mensagens $[T]$ podem ser:

- Um problema;
- Uma ajuda : dicas, explicações, etc;
- A resolução de um problema.

As mensagens $[A]$ podem ser:

- A solução de um problema;
- Um pedido de ajuda;
- Um problema.

4.2.2 A Necessidade de Modelar o Aprendiz

No esquema de interação apresentado acima, o Sistema Tutor deve ser capaz de realizar um conjunto de tarefas relacionadas ao envio de mensagens $[T]$ e a recepção de mensagens $[A]$. Estas tarefas chamaremos de *funções pedagógicas* do Sistema Tutor. Algumas funções pedagógicas são:

- Escolher problemas *adequados* a serem passados ao aprendiz;

- Compreender o processo de resolução de problemas do aluno de tal modo a poder ajudá-lo quando for necessário;
- Avaliar a solução de um problema dada pelo aprendiz;
- Resolver problemas para o aprendiz.

A realização de algumas funções pedagógicas importantes irá requerer necessariamente informações sobre o aprendiz. Tomemos como exemplo a primeira função pedagógica da lista acima. Anteriormente falamos que a escolha de problemas é feita *mediante certos critérios*; acima dissemos escolher problemas *adequados*. Certos critérios, problemas adequados são termos vagos. Que critérios podem ser empregados para escolher um problema adequado para um aluno particular? O mais natural, acreditamos, é fazer uma estimativa do *nível de conhecimento* do aluno e assim apresentar problemas de acordo com este nível. Neste caso o critério de adequação é o conhecimento do aluno. Uma outra abordagem poderia ser indagar diretamente o aprendiz sobre que tipos de problemas prefere e assim apresentar problemas deste tipo. Neste caso o critério de adequação são as preferências do aprendiz. Como vemos, em ambos os casos são necessárias informações sobre o aprendiz.

Se definirmos, como fizemos no Capítulo 3, *modelo do aprendiz* como uma representação computacional explícita de informações sobre os aprendizes que um ambiente de aprendizagem por computador dispõe para realizar atividades pedagógicas de forma efetiva então o Sistema Tutor dos agentes MATHEMA necessita de um modelo do aprendiz. Esta necessidade foi explicitamente indicada na definição original dos agentes MATHEMA [Cos97]. No entanto, não foi trabalhada em detalhes. Não foi especificado que tipos de aprendizes podem ser modelados, que informações são importantes modelar, que funções pedagógicas devem se beneficiar utilizando essas informações e como essas informações podem ser representadas, adquiridas e mantidas pelo Sistema Tutor. Isso, como já falamos, é o objetivo de nosso trabalho.

As direções gerais para a modelagem do aprendiz na arquitetura MATHEMA que apresentamos nas próximas seções deste capítulo são o nosso ponto de partida. Antes de iniciarmos, temos ainda que mostrar como o Sistema Tutor é estruturado internamente de modo a realizar suas funções pedagógicas. Necessitamos conhecer esta estrutura pois sobre ela montaremos o *Sistema de Modelagem do Aprendiz* dos agentes tutores MATHEMA.

4.2.3 Componentes do Sistema Tutor

O Sistema Tutor, de forma geral, é definido como sendo formado por três grandes componentes [Cos97, p.106]:

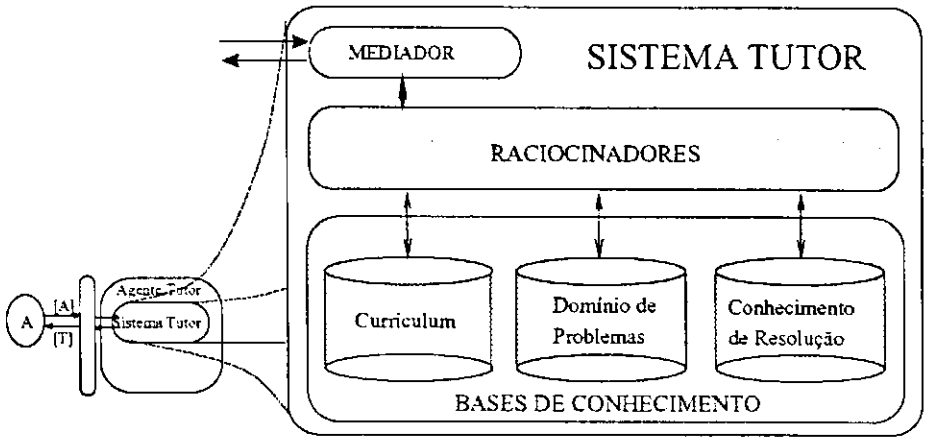


Figura 4.2: Componentes do Sistema Tutor

- o *Mediador* : módulo encarregado do controle geral de execução do Sistema Tutor. Ele é responsável por identificar que tipo de interação ocorre em determinado momento entre o aprendiz e o Sistema Tutor e assim passar as mensagens e o controle de execução para os mecanismos corretos presentes nos Raciocinadores;
- os *Raciocinadores* : um grande módulo formado por um conjunto de mecanismos encarregados de realizar as funções pedagógicas a cargo do Sistema Tutor;
- as *Bases de Conhecimentos* : reunem conhecimentos e informações necessárias à operacionalização das atividades desempenhadas pelos mecanismos presentes nos raciocinadores.

As Bases de Conhecimento

Há no mínimo três bases de conhecimento. O *Curriculum*, o *Domínio de Problemas* e o *Conhecimento de Resolução de Problemas*. Estas bases têm como conteúdo as informações, problemas e conhecimentos identificados durante a fase de estruturação e organização do conhecimento de aprendizagem usando a abordagem multi-dimensional apresentada na seção 2.4. A base *Curriculum* tem como conteúdo as informações sobre unidades pedagógicas e relação de ordem entre as unidades pedagógicas. A base *Domínio de Problemas* armazena os problemas a serem passados aos alunos. A base *Conhecimento de Resolução de Problemas* contém o conhecimento que habilita o sistema tutor a resolver problemas para o aluno.

Em meio a estas bases estabelecemos a seguir a existência de uma quarta base de conhecimento: o *Modelo do Aprendiz.*

4.3 Quatro Questões Principais

A tarefa de especificar a modelagem do aprendiz para ambientes de aprendizagem por computador envolve um amplo espaço de possibilidades e recursos computacionais. No estudo que fizemos no Capítulo 3, agrupamos este espaço em torno de quatro questões principais [Fin89]: Quem modelar? O que modelar? Para que modelar? e Como modelar? No restante deste capítulo nos ocuparemos em responder estas quatro questões para o Modelo do Aprendiz a ser incorporado no Sistema Tutor dos agentes MATHEMA. Essas repostas chamamos de *direções gerais para a modelagem do aprendiz nos agentes MATHEMA.*

Antes de iniciarmos é importante deixar claro o que as direções gerais pretendem ser. A aquitetura MATHEMA, como dissemos no Capítulo 2, é um modelo conceitual para o desenvolvimento de ambientes de aprendizagem por computador. Como um modelo conceitual, tem como proposta principal "prover princípios e uma arquitetura alternativa necessários para orientar o desenvolvimento de sistemas IA-ED [ambientes de aprendizagem por computador] particulares"[Cos97, p.47]. Neste contexto as direções gerais que apresentamos têm como principal objetivo servir como um conjunto de princípios e orientações que irão guiar o projeto e implementação de Sistemas de Modelagem do Aprendiz em ambientes de aprendizagem por computador construídos segundo a arquitetura MATHEMA.

4.3.1 Quem modelar?

Os ambientes de aprendizagem construídos a partir da arquitetura MATHEMA são projetados tendo em vista uma entidade chamada aprendiz humano; ver Figura 2.2. Ou seja, um indivíduo, pessoa interessada em aprender e que para tanto irá interagir com um agente tutor a partir de atividades de resolução de problemas.

Desta forma, utilizando-nos da terminologia empregada na seção 3.2.1, o modelo do aprendiz para os agentes MATHEMA deve ser *individual* e *persistente.*

4.3.2 O que modelar?

Das categorias de informação que apresentamos na seção 3.2.2, pelo menos três tipos de informações podem ser explorados na elaboração de um Modelo do Aprendiz para

os agentes MATHEMA. São eles:

1. O conhecimento de cada aprendiz em relação ao domínio de aprendizagem. No MATHEMA o domínio de aprendizagem é organizado em torno de uma estrutura pedagógica (seção 2.4);

2. Informações a respeito dos planos e estratégias utilizados por um aprendiz ao resolver problemas;

3. Informações gerais sobre as preferências de cada aprendiz, tais como, prefere resolver problemas ou ver problemas sendo resolvidos; prefere problemas envolvendo situações reais ou problemas abstratos; prefere dicas a qualquer ora ou apenas quando solicita ao sistema;

O interesse por tais informações é discutido a seguir.

4.3.3 Para que modelar?

Conforme já dissemos na seção 4.2.2, um Modelo do Aprendiz para os agentes MATHEMA terá como principal uso o fornecimento de informações sobre os aprendizes para a realização de algumas funções pedagógicas importantes. Abaixo listamos as funções pedagógicas que podem se beneficiar de cada um dos tipos de informação identificados na seção anterior.

1. Conhecimento do aprendiz:

   - Escolha de problemas adequados ao nível de conhecimento de cada aprendiz;
   - Fornecer dicas, ajudas, explicações de acordo com o grau de conhecimento de cada aprendiz;
   - Avaliar a solução de um problema dada pelo aprendiz;

2. Planos e estratégias:

   - Compreender o processo de resolução de problemas do aluno de tal modo a poder ajudá-lo quando for necessário;
   - Analisar a solução de um problema dada pelo aprendiz;

3. Preferências:

   - Tomar decisões estratégicas quanto a forma de interação com o aprendiz;
   - Escolher problemas adequados as preferêcias gerais de cada aprendiz.

Alguns comentários:

A lista acima não é exaustiva. No entando, cobre as funções pedagógicas básicas do Sistema Tutor relativas ao modelo de interação apresentado na seção 4.2.1.

As informações sobre o estado de conhecimento do aprendiz, usando a terminologia empregada na seção 3.2.3, devem ser utilizadas com fins elaborativos. Com isto queremos dizer que os agentes tutores MATHEMA não devem utilizar estas informações para impor pura e simplesmente seu conhecimento "correto" ao aluno e sim ajudar o aprendiz a refletir, ampliar e melhorar a qualidade de seu conhecimento. Como observamos na seção 3.2.3 esse é atualmente o uso indicado para o modelo do aprendiz em Ambientes de Aprendizagem Interativos.

As informações sobre o conhecimento do aprendiz podem ainda ser utilizados com fins avaliativos. Como exemplo podemos imaginar o Sistema Tutor utilizando estas informações para gerar um escore relativo ao desempenho do aprendiz após uma sessão de resolução de problemas.

O uso de informações sobre os objetivos e planos do aprendiz para o fornecimento de dicas e explicações foi ilustrado na seção 3.2.3.

Alguns exemplos de tomadas de decisões estratégicas apoiadas em informações sobre as preferências dos aprendizes podem ser: decidir operar com mais freqüência resolvendo problemas caso aluno prefira ver uma série de problemas resolvidos antes de tentar resolver alguns problemas; decidir não aborrecer o aprendiz interrompendo-o para fornecer uma ajuda não solicitada caso ele já tenha dito explicitamente que só quer ajuda quando achar necessário e pedir.

Em suma, pelo menos três tipos de informações mostram-se úteis e *justificam* a construção de modelos de aprendiz para os agentes tutores MATHEMA. A *viabilidade*, ou seja, que técnicas computacionais podem ser empregadas para representar, adquirir e manter essas informações durante a interação do Sistema Tutor com o aprendiz é o que veremos a seguir.

4.3.4 Como modelar?

Como definimos na seção 3.2.4 a modelagem do aprendiz é o processo de aquisição e manutenção de informações em um modelo do aprendiz. Quando realizada durante a

interação entre o ambiente de aprendizagem e o aprendiz – o esperado nos agentes tutores MATHEMA – a modelagem do aprendiz fica a cargo de um Sistema de Modelagem do Aprendiz.

Na arquitetura MATHEMA, estas idéias nos levam a incorporar na definição original do Sistema Tutor (Figura 4.2) um Sistema de Modelagem do Aprendiz. Na Figura 4.3 mostramos a estrutura do Sistema tutor contendo um Sistema de Modelagem do Aprendiz:

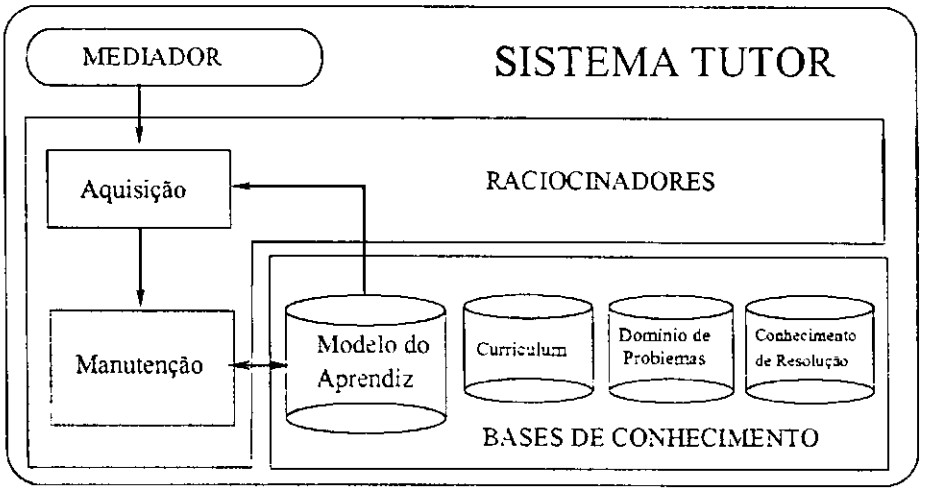


Figura 4.3: O Sistema Modelagem Aprendiz dos agentes MATHEMA

A funcionalidade geral de um Sistema de Modelagem do Aprendiz está comentada na seção 3.2.4. Há dois processos principais: a aquisição e a manutenção. Na arquitetura geral do Sistema Tutor estes processos devem ser realizados por mecanismos que se enquadram no grande módulo Raciocionadores. O modelo do aprendiz, como uma base de informações sobre os aprendizes, enquadra-se no grande módulo Bases de Conhecimento.

Na próxima seção detalhamos que técnicas computacionais podem ser utilizadas para realizar este Sistema de Modelagem do Aprendiz nos agentes tutores MATHEMA.

4.4 O Sistema de Modelagem do Aprendiz

No Capítulo 3, seções 3.3, 3.4 e 3.5, revisamos uma série de idéias e técnicas computacionais para projeto e implementação de sistemas de modelagem do aprendiz. A Figura 4.4 mostra um resumo esquemático das idéias e técnicas apresentadas. No restante dessa seção e capítulo discutiremos como algumas destas idéias e técnicas computacionais podem ser utilizadas no projeto e implementação de Sistemas de Modelagem do Aprendiz nos agentes MATHEMA.

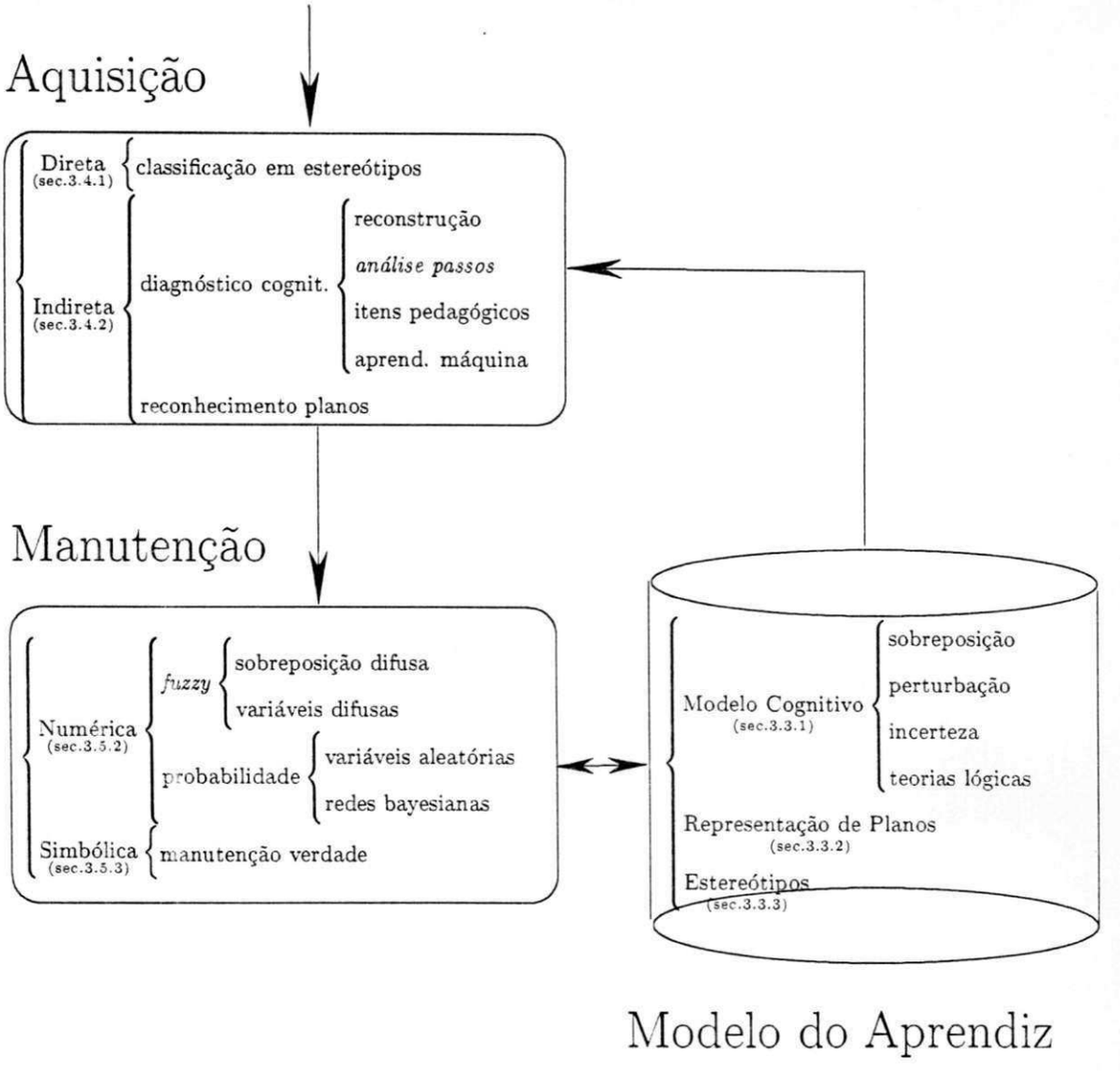


Figura 4.4: Resumo das idéias e técnicas computacionais para projeto e implementação de sistemas de modelagem do aprendiz.

4.4.1 O modelo do aprendiz

O Modelo do Aprendiz para os agentes MATHEMA foi caracterizado acima como uma base de conhecimento individual e persistente onde possivelmente estarão representados três itens: o conhecimentos de cada aprendiz em relação ao domínio de aprendizagem, seus planos e estratégias, e suas preferências gerais.

O conhecimento do aprendiz. Na Figura 4.4 identificamos três métodos de representar o conhecimento de um aprendiz em relação ao domínio de aprendizagem. São eles: a representação por sobreposição , a representação de perturbações e a repre-

sentação por sobreposição com incerteza. Estes métodos foram apresentadas na seção 3.3.1. Qualquer um dos três métodos pode ser utilizado nos agentes MATHEMA.

O método de representação por sobreposição pressupõe a existência de um modelo do domínio. No Sistema Tutor dos agentes MATHEMA, apresentado na seção 4.2, há duas bases de conhecimento que podem fazer o papel de modelo do domínio. São elas: o *Curriculum* e o Conhecimento de Resolução de Problemas. A representação por sobreposição pode então ser feita tomando como referência qualquer uma dessas duas bases.

O método de representação de perturbações, além da existência de um modelo de domínio, pressupõe uma forma de representar mal-entendidos. A técnica comumente utilizada para representar mal-entendidos é a extensão do modelo de domínio por meio de um catálogo de mal-entendidos. No Sistema Tutor dos agentes MATHEMA isto equivale a dizer que além das bases que já falamos deve haver uma terceira base contendo os erros mais comuns cometidos por aprendizes ao resolverem problemas relativos a um dado domínio de aprendizagem.

O método de representação por sobreposição com incerteza é uma variação do método de sobreposição. A utilização deste método no modelo do aprendiz dos agentes MATHEMA requer que façamos uma representação do conhecimento do aprendiz por sobreposição e escolhamos uma linguagem de representação de incerteza. A escolha da linguagem de representação de incerteza e sua relação com os elementos do modelo do aprendiz está intimamente ligada ao processo de manutenção (numérica). Ver a seguir.

Dos três métodos o mais recomendado é a utilização de sobreposição com incerteza. Como extensamente discutimos ao longo da seção 3.5 a incerteza é um fator inerente a todo processo de modelagem do aprendiz. E como vimos na mesma seção, existem técnicas relativamente simples para representação e manutenção de sobreposição com incerteza e que oferecem melhores resultados que a representação por sobreposição simples.

Além dos métodos de representação tendo por base um modelo de domínio, identificamos na Figura 4.4 uma quarta forma de representar o estado de conhecimento dos aprendizes: a construção de teorias lógicas. Esta forma de representação discutimos rapidamente ao final da seção 3.3.1. Nos agentes MATHEMA este método de representação não se aplica de forma natural. Os agentes MATHEMA possuem um corpo de conhecimento pré-definido que no decorrer de atividades de resolução de problemas

tentam transmitir ao aprendiz.

Planos e estratégias. Conforme discutimos na seção 3.3.2, não existem métodos padrões para a representação de planos e estratégias de um aprendiz ao resolver problemas. A representação de planos irá depender da técnica de reconhecimento de planos a ser utilizada. Ver a seguir.

Preferências. A representação de informações sobre preferências gerais dos aprendizes pode ser realizada nos agentes MATHEMA por meio de estereótipos. Os estereótipos foram discutidos na seção 3.3.3.

4.4.2 O Módulo de Aquisição

A aquisição de informações sobre o conhecimento do aprendiz, seus planos ao resolver problemas e preferências gerais pode ser feita de forma direta ou indireta. Ver Figura 4.4.

Aquisição Direta

A aquisição direta pode ser empregada para obter o conhecimento inicial dos aprendizes e/ou obter informações sobre as preferências gerais dos aprendizes. Ver seção 3.4.1.

Aquisição Indireta

A aquisição indireta pode ser empregada para inferir o estado de conhecimento dos aprendizes – diagnóstico cognitivo – e/ou seus planos ao resolverem problemas – reconhecimento de planos. Ver seção 3.4.2.

O conhecimento do aprendiz. Na Figura 4.4 identificamos quatro estratégias de diagnóstico cognitivo. São elas: reconstrução, análise de passos, reconhecimento de itens pedagógicos e aprendizagem de máquina.

As três primeiras estratégias podem ser utilizadas nos agentes MATHEMA em conjunto com os métodos de representação por sobreposição. Ver esquema geral apresentado na seção 3.4.2, página 46. Tanto a estratégia de reconstrução quanto a estratégia de análise de passos podem ser aplicadas para inferir o conhecimento utilizado pelo aprendiz ao resolver problemas. Neste caso a base Conhecimento de Resolução de Problemas presente no Sistema Tutor dos agentes MATHEMA pode fazer o papel de modelo de resolução de problemas ideal. Já a estratégia de reconhecimento de itens pedagógicos

pode ser empregada para adquirir uma descrição do conhecimento do aprendiz ao resolver problemas. Neste caso a base *Curriculum* pode fazer o papel de *curriculum* de referência.

As estratégias de diagnóstico cognitivo baseadas em aprendizagem de máquina podem também ser utilizadas. No entando, fazemos uma observação. As técnicas de aprendizagem de máquina são em geral utilizadas para a construção de teorias lógicas; quer dizer, inferir conhecimentos não representados previamente no sistema a partir do comportamento observado do aprendiz. Como dissemos anteriormente, os agentes MATHEMA já possuem um modelo de domínio que resume o conhecimento que um aluno deve adquirir interagindo com o tutor. Assim o uso de estratégias de diagnóstico cognitivo baseadas em aprendizagem de máquina não parece ser muito natural. Um possível uso, no entanto, poderia ser identificar mal-entendidos dos aprendizes não previstos em um catálogo de mal-entendidos.

Conforme dissemos ao final da seção 3.4.2 as quatro abordagens para diagnóstico cognitivo identificadas não são incompatíveis e podem, e até devem, ser combinadas.

Planos e estratégias. Conforme discutimos na seção 3.4.2, o método padrão para reconhecimento de planos é a criação de bibliotecas de planos. Das técnicas que comentamos naquela seção, pensamos que as idéias do sistema FITS-2 e as idéias de reconhecimento de planos baseada em granularidade possam servir de ponto de partida para a aquisição de informações sobre o planos nos agentes MATHEMA. Isto porque tais técnicas foram concebidas para serem utilizadas em ambientes de aprendizagem gerais.

No sistema FITS-2 os planos são representados através de hierarquias de planos. Ver seção 3.3.2, Figura 3.8. No reconhecimento de planos baseado em granularidade os planos são representados como grafos IGH. Ver seção 3.4.2, página 49.

4.4.3 O Módulo de Manutenção

A manutenção de informações em um modelo do aprendiz pode ser realizada seguindo um dentre dois caminhos: manutenção numérica ou manutenção simbólica. Ver Figura 4.4.

Manutenção Numérica

Na manutenção numérica a preocupação principal é integrar novas informações, providas da aquisição, às informações já representadas no modelo do aprendiz levando em

conta a incerteza inerente ao processo de modelagem. Considera-se ainda que o modelo do aprendiz é representado por sobreposição com incerteza.

As principais técnicas empregadas na manutenção numérica podem ser divididas em técnicas baseadas na teoria de conjuntos difusos e técnicas baseadas na teoria de probabilidades.

Conforme dissemos anteriormente, a representação por sobreposição com incerteza é o método mais recomendado para o Modelo do Aprendiz dos agentes MATHEMA. Desta forma, qualquer uma das técnicas para manutenção numérica mostradas na Figura 4.4 pode ser tentada para implementar o processo de manutenção nos Sistema de Modelagem do Aprendiz dos agentes MATHEMA. Em especial as técnicas probabilísticas que atualmente são as mais pesquisadas embora nem sempre fáceis de implementar por envolverem um grande número de parâmetros.

Manutenção Simbólica

Na manutenção simbólica a preocupação principal também é integrar novas informações, provindas da aquisição, às informações já representadas no modelo do aprendiz. Só que agora levando em consideração o problema de possíveis inconsistências que podem surgir em um modelo do aprendiz representado como uma teoria lógica.

A principal técnica empregada na manutenção simbólica são os sistemas de manutenção de crenças.

Conforme dissemos anteriormente, a representação do Modelo do Aprendiz dos agentes MATHEMA não é uma opção natural. Desta forma, o uso de sistemas de manutenção de crenças para implementar o processo de manutenção nos Sistema de Modelagem do Aprendiz dos agentes MATHEMA também não são uma opção natural.

Capítulo 5

Um Sistema de Modelagem do Aprendiz para os Agentes MATHEMA

5.1 Introdução

No capítulo anterior apresentamos direções gerais para a modelagem do aprendiz na arquitetura MATHEMA. No presente capítulo, seguindo algumas das direções apontadas, apresentamos uma proposta de Sistema de Modelagem do Aprendiz (SMA) a ser utilizado nos agentes tutores de ambientes de aprendizagem construídos a partir da arquitetura MATHEMA.

5.1.1 Especificação Formal

O SMA que propomos está definido por meio de uma linguagem de especificação formal – a linguagem VDM-SL [LHB+96, ABH+95]. O principal motivo que nos levou a utilizar uma linguagem de especificação formal para definir um SMA para os agentes MATHEMA[1] é a possibilidade de descrever com precisão e rigor matemático o comportamento esperado do SMA. Isto nos ajuda a ter uma compreensão mais clara do funcionamento do sistema antes de ser implementado.

Em particular, a escolha da linguagem VDM-SL[2] – uma linguagem *orientada a*

[1] Em oposição a uma abordagem informal utilizando linguagem natural e diagramas.

[2] Outros métodos formais de especificação são [FKV94]: **Z**, uma linguagem *orientada a modelos* para especificação formal de *sistemas seqüenciais*; **Larch**, uma linguagem *algébrica* para especificação formal de *sistemas seqüenciais*; e **Redes de Petri**, um método *orientado a modelos* para especificação formal de *sistemas concorrentes* e *paralelos*.

modelos para especificação formal de *sistemas seqüenciais* – foi influenciada principalmente por motivos pragmáticos:

(a) A adequação da linguagem para descrever o SMA proposto que é, em essência, um sistema seqüencial;

(b) A disponibilidade de ferramentas automatizadas para verificação e validação da especificação formal. Ferramentas tais como a IFAD VDM-SL Toolbox[3] desenvolvida pelo Instituto de Ciência da Computação Aplicada (IFAD) [Gro98];

(c) O conhecimento da linguagem VDM-SL pelo autor.

Especificações VDM-SL

Uma especificação VDM-SL consiste em um modelo matemático constituído de *objetos*, que representam entradas, saídas e o estado do sistema, e *operações* e *funções* que representam manipulações feitas nestes objetos. Classes de objetos, chamadas de *tipos de dados*, são definidas a partir de objetos matemáticos (tipos de dados) tais como *conjuntos, objetos compostos, listas e mapeamentos.* As operações e funções podem ser definidas de forma *direta* – através de *comandos* e/ou *expressões* que definem uma relação entre argumentos e resultado da operação ou função – ou especificadas de forma *implícita* – através de pré e pós-condições. A diferença básica entre operações e funções está no fato de que apenas operações podem modificar os objetos que representam o estado interno do sistema. No Apêndice A apresentamos um breve resumo da terminologia, notação e sintaxe VDM-SL para tipos de dados, funções e operações que utilizamos na especificação do SMA apresentada nas próximas seções deste capítulo.

Desenvolvimento Sistemático

A especificação VDM-SL, além de possibilitar um entendimento mais claro do SMA antes de sua implementação, pode ser utilizada como ponto de partida em um método sistemático de desenvolvimento de *software*: o *"Vienna Development Method"* - VDM[4] [Jon90].

VDM é uma coleção de técnicas para a especificação formal e desenvolvimento de sistemas computacionais. Consiste na linguagem de especificação formal VDM-SL em conjunto com regras estritas para refinamento de tipos de dados e operações.

[3] Esta ferramente encontra-se disponível no endereço `http://www.ifad.dk/Products/products.htm` e foi utilizada pelo autor na elaboração da especificação formal do SMA apresentada neste capítulo.

[4] A origem do VDM remonta a pesquisas sobre semânticas formais de linguagens de programação realizadas nas décadas de 1960 e 70 no laboratório da empresa IBM em Viena.

Estas regras permitem estabelecer ligações entre especificações abstratas de requisitos e detalhes incorporados nas etapas de projeto e implementação. Há ainda uma *teoria de prova* na qual argumentos rigorosos (ou plenamente formais) podem ser feitos a respeito das propriedades do sistema sendo especificado ou então sobre a *correção* (satisfação em relação a uma especificação inicial) de decisões de projeto. Na Figura 5.1 ilustramos o desenvolvimento sistemático de *software* usando VDM.

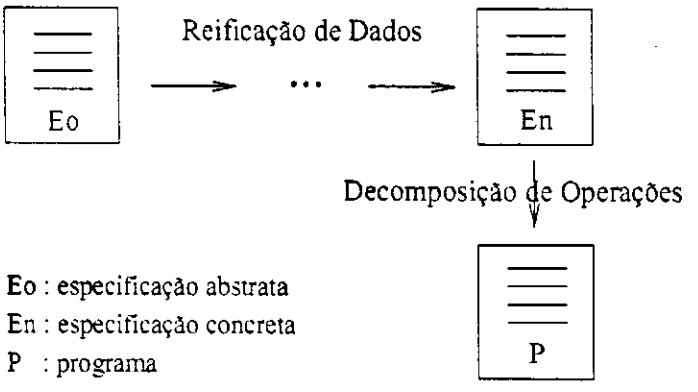


Figura 5.1: Desenvolvimento Sistemático de *Software*.

Na figura, etapas de desenvolvimento são representadas por *documentos* ou especificações VDM-SL. O documento inicial é chamado de *especificação abstrata*. *Reificação de dados* é o processo de refinar ou *concretizar* (tornar próximo de estruturas de dados usadas na fase de implementação) os tipos de dados iniciais tomando decisões quanto às suas representações e manipulações pelas operações e funções. Ao final da reificação de dados temos um documento final ou *especificação concreta*. *Decomposição de operações* é o processo de decompor as operações de uma especificação concreta em construções presentes em uma linguagem de implementação tais como comandos de atribuição, sequências, comandos condicionais, laços, etc. Ao final da decomposição de operações temos um programa que implementa o sistema especificado através do documento inicial.

A partir dessas idéias organizamos o capítulo como segue. Na seção 5.2 apresentamos uma especificação abstrata de um SMA para os agentes MATHEMA. Esta especifição forma um documento inicial. Partindo deste documento realizamos nas seções seguintes um passo no processo de reificação de dados. Neste passo incorporamos algumas decisões de projeto à especificação abstrata do SMA proposto. Na seção 5.3 mostramos como (re)definir a representação por sobreposição com incerteza do SMA a partir do *Curriculum* presente no Sistema Tutor dos agentes MATHEMA. Na seção 5.4 discutimos a aquisição por análise de passos do SMA à luz da resolução de problemas por meio de regras produção. Na seção 5.5 especializamos o processo de manutenção

de sobreposição difusa do SMA adotando um algoritmo específico para a atualização da medida de incerteza. Ao final, seção 5.6, juntamos os resultados do processo de reificação em um documento final e discutimos a utilidade deste documento final.

5.2 O Sistema de Modelagem do Aprendiz

5.2.1 Requisitos

O Sistema de Modelagem do Aprendiz que propomos tem as seguintes características principais:

Modelo do Aprendiz

- *Grau de especialização*: individual;
- *Duração temporal*: permanente;
- *Informação modelada*: conhecimento do aprendiz em relação ao domínio de aprendizagem;
- *Uso*: escolha de problemas;

Processo de Modelagem

- *Entrada*: resolução de problemas;
- *Método de representação*: sobreposição com representação de incerteza;
- *Método de aquisição*: análise dos passos de resolução;
- *Método de manutenção*: tratamento de incerteza através de conjuntos difusos.

Essas características formam um possível caminho por entre as direções gerais para a modelagem do aprendiz nos agentes MATHEMA apontadas no capítulo anterior. Destas características merecem comentários as escolhas que fizemos quanto ao que modelar, como representar, adquirir e manter.

Conforme discutimos na seção 4.2, nos agentes MATHEMA uma das funções pedagógicas mais importantes é a escolha de problemas a serem passados ao aluno. Desta forma decidimos modelar o estado de conhecimento do aluno tendo como principal uso a escolha de problemas. O uso das informações sobre o conhecimento do aprendiz na escolha de problemas será o tema do próximo capítulo.

Uma vez decidido modelar o estado de conhecimentos dos aprendizes escolhemos como método de representação a sobreposição com representação de incerteza. Este foi o método que recomendamos na seção 4.4.1. O método de aquisição é a análise de passos de resolução. Conforme dissemos na seção 4.4.2, qualquer um dos métodos apresentados no Capítulo 3 poderia ser empregado. Escolhemos a análise de passos por acreditarmos ser este método mais factível computacionalmente. Ver comentários na seção 3.4.2, página 45. O método de manutenção é o tratamento de incerteza através de conjuntos difusos.

A partir dessas características ou requisitos apresentamos a seguir uma especificação abstrata VDM-SL para o SMA. A especificação está dividida em duas partes: uma *Geral* e outra *Particular*.

5.2.2 Especificação Abstrata Geral

A especificação abstrata geral, Figura 5.2, define os aspectos básicos de um SMA para os agentes MATHEMA como proposto na Figura 4.3 no capítulo anterior.

Comecemos pela linha 3; nesta linha definimos o *estado* do SMA. Em uma especificação VDM-SL, o estado é uma coleção de objetos externos visíveis pelas operações do sistema sendo especificado. No nosso caso o estado consiste no Modelo do Aprendiz (linha 3.1). O Modelo do Aprendiz (linha 2.0) é especificado como um mapeamento entre nomes de aprendizes (linha 1.0) e representações de informações sobre os aprendizes. Para o SMA que propomos, a representação de informações deve ser feita por sobreposição com representação de incerteza e será especificada a seguir. A especificação do estado do SMA como mapeamento entre nomes e representações corresponde a noção de que nos agentes MATHEMA o Modelo do Aprendiz deve ser individual. Ver requisitos acima e seção 4.3.1.

Na linha 4 especificamos o processo de aquisição por meio de uma operação *AQUISICAO*. Esta operação tem três entradas: uma observação (linha 4.0), um nome de aprendiz (linha 4.0) e o estado, ou seja, o Modelo do Aprendiz (linha 4.1). A saída são informações sobre o estado de conhecimento do aprendiz (linha 4.1). Como precondição, a operação *AQUISICAO* requer que o nome do aprendiz já esteja associado a uma representação e conste no Modelo do Aprendiz (linha 4.2). Ainda, requer que tanto a observação quanto a representação do aprendiz satisfaçam as precondições de um método de aquisição particular (linha 4.3). Para o SMA que propomos, este método de aquisição é a análise de passos de resolução e será especificado a seguir. Como

types

1.0 $Anome = \mathsf{char}^{+}$;

2.0 $MA = Anome \xrightarrow{m} Representacao$

3.0 state SMA of
.1 $ma : MA$
.2 end

operations

4.0 $AQUISICAO\,(obs : Observacao, a : Anome)\ aif : Ainfo$
.1 ext rd $ma : MA$
.2 pre $a \in \mathsf{dom}\ ma \wedge$
.3 pre-$aquisicao\,(obs, ma\,(a))$
.4 post post-$aquisicao\,(obs, ma\,(a), aif) \wedge$
.5 pre-$manutencao\,(aif, ma\,(a))$;

5.0 $MANUTENCAO\,(aif : Ainfo, a : Anome)$
.1 ext wr $ma : MA$
.2 pre $a \in \mathsf{dom}\ ma \wedge$
.3 $\exists\, obs : Observacao \cdot$
.4 pre-$aquisicao\,(obs, ma\,(a)) \wedge$
.5 post-$aquisicao\,(obs, ma\,(a), aif) \wedge$
.6 pre-$manutencao\,(aif, ma\,(a))$
.7 post $\exists\, rep : Representacao \cdot$
.8 post-$manutencao\,(aif, \overleftarrow{ma}\,(a), rep) \wedge$
.9 $ma\,(a) = rep$;

6.0 $INCLUSAO\,(a : Anome)$
.1 ext wr $ma : MA$
.2 pre $a \notin \mathsf{dom}\ ma$
.3 post $\exists\, rep : Representacao \cdot$
.4 $ma\,(a) = rep$

Figura 5.2: Especificação VDM-SL: O Sistema de Modelagem do Aprendiz.

pós-condição, a operação *AQUISICAO* faz com que as informações obtidas a partir de uma observação e das atuais informações sobre o aprendiz sejam aquelas obtidas pelo método de aquisição particular (linha 4.4). Ainda, faz com que estas informações sejam tais que possam ser utilizadas pelo posterior processo de manutenção (linha 4.5).

Na linha 5 especificamos o processo de manutenção por meio de uma operação *MANUTENCAO* com três entradas: as informações provindas do processo de manutenção (linhas 5.0, 5.3 a 5.6), um nome de aprendiz (linha 5.0) e o Modelo do Aprendiz (linha 5.1). Dado que o aprendiz de nome a esteja representado no Modelo do Aprendiz (linha 5.2), a operação *MANUTENCAO* escreve no Modelo do Aprendiz (linha 5.9) uma nova representação para o aprendiz (linha 5.7). Representação esta obtida a partir de um método particular de manutenção (linha 5.8). No SMA que propomos, este método particular de manutenção é a manutenção com tratamento de incerteza através de conjuntos difusos e será especificado a seguir.

Em conjunto as operações *AQUISICAO* e *MANUTENCAO* especificam o processo de modelagem do aprendiz a ser realizado no Sistema Tutor dos agentes MATHEMA conforme discutido no capítulo anterior. Uma vez que o processo de modelagem do aprendiz só é realizado caso o nome do aprendiz já esteja associado a uma representação no Modelo do Aprendiz (linhas 4.2 e 5.2) há a necessidade de uma operação para associar uma representação a um novo nome de aprendiz. Esta operação é especificada na linha 6 e consiste em: tomar um nome que ainda não esteja no Modelo do Aprendiz (linha 6.2); criar um objeto para a representação de informações sobre este aprendiz (linha 6.3) e associar este objeto ao nome do aprendiz incluindo esta nova entrada no Modelo do Aprendiz (linha 6.4).

As operações *AQUISICAO*, *MANUTENCAO* e *INCLUSAO* são as operações básicas do SMA que propomos. Além destas, outras operações podem ser definidas. Uma, que pode ser utilizada por um módulo que utiliza as informações presentes no Modelo do Aprendiz, é a operação de consulta[5]. Uma especificação para esta operação deve ler o Modelo do Aprendiz, verificar se há uma representação associada a um nome de aprendiz de entrada e caso haja retornar esta representação.

A especificação abstrata geral, como dissemos, define aspectos gerais de SMA para os agentes MATHEMA como proposto no capítulo anterior. Serve assim como ponto de partida para especificações particulares. Ou seja, especificações comprometidas com métodos particulares de representação, aquisição e manutenção do Modelo do Aprendiz.

[5] Para o uso do Modelo do Aprendiz discutido no próximo capítulo não especificamos uma operação de consulta; ao invés disso fazemos o próprio Modelo do Aprendiz (linha 3) parte do estado das operações que utilizam o Modelo do Aprendiz. Ver seção 6.4

O que fazemos a seguir.

5.2.3 Especificação Abstrata Particular

A especificação abstrata particular reflete as decisões e escolhas que fizemos quanto a representação, aquisição e manutenção do Modelo do Aprendiz.

Comecemos pela representação, Figura 5.3, linha 7. A representação por sobreposição com incerteza é especificada como um objeto composto formado por um modelo de domínio de referência (linha 7.0) e um mapeamento entre sentenças e uma linguagem de representação de incerteza (linha 7.1).

types

7.0 $Representacao :: md : MD$

.1 $mc : Sentenca \xrightarrow{m} LI$

.2 inv mk-$Representacao\ (md, mc) \triangleq$

.3 dom $mc = md$;

8.0 $MD = Sentenca$-set;

9.0 $Sentenca = \text{char}^{+}$;

10.0 $LI = \mathbb{R}$

.1 inv $i \triangleq 0 \leq i \wedge i \leq 1$

Figura 5.3: Especificação VDM-SL: Representacao do Modelo do Aprendiz.

O modelo de domínio de referência é especificado como um conjunto de sentenças (linha 8.0), onde cada sentença é uma seqüência de um ou mais caracteres (linha 9.0). A linguagem de representação de incerteza de acordo com a escolha que fizemos são números reais no intervalo de 0 a 1 (linha 10). O *invariante* (linha 7.2), ou seja, uma propriedade comum a toda representação por sobreposição com representação de incerteza é que o domínio do mapeamento seja igual ao modelo de domínio de referência (linha 7.3). Esta especificação captura os aspectos essenciais da representação por sobreposição com incerteza discutida em detalhes na seção 3.3.1.

A estratégia de aquisição escolhida é a análise dos passos de resolução. Esta estratégia é especificada na Figura 5.4 por meio de uma função; linha 14. Uma função que deve ter como entrada uma observação (linha 11.0) e uma representação e gerar

como saída uma coleção de sentenças a serem incorporadas ao modelo do aprendiz (linha 13.0).

types

11.0 $Observacao = Passo \xrightarrow{m} Sentenca;$

12.0 $Passo = \mathsf{char}^{+};$

13.0 $Ainfo = Sentenca\text{-}\mathsf{set}$

functions

14.0 $aquisicao\,(obs : Observacao, rep : Representacao)\ aif : Ainfo$

.1 pre true

.2 post $aif = \{s \mid s \in rep.md \cdot$

.3 $\exists p \in \mathsf{dom}\ obs \cdot$

.4 $obs\,(p) = s\}$

Figura 5.4: Especificação VDM-SL: Aquisição do Modelo do Aprendiz.

Em mais detalhes, uma observação é especificada como um mapeamento entre passos (linha 12.0) e sentenças. Este mapeamento formaliza a idéia de que na análise de passos o sistema assume que cada passo de resolução corresponde à aplicação de conhecimentos (sentenças) de um modelo de resolução de problemas ideal (modelo de domínio de referência); ver seção 3.4.2, página 45. A pós-condição da função de aquisição formaliza o processo de inferência dos conhecimentos (sentenças) pertencentes ao modelo de resolução de problemas (linha 14.2) tais que, para cada conhecimento, existe um passo observado (linha 14.3) a que o conhecimento corresponde (linha 14.3). A coleção dos conhecimentos (sentenças) aplicados é a informação de saída do processo de aquisição (linha 14.2).

Por fim o processo de manutenção, Figura 5.5. O processo de manutenção também é expresso através de uma função; linha 15. Uma função que deve ter como entrada um conjunto de sentenças e uma representação e gerar como saída uma nova representação (linha 15.0). A nova representação é criada a partir da antiga substituindo o antigo mapeamento entre sentenças e números reais entre 0 e 1 por um novo mapeamento mc' (linha 15.5). O novo mapeamento mc' é gerado a partir do antigo substituindo antigas associações entre sentenças valores numéricos por novas associações (linha 15.4). Novas associações (linha 15.2) onde apenas associações (linha 15.4). Novas associações (linha 15.2) onde apenas sentenças (informações) de entrada (linha 15.3) são consideradas.

```
functions

15.0  manutencao (aif : Ainfo, rep : Representacao) rep' : Representacao
 .1   pre true
 .2   post ∃ m : Sentenca -m-> LI ·
 .3             dom m = aif ∩ rep.md ∧
 .4             let mc' = rep.mc † m in
 .5             rep' = mk-Representacao (rep.md, mc')
```

Figura 5.5: Especificação VDM-SL: Manutenção do Modelo do Aprendiz.

Esta função captura as características básicas do processo de manutenção com tratamento de incerteza baseado em sobreposição difusa discutido em detalhes na seção 3.5.2.

Completamos então a especificação abstrata VDM-SL de nossa proposta de SMA para os agentes MATHEMA. O documento inicial no processo de desenvolvimento sistemático ilustrado na Figura 5.1. No que segue tratamos de refinar este documento inicial.

5.3 O Modelo do Aprendiz

Nesta seção mostramos como elementos da arquitetura do Sistema Tutor dos agentes MATHEMA são incorporados em uma especificação mais concreta da representação por sobreposição com incerteza.

5.3.1 Bases de Conhecimento

Conforme dissemos na seção 4.4.1, há duas bases de conhecimento no Sistema Tutor dos agentes MATHEMA que podem fazer o papel de modelo de domínio na representação por sobreposição com incerteza: o *Curriculum* e o Conhecimento de Resolução de Problemas. Vejamos então como definir formalmente essas bases de conhecimento.

As bases de conhecimento que fazem parte do Sistema Tutor dos agentes MATHEMA (seção 4.2.3) estão especificadas na Figura 5.6, linha 22. São três bases: o *Curriculum* (linha 22.0) o Domínio de Problemas (linha 22.1) e o Conhecimento de Resolução de Problemas (linha 22.2).

O *Curriculum*. O *Curriculum* foi discutido em detalhes na seção 2.4.2. Relem-

types

16.0 $UndPedagogica = UndConhecimento\text{-set};$

17.0 $ParticaoDij = UndPedagogica\text{-set}$

.1 inv $prt \triangleq$

.2 $\{\} \notin prt \wedge$

.3 $\forall up1, up2 \in prt \cdot$

.4 $(up1 \neq up2) \Rightarrow (up1 \cap up2 = \{\});$

18.0 $Ordem = (UndPedagogica \times UndPedagogica)\text{-set}$

.1 inv $ord \triangleq$

.2 $\forall up, up1, up2, up3 : UndPedagogica \cdot$

.3 $(\mathsf{mk}\text{-}(up, up) \in ord) \wedge$

.4 $(\mathsf{mk}\text{-}(up1, up2) \in ord \wedge \mathsf{mk}\text{-}(up2, up1) \in ord \Rightarrow up1 = up2) \wedge$

.5 $(\mathsf{mk}\text{-}(up1, up2) \in ord \wedge \mathsf{mk}\text{-}(up2, up3) \in ord \Rightarrow \mathsf{mk}\text{-}(up1, up3) \in ord);$

19.0 $Curriculum :: prt : ParticaoDij$

.1 $ord : Ordem$

.2 inv $\mathsf{mk}\text{-}Curriculum\,(prt, ord) \triangleq$

.3 $\forall \mathsf{mk}\text{-}(up1, up2) \in ord \cdot$

.4 $up1 \in prt \wedge up2 \in prt;$

20.0 $Enunciado = \mathsf{char}^{+};$

21.0 $Problema :: enc : Enunciado$

.1 $ucs : UndConhecimento\text{-set}$

.2 inv $\mathsf{mk}\text{-}Problema\,(\text{-}, ucs) \triangleq ucs \neq \{\};$

22.0 $BasesConhecimento :: cur : Curriculum$

.1 $dpr : Problema\text{-set}$

.2 $crp : UndConhecimento\text{-set}$

.3 inv $\mathsf{mk}\text{-}BasesConhecimento\,(cur, dpr, crp) \triangleq$

.4 $dpr \neq \{\} \wedge$

.5 $crp \neq \{\} \wedge$

.6 $\bigcup cur.prt = crp \wedge$

.7 $\forall p \in dpr \cdot$

.8 $p.ucs \cap crp \neq \{\}$

Figura 5.6: Especificação VDM-SL: Bases de Conhecimento.

bremos. O primeiro passo na definição do *Curriculum* (linha 19.0) é estabelecer um particionamento do subdomínio d_{ij} (associado ao agente tutor, seção 2.5.1) em unidades pedagógicas (linha 17.0). Neste caso cada unidade pedagógica é um conjunto de unidades de conhecimento relativas ao subdomínio d_{ij}. Na seção 5.3.3 definiremos unidade de conhecimento. Por enquanto notemos que o particionamento possui um invariante (linhas 17.1 a 17.4), uma propriedade que o torna uma *partição* em sentido matemático: um conjunto de conjuntos ao qual o conjunto vazio não pertence (linha 17.2); no qual quaisquer dois conjuntos elementos são disjuntos (linhas, 17.3 e 17.4) e cuja união de seus elementos é igual ao conjunto sendo particionado.

O segundo passo na definição do *Curriculum* (linha 19.1) é estabelecer um ordenamento entre as unidades pedagógicas (linha 18.0). Um ordenamento ou relação de ordem (parcial) sobre um conjunto A é uma relação (conjunto de pares ordenados) reflexiva (linha 18.3), anti-simétrica (linha 18.4) e transitiva (linha 18.5) sobre o conjunto A. O invariante (linhas 19.2-19.4) garante que a ordem em um *Curriculum* é sobre a partição do *Curriculum*.

O Domínio de Problemas. O Domínio de Problemas (linha 22.1) consiste em uma coleção (não vazia, linha 22.4) de problemas que o Sistema Tutor dispõe para propor ao aprendiz; ver seção 2.4.2. Um problema (linha 21) é caracterizado por um enunciado (linha 21.0) e um conjunto de unidades de conhecimento necessárias a sua resolução (linha 21.1). Como veremos no próximo capítulo a escolha dos problemas a serem passados aos aprendizes será feita comparando-se as unidades de conhecimento associadas a um problema com o estado atual de conhecimento do aprendiz. A classe de problemas possui um invariante (linha 22.2): um problema só é um problema se envolver unidades de conhecimento em sua resolução.

O Conhecimento de Resolução de Problemas. O Conhecimento de Resolução de Problemas (linha 22.3) consiste em um conjunto (não vazio, linha 22.5) de unidades de conhecimento. É, em essência, o conhecimento de um agente tutor MATHEMA a respeito de um subdomínio; é o conhecimento que habilita o agente tutor a responder problemas para o aprendiz quando este assim deseja. Como veremos a seguir, esta base de conhecimento será caracterizada como um conjunto *regras se-então* passíveis de serem utilizadas por um *sistema de produção*.

As bases de conhecimento estão relacionadas pelo invariante (linhas 22.6 a 22.8). A união das unidades pedagógicas do *Curriculum* deve ser igual ao Conhecimento de Resolução de Problemas (linha 22.6). E, os problemas pertencentes ao Domínio de Problemas devem envolver ao menos uma uma unidade de conhecimento presente do Conhecimento de Resolução de Problemas (linhas 22.7 e 22.8). Estas duas propriedades

fazem com que *Curriculum* e Domínio de Problemas formem uma *estrutura pedagógica* definida sobre o Conhecimento de Resolução de Problemas relativo a um subdomínio d_{ij} de um domínio de aprendizagem; ver seção 2.4.2.

5.3.2 Representação por Sobreposição com Incerteza

Adotando o *Curriculum* como modelo de domínio, passamos a apresentar uma nova especificação para representação por sobreposição com incerteza. Discutimos também como esta nova especificação é dita reificar a especificação abstrata da Figura 5.3.

O *Curriculum* é a base para o estabelecimento de seqüências nas quais o conteúdo de um subdomínio d_{ij} pode ser aprendido; ver seção 2.4.2. Uma seqüência de aprendizagem consiste em uma enumeração de unidades pedagógicas de um *Curriculum* respeitando a ordem existente entre elas. Uma versão formal dessa noção é apresentada na Figura 5.7, linha 23. Em essência um objeto composto formado por uma sequência *up* de elementos *UndPedagogica* (Figura 5.6, linha 16) e por um *Curriculum* (Figura 5.6, linha 19). Notemos o invariante; as linhas 23.3 a 23.5 nos dizem que a seqüência *up* forma uma enumeração sobre o *Curriculum*, ou seja, uma associação um-a-um entre conjunto de índices e unidades pedagógicas do *Curriculum*. A linha 23.6 nos mostra que a seqüência *up* deve respeitar a ordem existente entre as unidades pedagógicas do *Curriculum*. As linhas 23.7 a 23.9 serão explicadas na seção 5.4.3.

Usando a noção de seqüência de aprendizagem, apresentamos na linha 24 uma nova especificação para a representação do modelo do aprendiz. A nova especificação é composta por uma seqüência de aprendizagem *mp* (linha 24.0) e por um mapeamento *mc* entre unidades de conhecimento e números reais entre 0 e 1 (linha 24.1). A seqüência de aprendizagem *mp* retém informações sobre um caminho de aprendizagem sendo trilhado pelo aprendiz por entre as unidades pedagógicas do *Curriculum*. O mapeamento *mc* é uma estimativa numérica feita pelo SMA a respeito do conhecimento do aprendiz em relação a cada unidade de conhecimento pertencente às unidades pedagógicas do *Curriculum* (linhas 24.2 e 24.3). Essa especificação tem como objetivo básico favorecer o processo de escolha de problemas. Este ponto será discutido no próximo capítulo.

A nova especificação é uma reificação da especificação abstrata mostrada na Figura 5.3. Essa noção é formalizada com o auxílio de uma função "*retrieve*" definida entre a atual e a anterior especificação (linha 26).

A função "*retrieve*" é o elemento fundamental no processo de reificação de dados no desenvolvimento sistemático usando VDM [Jon90, cap.8]. Por meio dessa função

types

23.0 $SequenciaUP :: up : UndPedagogica^*$
.1 $cur : Curriculum$

.2 inv mk-$SequenciaUP\,(up, \text{mk-}Curriculum\,(prt, ord))\ \triangleq$
.3 $(\forall\, i, j \in \text{inds } up \cdot$
.4 $(up\,(i) \in prt) \wedge$
.5 $(up\,(i) = up\,(j) \ \Rightarrow\ i = j) \wedge$
.6 $(\text{mk-}(up\,(i), up\,(j)) \in ord \ \Rightarrow\ i \leq j)) \wedge$
.7 $(\neg \exists\, upx \in prt \setminus \text{elems } up \cdot$
.8 let $n = \text{len } up$ in
.9 $\text{mk-}(upx, up\,(n)) \in ord);$

24.0 $Representacao\text{-}1 :: md : SequenciaUP$
.1 $mc : UndConhecimento \xrightarrow{m} LI$

.2 inv mk-$Representacao\text{-}1\,(md, mc)\ \triangleq$
.3 $\text{dom } mc = \bigcup \text{elems } md.up$

functions

25.0 $retr\text{-}MD : SequenciaUP \to MD$
.1 $retr\text{-}MD\,(md)\ \triangleq$
.2 $\{retr\text{-}Sentenca\,(uc) \mid uc \in \bigcup \text{elems } md.up\};$

26.0 $retr\text{-}Representacao : Representacao\text{-}1 \to Representacao$
.1 $retr\text{-}Representacao\,(\text{mk-}Representacao\text{-}1\,(md, mc))\ \triangleq$
.2 let $md' = retr\text{-}MD\,(md)$ in
.3 let $mc' = \{retr\text{-}Sentenca\,(uc) \mapsto mc\,(uc) \mid uc \in \text{dom } mc\}$ in
.4 $\text{mk-}Representacao\,(md', mc')$

Figura 5.7: Especificação VDM-SL: Representação por Sobreposição com Incerteza.

podemos mostrar que um passo no processo de reificação é "*correto*". Em VDM diz-se que um passo de reificação de dados é "*correto*" se: (i) existe uma função total $retr : E_c \rightarrow E_a$, onde E_a, E_c são duas especificações de tipos de dados, e (ii) esta função é tal que: $\forall\, a \in E_a \cdot \exists\, c \in E_c \cdot retr(c) = a$. Estas condições garantem que (1) todo objeto especificado por E_c corresponde a um objeto abstrato E_a; ou seja, um objeto E_c é uma nova forma, comprometida com decisões de projeto, de vermos algum objeto E_a; (2) todos os objetos anteriormente especificados por E_a têm uma nova forma especificada por E_c.

Proposição 5.3.2.1

(i) A função *retr-Representacao* : *Representacao-1* $\rightarrow$ *Representacao* é total

(ii) $\forall\, r$: *Representacao* $\cdot \exists\, r_1$: *Representacao-1* $\cdot$ *retr-Representacao*$(r_1) = r$

A demonstração da proposição 5.3.2.1 encontra-se no Apêndice B.

Pela proposição 5.3.2.1 a nova especificação para a representação por sobreposição com incerteza mostrada na Figura 5.7 é dita reificar a especificação abstrata mostrada na Figura 5.3.

5.3.3 Unidades de Conhecimento

Na seção 2.4.2 dissemos que unidades de conhecimento eram basicamente *conceitos* ou *habilidades* que um aluno irá aprender a partir de atividades de resolução de problemas. A questão agora é: como codificar estes conceitos ou habilidades de resolução de problemas? A pesquisa em representação de conhecimento da Inteligência Artificial nos dá algumas alternativas.

Uma delas, a que iremos adotar na especificação do SMA que propomos é a codificação por meio de *regras de produção*; Figura 5.8, linha 27. As regras de produção (linha 28) representam o conhecimento de resolução de problemas através de relações entre ações (linhas 28.1, 30.0) que uma pessoa pode tomar ao perceber que determinadas condições (linhas 28.0, 29.0) são satisfeitas durante a resolução de um problema. O encadeamento de ações, ou seja, de aplicações de regras conduzem à resolução de problemas; seção 5.4.1.

Há vários domínios nos quais o conhecimento a ser aprendido pode ser pensado como regras de produção e o processo de resolução de problemas como aplicação de regras. Para citar alguns:

types

27.0 $UndConhecimento = Regra;$

28.0 $Regra :: se : Condicao^*$
.1 $entao : Acao$

29.0 $Condicao = \mathsf{char}^+;$

30.0 $Acao = \mathsf{char}^+$

functions

31.0 $retr\text{-}Sentenca : UndConhecimento \rightarrow Sentenca$
.1 $retr\text{-}Sentenca\ (\mathsf{mk}\text{-}Regra\ (se, entao)) \triangleq$
.2 $\mathsf{conc}\ [se\ (i) \mid i \in \mathsf{inds}\ se] \frown entao$

Figura 5.8: Especificação VDM-SL: Unidades de Conhecimento.

- Praticamente todas as áreas da matemática de 1 e 2 graus ensinada nas escolas;
- Áreas da física tais como cinemática, eletricidade, ótica, etc;
- Diagnóstico especializado tais como diagnóstico doenças, defeitos em equipamentos, etc.

Este fato é a motivação básica para a nossa especificação por meio de regras de produção para o conhecimento de resolução de problemas e, por conseguinte, o estado de conhecimento dos aprendizes que iteragem com os agentes MATHEMA. Notemos ainda que as regras de produção têm sido um dos principais mecanismos utilizados em Inteligência Artificial e Ciências Cognitivas para codificar conhecimento de resolução de problemas em várias aplicações. Em especial na construção de ambientes de aprendizagem por computador. Os tutores cognitivos desenvolvido a partir do sistema de produção ACT-R são um bom exemplo do que dizemos [ACKP95].

Uma vez que tomamos mais uma decisão de projeto, vejamos como a definição de unidade de conhecimento como regras de produção é dita reificar o tipo de dados *Sentenca* presente na especificação abstrata da representação por sobreposição com incerteza mostrada na Figura 5.3.

Proposição 5.3.3.1

(i) A função *retr-Sentenca*: $UndConhecimento \rightarrow Sentenca$ é total

(ii) $\forall s : Sentenca \cdot \exists uc : UndConhecimento \cdot retr\text{-}Sentenca(uc) = s$

A demonstração da proposição 5.3.3.1 encontra-se no Apêndice B.

5.4 A Estratégia de Aquisição

A decisão de considerar unidades de conhecimento como regras de produção tem influência direta na estratégia de aquisição de informações a partir da análise de passos de resolução de problemas. Nesta seção mostramos como modelar a função de aquisição especificada na Figura 5.4 considerando a resolução de problemas por meio de regras de produção. Mostramos ainda como estender o processo de aquisição para lidar com informações sobre a seqüência de aprendizagem seguida por um aprendiz ao longo do *Curriculum*.

5.4.1 A Resolução de Problemas

O modelo de ensino e aprendizagem adotado pelos agentes tutores MATHEMA contempla a aprendizagem por resolução de problemas a partir de duas modalidades: ver seção 4.2.1. Em uma delas o agente tutor MATHEMA escolhe, *mediante certos critérios*, um problema e o apresenta ao aprendiz. O aprendiz tenta então resolvê-lo. A resposta do aprendiz, seus passos ao resolver o problema, formam o insumo principal para o processo de aquisição do SMA que propomos neste capítulo.

O processo de resolução de problemas, como consideramos nesta seção, é um encadeamento de ações resultantes de aplicações de regras de produções que conduzem a respostas para problemas. Um problema neste contexto é visto como uma coleção de dados ou fatos que iniciam um processo de aplicação de regras. Na Figura 5.9 ilustramos o processo de resolução de problemas com uma resolução para um problema no domínio estudo de retas apresentado no Capítulo 2, seção 2.4.2.

Na Figura 5.9 vemos que iniciamente o aprendiz identifica o conjunto de dados do problema. Após, passa a aplicar regras realizando três passos antes de chegar à resposta final do problema. As regras aplicadas em cada passo aparecem logo ao lado da resolução do problema.

A idéia básica por trás desse processo de resolução de problemas utilizando regras de produção é capturada por meio da especificação mostradas na Figura 5.10. A resolução de problemas conforme vemos na linha 32.0 é formalizada por meio de uma seqüência de objetos compostos *Aplicacao*. O tipo de dados *Aplicacao* (linha 33) representa

Problema: Escreva a equação da reta r que passa pelo ponto $A(3,2)$ e é perpendicular à reta $s : y = -2x + 1$.

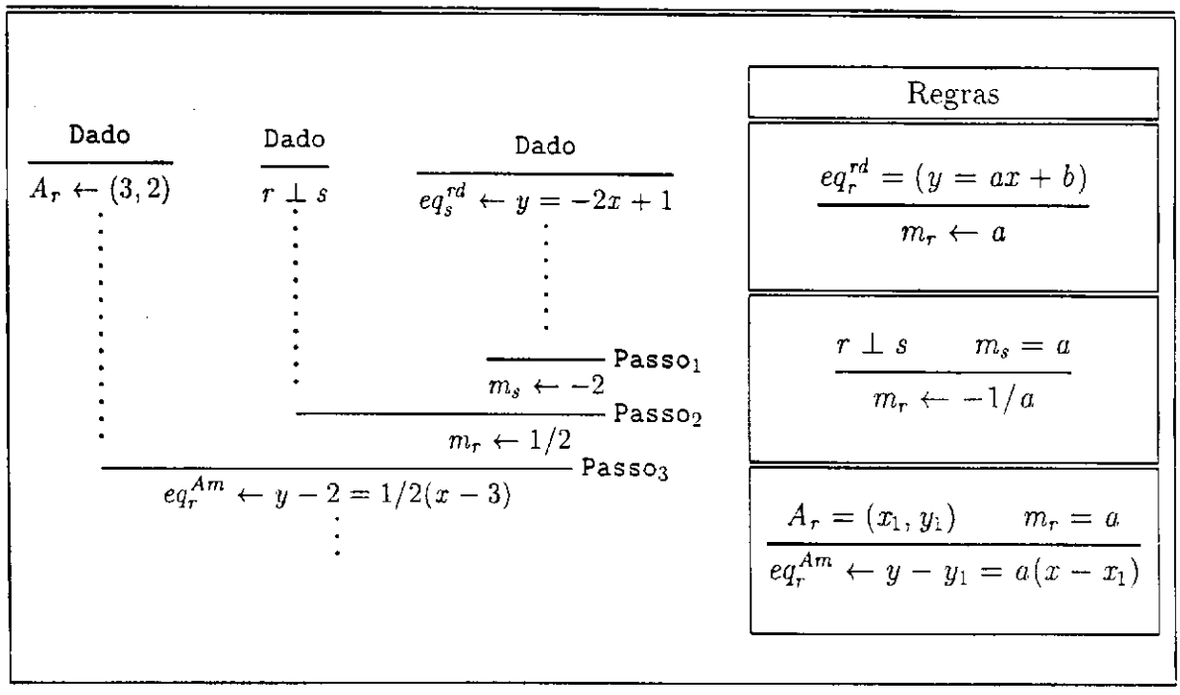


Figura 5.9: A resolução de problemas.

as ações ou passos do aprendiz durante o processo de resolução de problemas. Um passo ou ação (linha 33.2) é o resultado da aplicação de uma regra (linha 33.1). Esta regra, dependendo do conhecimento do aprendiz, deve estar entre um conjunto de regras aplicáveis em dado momento (linha 33.0). Notemos que associada a cada regra aplicável em dado momento está uma ação que corresponde ao resultado da aplicação da regra (linha 34.0). Essa estrutura retém dados importantes e necessários à estratégia de aquisição de informações sobre o estado de conhecimento do aprendiz que apresentamos a seguir.

As informações a serem obtidas pela estratégia de aquisição são conjuntos de regras (linha 35): regras que o aprendiz aplicou de forma correta (linha 35.2) ou de forma errada (linhas 35.0 e 35.1) durante a resolução de um problema. Há assim dois conjuntos de regras que foram utilizadas de forma errada e apenas um de regras utilizadas de forma correta. O porquê distinguir entre níveis de erros e por que apenas um nível para acertos será explicado na seção 5.5. Adiantamos que esta estrutura para as informações geradas pelo processo de aquisição é ditada pela estratégia de manutenção do modelo do aprendiz que adotamos.

Mais uma vez, tendo feito decisões quanto à estrutura representacional de objetos anteriormente especificados de forma abstrata (Figura 5.4), temos que mostrar como a

types

32.0 $RslProblema = Aplicacao^*$;

33.0 $Aplicacao :: ra : RegrasAplicaveis$
.1 $r : [Regra]$
.2 $a : Acao$

34.0 $RegrasAplicaveis = Regra \xrightarrow{m} Acao$;

35.0 $Regras :: err\text{-}2 : Regra\text{-set}$
.1 $err\text{-}1 : Regra\text{-set}$
.2 $crr\text{-}1 : Regra\text{-set}$

functions

36.0 $retr\text{-}Passo : Aplicacao \to Passo$
.1 $retr\text{-}Passo\ (\mathsf{mk}\text{-}Aplicacao\ (\text{-}, \text{-}, acao)) \triangleq$
.2 $acao$;

37.0 $retr\text{-}Observacao : RslProblema \to Observacao$
.1 $retr\text{-}Observacao\ (rsl) \triangleq$
.2 $\{retr\text{-}Passo\ (p) \mapsto retr\text{-}Sentenca\ (r) \mid p \in \mathsf{elems}\ rsl, r : Regra \cdot$
.3 $\mathsf{if}\ p.r = \mathsf{nil}$
.4 $\mathsf{then}\ \exists r' \in \mathsf{dom}\ p.ra \cdot r = r'$
.5 $\mathsf{else}\ r = p.r\}$;

38.0 $retr\text{-}Ainfo : Regras \to Ainfo$
.1 $retr\text{-}Ainfo\ (\mathsf{mk}\text{-}Regras\ (err\text{-}2, err\text{-}1, crr\text{-}1)) \triangleq$
.2 $\{retr\text{-}Sentenca\ (r) \mid r \in \bigcup \{err\text{-}2, err\text{-}1, crr\text{-}1\}\}$

Figura 5.10: Especificação VDM-SL: Resolução de Problemas, Informações.

nova especificação relaciona-se com a antiga. De forma rápida. *RslProblema* (linha 32) reifica *Observacao* (Figura 5.4, linha 11) através da função *retr-Observacao* (linha 37). E, *Regras* (linha 35) reifica *Ainfo* (Figura 5.4, linha 13) através da função *retr-Ainfo* (linha 38).

Proposição 5.4.1.1

(i) A função $retr\text{-}Observacao : RslProblema \rightarrow Observacao$ é total

(ii) $\forall\, obs : Observacao \cdot \exists\, rsl : RslProblema \cdot retr\text{-}Observacao(rsl) = obs$

A demonstração da proposição 5.4.1.1 encontra-se no Apêndice B.

Proposição 5.4.1.2

(i) A função $retr\text{-}Ainfo : Regras \rightarrow Ainfo$ é total

(ii) $\forall\, i : Ainfo \cdot \exists\, rs : Regras \cdot retr\text{-}Ainfo(rs) = i$

A demonstração da proposição 5.4.1.2 encontra-se no Apêndice B.

5.4.2 Análise dos Passos de Resolução

Em um dado passo do processo de resolução de problema podem ocorrer três situações distintas:

(a) O aprendiz não sabe que regra aplicar;

(b) O aprendiz pensa que sabe que regra aplicar e tenta uma regra não aplicável;

(c) O aprendiz sabe que regra aplicar mas não a aplica corretamente;

(d) O aprendiz sabe que regra aplicar e a aplica corretamente.

Por exemplo, na resolução de problemas mostrada na Figura 5.9, o aprendiz poderia identificar os dados do problema e não saber que regra aplicar inicialmente (situação (a)). Ou então, o aprendiz poderia ter realizado o primeiro passo, ou seja, saber como identificar o coeficiente angular em uma equação reduzida da reta e identificar corretamente (situação (d)) e, no segundo passo pensar em aplicar a regra que relaciona retas paralelas e coeficientes angulares ao invés da regra que relaciona retas perpendiculares e coeficientes angulares (situação (b)). Isto devido a vários motivos: ter se confundido com os dados iniciais, ter esquecido a regra certa, não saber a forma da regra, etc.

Por fim, o aprendiz poderia ter realizado o primeiro passo (situação (d)) e, no segundo, escolher a regra adequada mas por descuido, falta de atenção ou por não saber aplicar a regrar corretamente chegar a um resultado incorreto tal como $m_r = -1/2$ (situação (c)). Um resultado incorreto, ressaltamos, em relação às premissas utilizadas na aplicação da regra[6].

O processo de aquisição de informações sobre o estado de conhecimento de um aprendiz, que agora passamos a discutir, parte de uma resolução de problemas, analisa as quatro situações acima e resume as informações obtidas em uma estrutura contendo regras aplicadas corretamente e regras aplicadas de forma errada. O processo é especificado formalmente por meio da função *analise-passos-resolucao* mostrada na Figura 5.11, linha 39. A função tem como parâmetros uma resolução de problema *RslProblema* (Figura 5.10, linha 32) e uma representação de modelo do aprendiz *Representacao-1* (Figura 5.7, linha 24) e produz como valor de saída conjuntos de regras *Regras* (Figura 5.10, linha 35).

Sendo *rsl* e *rep*, respectivamente, parâmetros de entradas para a função *analise-passos-resolucao*, o valor de saída *rs* será igual a mk-*Regras*(*err*-2, *err*-1, *crr*-1) (linha 39.12) onde: o componente *err*-2 (linha 39.2) consiste no conjunto das regras r pertencentes ao domínio do mapeamento *mc* componente de *rep* (ver definição na Figura 5.7, linha 24.1) tais que: há um passo de resolução (linha 39.3) onde r era aplicável mas o aprendiz não soube que regra aplicar (linha 39.4) (situação (a)); ou então, o aprendiz aplicou r mas r não era aplicável no passo em questão (linha 39.5) (situação (b)).

Nesse ponto, relembremos a definição do tipo de dados *Aplicacao* (Figura 5.10, linha 33) e notemos como seus componentes veiculuam informações necessários ao processo de análise de passos. O componente *ra* resume as regras que podem ser aplicadas em um passo de resolução; r, um componente opcional, sendo igual a nil representa que nenhuma regra foi aplicada em um passo de resolução. Consideramos ainda que a construção da seqüência de objetos *Aplicacao* contendo essas informações fica a cargo de uma *interface* de resolução de problemas. A especificação dessa *interface* é uma

[6] A seguinte imagem também é ilustrativa: uma pessoa tentando abrir uma porta dispondo de um molho de chaves no qual há uma ou mais chaves que abrem a porta. Inicialmente a pessoa pode ficar confusa e não saber por onde começar; que chave usar (situação (a)). Depois a pessoa pode identificar uma chave que talvez sirva para abrir a porta; no entanto a chave não entra na fechadura, ou então, entra mais não gira (situação (b)). Depois de muito tentar a pessoa pode descobrir uma chave que abre a porta mas por já estar agastada tentar girar a chave ao contrário não conseguindo assim abrir a porta (situação (c)). Por fim a pessoa encontra uma chave correta e gira corretamente na fechadura abrindo a porta (situação (d)). Havendo mais portas que abrir com as chaves o descrito acima é apenas um passo. Um passo de um processo de resolução de problemas.

```
functions

39.0  analise-passos-resolucao (rsl : RslProblema, rep : Representacao-1) rs : Regras
 .1   pre true
 .2   post let err-2 = {r | r ∈ dom rep.mc ·
 .3                          ∃p ∈ elems rsl ·
 .4                              r ∈ dom p.ra ∧ p.r = nil ∨
 .5                              p.r = r ∧ r ∉ dom p.ra},
 .6           err-1 = {r | r ∈ dom rep.mc ·
 .7                          ∃p ∈ elems rsl ·
 .8                              p.r = r ∧ r ∈ dom p.ra ∧ p.ra (r) ≠ p.a},
 .9           crr-1 = {r | r ∈ dom rep.mc ·
 .10                         ∃p ∈ elems rsl ·
 .11                             p.r = r ∧ r ∈ dom p.ra ∧ p.ra (r) = p.a} in
 .12      rs = mk-Regras (err-2, err-1, crr-1)
```

Figura 5.11: Especificação VDM-SL: Análise de Passos de Resolução.

das direções futuras de pesquisa que apontamos na conclusão, Capítulo 7. Por fim, queremos ressaltar que ao delegarmos a uma *interface* a tarefa de coletar e organizar os passos de resolução de tal modo a facilitar o processo de aquisição, seguimos uma das máximas que John Self sugeriu como forma de amenizar as dificuldades computacionais enfrentadas pelo processo de aquisição [Sel90]; ver seção 3.4.3.

O componente *err*-1 (linha 39.6) consiste no conjunto das regras r pertencentes ao domínio do mapeamento *mc* componente de *rep* tais que: há um passo de resolução (linha 39.7) onde r era aplicável, o aprendiz aplicou r, no entanto, de forma incorreta (linha 39.8) (situação (c)). Relembremos que $p.ra(r)$ é a ação esperada ao se aplicar a regra r.

O componente *crr*-1 (linha 39.9) consiste no conjunto das regras r pertencentes ao domínio do mapeamento *mc* componente de *rep* tais que: há um passo de resolução (linha 39.10) onde r era aplicável, o aprendiz aplicou r e de forma correta (linha 39.11) (situação (d)).

Assim especificada, a função *analise-passos-resolucao* classifica as regras empregadas em uma resolução de problemas em três categorias; duas das quais sendo regras aplicadas de forma incorreta. Essa classificação forma o insumo para o processo de manutenção e será discutida em mais detalhes na seção 5.5. A função *analise-passos-resolucao* é ainda dita modelar a função *aquisicao* especificada na Figura

5.4.

Em VDM [Jon90, cap.8] diz-se que uma função ou operação f_c especificada de forma implícita, ou seja, através de pré e pós-condições, modela uma outra função (abstrata) f_a também especificada de forma implícita se existem funções "*retrieve*" $retr\text{-}D_a : \mathsf{dom}\ f_c \to \mathsf{dom}\ f_a$ e $retr\text{-}I_a : \mathsf{rng}\ f_c \to \mathsf{rng}\ f_a$ tais que:

(i) $\forall x \in \mathsf{dom}\ f_c \cdot \mathsf{pre}\text{-}f_a(retr\text{-}D_a(x)) \Rightarrow \mathsf{pre}\text{-}f_c(x)$

(ii) $\forall x \in \mathsf{dom}\ f_c, y \in \mathsf{rng}\ f_c \cdot$
$\mathsf{pre}\text{-}f_a(retr\text{-}D_a(x)) \wedge \mathsf{post}\text{-}f_c(x, y) \Rightarrow \mathsf{post}\text{-}f_a(retr\text{-}D_a(x), retr\text{-}I_a(y))$

A modelagem de função em essência significa que o *comportamento* especificado por f_c, quando *visto por meio de* funções $retr\text{-}D_a$ e $retr\text{-}I_a$, é um comportamento previamente especificado por f_a. Aqui, o termo comportamento denota a relação estabelecida entre entradas e saídas por meio de uma função ou operação f. Ou seja, o conjunto $\{(x, y) \mid x \in \mathsf{dom}\ f, y \in \mathsf{rng}\ f \cdot \mathsf{pre}\text{-}f(x) \wedge \mathsf{post}\text{-}f(x, y)\}$. Através da noção de modelagem de funções podemos justificar rigorosa ou formalmente uma nova especificação para funções ou operações durante o processo de reificação de dados. O que é fazemos por meio da seguinte proposição:

Proposição 5.4.2.1

(i) $\forall\, rsl : RslProblema, rep_1 : Representacao\text{-}1 \cdot$
$\mathsf{pre}\text{-}aquisicao(retr\text{-}Observacao(rsl), retr\text{-}Representacao(rep_1)) \Rightarrow$
$\mathsf{pre}\text{-}analise\text{-}passos\text{-}resolucao(rsl, rep_1)$

(ii) $\forall\, rsl : RslProblema, rep_1 : Representacao\text{-}1, rs : Regras \cdot$
$\mathsf{pre}\text{-}aquisicao(retr\text{-}Observacao(rsl), retr\text{-}Representacao(rep_1)) \wedge$
$\mathsf{post}\text{-}analise\text{-}passos\text{-}resolucao(rsl, rep_1, rs) \Rightarrow$
$\mathsf{post}\text{-}aquisicao(retr\text{-}Observacao(rsl), retr\text{-}Representacao(rep_1), retr\text{-}Ainfo(rs))$

A demonstração da proposição 5.4.2.1 encontra-se no Apêndice B.

5.4.3 Aquisição Direta

A especificação da representação por sobreposição com incerteza apresentada na Figura 5.7 consiste em um objeto composto formado por uma seqüência de aprendizagem *md* (linha 24.0) e por um mapeamento *mc* entre unidades de conhecimento (regras) e números reais entre 0 e 1 (linha 24.1). As informações obtidas pela estratégia de análise de passos de resolução apresentada acima, conforme veremos na seção 5.5,

formam o insumo para a manutenção do mapeamento *mc*. As informações necessárias à manutenção da seqüência de aprendizagem *md*, o que agora passamos a discutir, são obtidas por meio de um processo de aquisição direta (seção 3.4.1).

Um seqüência de aprendizagem é uma enumeração na qual as unidades pedagógicas são dispostas (para estudo) respeitando a ordem imposta pelo *Curriculum* (Figura 5.7, linha 23). Durante o processo de aprendizagem por resolução de problemas o aluno segue uma seqüência de aprendizagem. O estabelecimento das unidades pedagógicas que compõem esta seqüência é um processo dinâmico e interativo que depende de vários fatores além do *Curriculum*; fatores tais como o rendimento do aluno, ou ainda, a escolha do próprio aluno quanto a que seqüência seguir. Este processo, colocado ao lado da análise de passos de resolução, completa a caracterização da etapa aquisição que propomos para o SMA dos agentes tutores MATHEMA.

Na Figura 5.12 especificamos uma nova função de aquisição que une a análise de passos de resolução com o estabelecimento da seqüência de aprendizagem. A função *aquisicao*-1 (linha 44) lida com duas modalidades de observação: uma direta e outra indireta (linha 40.0). A observação indireta consiste em uma resolução de problemas (linha 42.0); a observação direta consiste em uma unidade pedagógica (linha 41.0). Os nomes direta e indireta são uma alusão à forma com que estas observações são tratadas pela função de aquisição para gerar informações a serem incluídas no modelo do aprendiz. Distingue-se então entre aquisição direta (linha 44.5) e aquisição indireta (linha 44.7); na aquisição direta a observação é diretamente passada para a manutenção (linha 44.6); na aquisição indireta a observação (linha 44.8) é passada para a análise de passos que infere as informações a serem incluídas na representação do conhecimento do aprendiz (linha 44.9). Assim, a saída da função *aquisicao*-1 (linha 44.0) pode ser tanto a própria observação direta ou um objeto composto *Regras* criado pela análise de passos (linha 43.0).

Na aquisição direta é passada à função *aquisicao*-1 uma unidade pedagógica. Esta, corresponde a próxima unidade pedagógica que o aprendiz está apto a estudar. Como dissemos alguns fatores irão influenciar na escolha desta unidade pedagógica. A precondição da função *aquisicao*-1 é um deles. Sendo uma observação direta (linha 44.1), tem-se uma unidade pedagógica (linha 44.2); esta unidade pedagógica deve ser tal que concatenada com a atual seqüência de aprendizagem (linha 44.3) forme uma seqüencia de aprendizagem válida (linha 44.4). Em nossa discussão na seção 5.3.2 deixamos de falar sobre algumas linhas do invariante das seqüências de aprendizagem: as linhas 23.7 a 23.9, Figura 5.7. Estas linhas nos mostram que, em dado instante,

uma seqüência de aprendizagem válida é uma enumeração incompleta das unidades pedagógicas do *Curriculum*, em ordem, onde não há elementos que impedem a formação de uma enumeração completa. Ou como os símbolos nos dizem: não deve haver unidade pedagógica *upx* fora da atual seqüência (linha 23.7) de tamanho n (linha 23.8) tal que o último elemento ($up(n)$) seja, na ordem, precedido por *upx* (linha 23.9).

```
types
 40.0  Observacao-1 = Direta | Indireta;

 41.0  Direta :: UndPedagogica;

 42.0  Indireta :: RslProblema;

 43.0  Ainfo-1 = Direta | Regras
functions
 44.0  aquisicao-1 (obs : Observacao-1, rep : Representacao-1) aif : Ainfo-1
   .1  pre is-Direta (obs) ⇒ ∃ upx : UndPedagogica ·
   .2          obs = mk-Direta (upx) ∧
   .3          let up' = rep.md.up ↷ [upx] in
   .4          inv-SequenciaUP (mk-SequenciaUP (up', rep.md.cur))
   .5  post (is-Direta (obs) ∧
   .6        aif = obs) ∨
   .7       (is-Indireta (obs) ∧ ∃ rsl : RslProblema ·
   .8             obs = mk-Indireta (rsl) ∧
   .9             post-analise-passos-resolucao (rsl, rep, aif))
```

Figura 5.12: Especificação VDM-SL: Aquisição Direta.

Enquanto a função de *analise-passos-resolucao* (Figura 5.11) modela a função *aquisicao* (Figura 5.4), a função *aquisicao*-1 não modela adequadamente a função *aquisicao*. Quer dizer, o comportamento de *aquisicao*-1 quando visto por funções "*retrieve*" adequadas mostra-se diferente do comportamento especificado por *aquisicao*. Para constatarmos este fato notemos que a precondição pre-*aquisicao*(-,-) = true. Já a precondição pre-*aquisicao*-1(-,-) nem sempre é verdadeira. Assim não podemos provar que: $\forall$ -,- · pre-*aquisicao*(*retr*-, *retr*-) $\Rightarrow$ pre-*aquisicao*-1(-,-); a primeira condição necessária para que *aquisicao*-1 modele *aquisicao*. Então, qual a relação entre *aquisicao* e *aquisicao*-1?

Bem, para lidar com a representação por sobreposição com incerteza da forma como realmente desejamos, chegamos a conclusão, após tentar definir uma função *aquisicao*-1 que modele *aquisicao*, que nossa especificação inicial da etapa de aquisição mostra-se inadequada. Partimos da idéia de aquisição por análise de passos de resolução; ao longo do processo de reificação de dados incorporamos estruturas à especificação abstrata que nos levou a considerar a possibilidade de aquisição direta. Esta, em conjunto com a aquisição por análise de passos, requer uma especificação mais extensa. Percebendo estes fatos optamos por conservar a especificação original de *aquisicao* devido a sua simplicidade e por veicular a idéia central, a que queremos passar sobre a etapa de aquisição, e apresentar a função *aquisicao*-1 como um começo para repensarmos a especificação abstrata do processo de aquisição.

Além disso, podemos ainda argumentar que todo o comportamento especificado pela função *aquisicao* encontra-se contido no comportamento especificado por *aquisicao*-1. Quer dizer a relação estabelecida entre entradas e saídas pela função *aquisicao* é ainda preservada pela função *aquisicao*-1. Quanto a este fato notemos que na especificação da função *aquisicao*-1 quando uma observação de entrada é uma resolução de probemas a entrada é passada para a função *analise-passos-resolucao* (linhas 44.7 a 44.9). E esta, como mostramos anteriormente, modela a função *aquisicao*[7].

5.5 A Estratégia de Manutenção

A estratégia de aquisição discutida na seção anterior infere conjuntos de regras relacionados com uma resolução de problema. Ainda, pode ser estendido para lidar com unidades pedagógicas a serem acrescentadas à atual seqüência de aprendizagem. Nesta seção mostramos como modelar a função de manutenção especificada na Figura 5.5 de modo a incorporar essas informações na representação por sobreposição com incerteza discutida na seção 5.3.

5.5.1 Manutenção de Sobreposição Difusa

Em geral as informações sobre o estado de conhecimento do aprendiz produzidas pelo processo de aquisição são ditadas pela forma com que planejamos utilizar essas informações para atualizar a representação do estado de conhecimento do aprendiz. No nosso caso pretendemos utilizar o método de manutenção de sobreposição difusa

[7]No mais, esta preocupação é válida pois tentamos manter coerência com o método de desenvolvimento sistemático VDM.

que apresentamos na seção 3.5.2. Fora a motivação básica, há algumas considerações relacionadas a esta escolha particular que agora passamos a discutir.

A motivação básica para o emprego do método de manutenção de sobreposição difusa apresentado na seção 3.5.2 e, em última análise, da representação de incerteza por conjuntos difusos, com relação à outros métodos, alguns dos quais analisados na seção 3.5.2, é sua simplicidade aliada à sua adequação ao nosso objetivo de definir funções de seleção de problemas (próximo capítulo). Dizemos que o métodos é simples pois funciona a partir de uma única função de manutenção w' mostrada na Figura 5.13. Outros métodos, em especial os baseados em probabilidades, apesar de oferecerem

values

45.0 $t = 0.86$

functions

46.0 *manut-sobreposicao-difusa* (rs:*Regras*, rep:*Representacao*-1) rep':*Representacao*-1
.1 pre true
.2 post let $mc = rep.mc$ in
.3 let $mc' = mc \dagger \{r \mapsto w'(+1, mc(r)) \mid$
.4 $r \in \text{dom } mc \cdot$
.5 $r \in rs.crr\text{-}1\}$ in
.6 let $mc'' = mc' \dagger \{r \mapsto w'(-1, mc'(r)) \mid$
.7 $r \in \text{dom } mc' \cdot$
.8 $r \in rs.err\text{-}1\}$ in
.9 let $mc''' = mc'' \dagger \{r \mapsto w'(-2, mc''(r)) \mid$
.10 $r \in \text{dom } mc'' \cdot$
.11 $r \in rs.err\text{-}2\}$ in
.12 $rep' = \text{mk-}Representacao\text{-}1\ (rep.md, mc''')$;

47.0 $w' : \mathbb{Z} \times \mathbb{R} \to \mathbb{R}$
.1 $w'(k, w) \triangleq$
.2 if $k > 0$
.3 then $w \uparrow (1/(2 \uparrow k))$
.4 elseif $w < t$
.5 then $w \uparrow (2 \uparrow (\text{abs } k))$
.6 else $w \uparrow (\text{abs } k)$

Figura 5.13: Especificação VDM-SL: Manutenção de Sobreposicao Difusa.

um tratamento mais elaborado da incerteza inerente ao processo de modelagem do aprendiz, requerem estimativas de várias distribuições de probabilidades iniciais. Na conclusão, Capítulo 7, voltamos a este ponto como possível trabalho futuro.

Os detalhes de como a função w' (linha 47) funciona foram discutidos na seção 3.5.2. Um deles é que a função supõe que a etapa de aquisição gera hipóteses numéricas sobre o uso de unidades de conhecimento durante a resolução de problemas. O objeto composto *Regras* (Figura 5.10, linha 35), produzido ao final da análise de passos de resolução,méricas. Há três campos em um objeto *Regras* : *err*-2, *err*-1 e *crr*-1. Estes campos são conjuntos de regras, respectivamente, associadas aos valores $-2, -1$ e $+1$. Valores negativos correspondem a regras utilizadas de forma incorreta; valores positivos a regras utilizadas de forma correta. Valores mais negativos correspondem a erros mais graves. Consideramos que só há uma forma de utilizar uma regra corretamente. Estas informações são extraídas de uma resolução de problemas pela análise de passos de resolução considerando as quatro situações discutidas na seção 5.4.2. As duas primeiras, situações (a) e (b), são consideradas os erros mais graves. Às regras correspondentes a estas situações é associado o valor -2. A terceira, situação (c) é considerada um erro, porém, não tão grave. Às regras correspondentes é associado o valor -1. Por fim, a situação (d), envolve regras aplicadas corretamente. A estas regras é associado o valor $+1$.

O método de manutenção de sobreposição difusa utilizando a função w' é especificado formalmente por meio da função *manut-sobreposicao-difusa* mostrada na Figura 5.13, linha 46. A partir de informações sobre regras *rs* : *Regras* e da atual representação *rep* : *Representacao*-1, a função *manut-sobreposicao-difusa* (linha 46.0) gera como saída uma nova representação *rep'* : *Representacao*-1 atualizada com as informações presentes em *rs*. Isto é feito da seguinte forma. Primeiro, utiliza-se w' para atualizar a medida de incerteza associada a uma regra r (linha 46.3), pertencente ao mapeamento *rep.mc* de entrada (linhas 46.2, 46.4); regra que foi utilizada de forma correta durante a resolução de problemas (linha 46.5). Após, utiliza-se w' para realizar o mesmo processo só que agora com relação às regras utilizadas de forma errada, valor -1 (linhas 46.6, 46.8), pertencentes ao mapeamento mc' (linhas 46.7) resultante da atualização anterior (linha 46.3). Terceiro, utiliza-se w' para atualizar a medida de incerteza da regras, valor -2 (linhas 46.9,46.11), pertencentes ao mapeamento mc'' (linha 46.10) resultante da atualização anterior (linha 46.6). O mapeamento mc''' resultante da última atualização (linha 46.9) é utilizada para criar uma nova *rep'* : *Representacao*-1 que será o valor

de saída da função (linha 46.12). Notemos que *rep'* preserva a antiga seqüência de aprendizagem *rep.md*. Ou seja, a função *manut-sobreposicao-difusa* lida apenas com as informações provindas da análise de passos de resolução (seção 5.4.2). A atualização da seqüência de aprendizagem a partir de informações provindas da aquisição direta (seção 5.4.3) será tratada a seguir.

Por fim, a função *manut-sobreposicao-difusa* é dita modelar a função *manutencao* especificada na Figura 5.5.

Proposição 5.5.1.1

(i) $\forall\, rs : Regras, rep_1 : Representacao\text{-}1 \cdot$
pre-$manutencao(retr\text{-}Ainfo(rs), retr\text{-}Representacao(rep_1)) \Rightarrow$
pre-$manut\text{-}sobreposicao\text{-}difusa(rs, rep_1)$

(ii) $\forall\, rs : Regras, rep_1 : Representacao\text{-}1, rep_1' : Representacao\text{-}1 \cdot$
pre-$manutencao(retr\text{-}Ainfo(rs), retr\text{-}Representacao(rep_1)) \wedge$
post-$manut\text{-}sobreposicao\text{-}difusa(rs, rep_1, rep_1') \Rightarrow$
post-$aquisicao(retr\text{-}Ainfo(rs), retr\text{-}Representacao(rep_1), retr\text{-}Representacao(rep_1'))$

A demonstração da proposição 5.5.1.1 encontra-se no Apêndice B.

5.5.2 Valores Iniciais

A função *manut-sobreposicao-difusa* lida apenas com as informações provindas da análise de passos de resolução (seção 5.4.2). O processo de manutenção, no entanto, deve lidar com as informações provindas da aquisição direta discutida na seção 5.4.3. As informações obtidas pela aquisição direta são unidades pedagógicas a serem adicionadas à atual seqüência de aprendizagem *md* de um objeto composto *rep* : *Representacao*-1 (Figura 5.7). Uma conseqüência imediata de adicionar uma unidade pedagógica à seqüência de aprendizagem *rep.md* é que o domínio de sobreposição dom *rep.mc* ganha novas unidades de conhecimentos (ver invariante de *Representacao*-1, Figura 5.7, linha 24.3). E, estas novas unidades de conhecimento irão requerer valores iniciais com os quais se associar. Valores que representam o que o SMA estima ser o conhecimento inicial do aluno em relação a cada unidade de conhecimento da nova unidade pedagógica.

A função *manutencao*-1 especificada na Figura 5.14 tem por objetivo unir o método de manutenção de sobreposição difusa especificado através da função

values

48.0 $wi = 0.1$

functions

49.0 $manutencao\text{-}1\,(aif : Ainfo\text{-}1, rep : Representacao\text{-}1)\ rep' : Representacao\text{-}1$

.1 pre is-$Direta\,(aif)\ \Rightarrow\ \exists\, upx : UndPedagogica \cdot$

.2 $aif = \text{mk-}Direta\,(upx)\ \wedge$

.3 let $up' = rep.md.up \curvearrowright [upx]$ in

.4 inv-$SequenciaUP\,(\text{mk-}SequenciaUP\,(up', rep.md.cur))$

.5 post (is-$Direta\,(aif) \wedge \exists\, upx : UndPedagogica \cdot$

.6 $aif = \text{mk-}Direta\,(upx)\ \wedge$

.7 let $md' = \text{mk-}SequenciaUP\,(rep.md.up \curvearrowright [upx], rep.md.cur)$ in

.8 let $mc' = rep.mc \dagger \{r \mapsto wi \mid r \in upx\}$ in

.9 $rep' = \text{mk-}Representacao\text{-}1\,(md', mc'))\ \vee$

.10 $(\text{is-}Regras\,(aif)\ \wedge$

.11 post-$manut\text{-}sobreposicao\text{-}difusa\,(aif, rep, rep'))$

Figura 5.14: Especificação VDM-SL: Valores Iniciais.

manut-sobreposicao-difusa com um tratamento adequado das informações provindas da aquisição direta. Vejamos como funciona.

As entradas são um objeto *aif* : *Ainfo*-1 (Figura 5.12, linha 43) e um objeto *rep* : *Representacao*-1 (Figura 5.7, linha 24); a saída é um novo objeto *rep'* : *Representacao*-1 a ser criada por *manutencao*-1 de duas formas distintas. Primeira: se o objeto *aif* contém informações provindas da aquisição por análise de passos (linha 49.10) então o novo objeto *rep'* deve ser criada pela função *manut-sobreposicao-difusa* (linha 49.11). Segunda: se o objeto *aif* contém informações obtidas por aquisição direta (linha 49.5), ou seja, uma unidade pedagógica *upx* (linha 49.6), então *upx* deve ser concatenada a atual seqüência de aprendizagem *rep.md.up* dando origem a uma nova seqüência (linha 49.7). Com uma nova seqüência de aprendizagem o domínio do mapeamento *rep.mc* ganha novas unidades de conhecimento. Estas são associadas a um valor real fixo $wi = 0.1$ (linha 49.8). Este valor inicial pode e deve ser ajustado a partir de testes com implementações particulares de nossa especificação. Por fim, a nova seqüência de aprendizagem e o mapeamento *rep.mc* com novas unidades de conhecimento são usados para criar o objeto *rep'* : *Representacao*-1 resultado da função *manutencao*-1 (linha 49.9).

Da mesma forma que a função *aquisicao*-1 não modela a função *aquisicao* (seção

5.4.3) , a função *manutencao*-1 não modela o comportamento especificado pela função *manutecao* mostrada na Figura 5.5. O problema, como antes, reside nas precondições das duas funções. Notemos a precondição de *manutencao*-1. Esta é literalmente igual à que foi definida para a função *aquisicao*-1 (Figura 5.11, linhas 44.1 a 44.4). Quer dizer, a função *manutencao*-1, antes de qualquer coisa deve verificar se a unidade pedagógica provida da aquisição direta (linhas 49.1 e 49.2) é tal que concatenada com a atual seqüência de aprendizagem (linha 49.3) forme uma seqüencia de aprendizagem válida (linha 49.4).

A precondição de *manutencao*, por outro lado, é sempre verdadeira. Donde não podemos mostrar que : $\forall$-,- · pre-*manutencao*(*retr*-, *retr*-) $\Rightarrow$ pre-*manutencao*-1(-,-); a primeira condição necessária para que *manutencao*-1 modele *manutencao*. Neste caso, as mesmas considerações que fizemos ao final da seção 5.4.3 sobre as funções *aquisicao* e *aquisicao*-1 valem para as funções *manutencao* e *manutencao*-1.

5.6 Juntando as partes

Após reificar tipos de dados e modelar funções da especificação inicial VDM-SL do SMA que propomos, nesta última seção reunimos nosso trabalho em um documento final. Discutimos ainda como este documento pode ser utilizado como ponto de partida para uma implementação do SMA proposto ao longo deste capítulo.

5.6.1 Documento Final

O documento final é uma nova versão da especificação abstrata geral apresentada na seção 5.2.2. Uma nova versão onde a forma do Modelo do Aprendiz e das operações de aquisição, manutenção e inclusão permanece a mesma mas onde os nomes mudam. A mudança de nomes reflete o nosso trabalho ao longo deste capítulo em especificar com mais detalhes tipos de dados, operações e funções componentes da especificação abstrata inicialmente proposta. Detalhes surgidos principalmente a partir de considerações quanto à arquitetura e funcionalidade do Sistema Tutor dos agentes MATHEMA.

A forma permanece a mesma, mudam nomes, e o comportamento geral do SMA especificado no início do capítulo de certo modo é preservado. Mostramos ao longo deste capítulo que os tipos de dados *Representacao*-1, *Observacao*-1 e *Ainfo*-1 que agora aparecem no documento final (Figura 5.15) são, respectivamente, versões mais concretas dos tipos *Representacao*, *Observacao* e *Ainfo* que aparecem no documento inicial (seção 5.2). Mostramos ainda que as funções *manutencao*-1 e *aquisicao*-1 que também aparecem no documento final contêm o comportamento especificado pelas

types

50.0 $MA\text{-}1 = Anome \xrightarrow{m} Representacao\text{-}1$

51.0 state $sma\text{-}1$ of
.1 $cur : Curriculum$
.2 $ma : MA\text{-}1$
.3 inv mk-$sma\text{-}1\,(cur, ma) \triangleq \forall\, a \in \mathsf{dom}\ ma \cdot$
.4 $ma\,(a).md.cur = cur$
.5 end

functions

52.0 $retr\text{-}MA : MA\text{-}1 \to MA$
.1 $retr\text{-}MA\,(ma) \triangleq$
.2 $\{a \mapsto retr\text{-}Representacao\,(ma\,(a)) \mid a \in \mathsf{dom}\ ma\}$

operations

53.0 $AQUISICAO\text{-}1\,(obs : Observacao\text{-}1, a : Anome)\ aif : Ainfo\text{-}1$
.1 ext rd $ma : MA\text{-}1$
.2 pre $a \in \mathsf{dom}\ ma \wedge$
.3 pre-$aquisicao\text{-}1\,(obs, ma\,(a))$
.4 post post-$aquisicao\text{-}1\,(obs, ma\,(a), aif) \wedge$
.5 pre-$manutencao\text{-}1\,(aif, ma\,(a))$;

54.0 $MANUTENCAO\text{-}1\,(aif : Ainfo\text{-}1, a : Anome)$
.1 ext wr $ma : MA\text{-}1$
.2 pre $a \in \mathsf{dom}\ ma \wedge$
.3 $\exists\, obs : Observacao\text{-}1 \cdot$
.4 pre-$aquisicao\text{-}1\,(obs, ma\,(a)) \wedge$
.5 post-$aquisicao\text{-}1\,(obs, ma\,(a), aif) \wedge$
.6 pre-$manutencao\text{-}1\,(aif, ma\,(a))$
.7 post $\exists\, rep : Representacao\text{-}1 \cdot$
.8 post-$manutencao\text{-}1\,(aif, \overleftarrow{ma}\,(a), rep) \wedge$
.9 $ma\,(a) = rep$;

55.0 $INCLUSAO\text{-}1\,(a : Anome)$
.1 ext wr $ma : MA\text{-}1$
.2 rd $cur : Curriculum$
.3 pre $a \notin \mathsf{dom}\ ma$
.4 post let $md =$ mk-$SequenciaUP\,([], cur)$ in
.5 let $mc = \{\mapsto\}$ in
.6 $ma\,(a) =$ mk-$Representacao\text{-}1\,(md, mc)$

Figura 5.15: Especificação VDM-SL: Documento Final.

funções *manutencao* e *aquisicao* presentes no documento inicial (ver comentários ao final das seções 5.4 e 5.5). Destes fatos é de se esperar que as operações *AQUISICAO*-1 (linha 53) e *MANUTENCAO*-1 (linha 54), por suas formas, também contenham o comportamento antes especificado pelas operações *AQUISICAO* e *MANUTENCAO* (Figura 5.2, linhas 4 e 5).

Devemos, no entanto, observar o seguinte detalhe. Antes, as operações *AQUISICAO* e *MANUTENCAO* tinham como estado um mapeamento *MA* (linhas 2.0, 3.1, 4.1 e 5.1). Agora, as operações *AQUISICAO*-1 e *MANUTENCAO*-1, são especificadas (linhas 51.2, 53.1, 54.1) tendo como estado um mapeamento *MA*-1 (linha 50.0). O que significa que temos que mostrar que *MA*-1 reifica *MA* para corroborar a hipótese levantada no fim do parágrafo anterior. Isto é feito demonstrado duas propriedades da função *retr-MA* mostrada na linha 52:

Proposição 5.6.1.1

(i) A função $retr\text{-}MA : MA\text{-}1 \rightarrow MA$ é total

(ii) $\forall\, ma : MA \cdot \exists\, ma_1 : MA\text{-}1 \cdot retr\text{-}MA(ma_1) = ma$

A demonstração da proposição 5.6.1.1 é um exercício fácil uma vez que já demonstramos (seção 5.3) uma proposição similar para a função *retr-Representacao* que aparece na definição de *retr-MA* (linha 52.2). No Apêndice B encontra-se a demonstração.

A operação *INCLUSAO*-1 (linha 55), por sua vez, tem forma ligeiramente diferente que sua correspondente *INCLUSAO* que aparece no documento inicial (Figura 5.2, linha 6). Dizer que *INCLUSAO*-1 preserva o comportamento especificado por *INCLUSAO* já não é um fato tão evidente. A rigor, temos que mostrar através de uma proposição que *INCLUSAO*-1 modela *INCLUSAO*, no sentido empregado ao termo no fim da seção 5.4.2. Não o fazemos por termos ciência de que a operação de inclusão não é tão importante em nossa apresentação. Mesmo assim, observemos que *INCLUSAO* estabelece como pós-condição que deve existir uma representação (linha 6.3) tal que esta representação é a representação inicial associada a um aprendiz de nome *a* (linha 6.4). A operação *INCLUSAO*-1 modela este fato construindo uma representação (linha 55.6) tendo como referência um *Curriculum* (linha 55.2). Uma representação inicial onde tanto a seqüência de aprendizagem *md* (linha 55.4) quanto o mapeamento *mc* (linha 55.5) são vazios; é tarefa da aquisição e manutenção atualizarem estes componentes.

A propósito, o *Curriculum* usado como referência na construção da representação pela operação *INCLUSAO*-1 passa a ser parte do estado global da especificação (linha 51). Notemos que, como só há um *Curriculum* no Sistema Tutor de cada agente MATHEMA, uma propriedade comum a todo estado no qual pode se encontrar o SMA de um agente tutor é que o *Curriculum* em relação ao qual um aprendiz esteja sendo modelado seja o mesmo para todos aprendizes representados no Modelo do Aprendiz (linhas 51.3 e 51.4). O que deve mudar são as seqüências de aprendizagem e a estimativa de quanto cada aprendiz já assimilou do conteúdo apresentado por meio de situações de resolução de problemas.

5.6.2 Decomposicação de Operacões

Usando a terminologia empregada na seção 5.1 podemos chamar o documento final acima de *especificação concreta geral* de um SMA para o Sistema Tutor dos agentes MATHEMA. Como dissemos uma especificação concreta é o resultado de um processo de reificação de dados a partir de uma especificação abstrata. E como o nome que demos à especificação abstrata foi especificação abstrata geral então o termo acima justifica-se.

Uma especificação concreta, conforme dissemos na seção 5.1, é o ponto de partida para um processo de decomposição de operações [Jon90, cap.10]. Através do processo de decomposição de operações uma especificação concreta é transformada em um programa escrito em uma linguagem de programação específica que implementa a especificação. O processo de decomposição de operações assemelha-se ao processo de reificação de dados em um ponto: a cada passo de decomposições algumas provas relacionando especificação e código de implementação devem ser feitas. Estas provas são feitas para garantir que o programa final realmente implemente o comportamento especificados pelas operações.

Caso queiramos ser rigorosos e manter total coerência com o método de desenvolvimento sistemático VDM a especificação do SMA que propomos neste capítulo deve ser submetida ao processo de decomposição de operações. Devemos também dizer que alguns dos tipos de dados presentes na especificação concreta que propomos podem ainda se mostrar inadequados ao processo de decomposição de operações. Neste caso mais uma etapa de reificação de dados deve ser conduzida.

Além do processo de decomposição de operação, a especificação do SMA que propomos é útil como ponto de referência para uma implementação seguindo um método informal de programação. Como dissemos de início uma especificação formal nos ajuda

a ter uma compreensão mais clara do funcionamento do sistema a ser implementado.

Quer prossigamos com o método de desenvolvimento sistemático VDM ou partamos para uma implementação direta, várias linguagens podem ser empregadas para implementar o SMA especificado. Uma vez que a especificação de operações e funções é feita através de pré e pós-condições de forma declarativa (mostramos qual deve ser o comportamento do sistema, não mostramos como produzi-lo), linguagens declarativas tais como PROLOG talvez sejam uma opção mais natural. Uma outra opção é a própria linguagem VDM-SL.

VDM-SL possui muito mais recursos do que os que utilizamos neste capítulo. Um deles é especificar de forma direta operações e funções através de comandos e/ou expressões. Apesar de não ser propriamente uma linguagem de programação, VDM-SL possui comandos de atribuição, comandos condicionais, laços, etc., com semântica denotacional bem definida e já estabelecida como um padrão ISO [LHB+96, ABH+95]. Tendo por base este padrão existem algumas ferramentas de desenvolvimento de *software* que implementam interpretadores para os comandos e expressões VDM-SL. Este é o caso da ferramenta IFAD VDM-SL Toolbox[8] que utilizamos na elaboração da especificação apresentada neste capítulo. Com esta ferramenta e tendo transformado a especificação que propomos em código VDM-SL podemos simular o funcionamento do SMA proposto. Mais ainda, podemos também criar protótipos com pouco trabalho pois essa ferramenta é capaz de traduzir código VDM-SL em código executável C++.

[8] Ver nota de rodapé, página 80.

Capítulo 6

O Uso do Modelo do Aprendiz

6.1 Introdução

A especificação do sistema de modelagem do aprendiz apresentado no capítulo anterior não está completa e nem tem sentido se não apresentarmos como o Modelo do Aprendiz adquirido, representado e mantido pode ser utilizado pelo Sistema Tutor dos agentes MATHEMA. Este é o tema deste capítulo.

No Sistema Tutor dos agentes MATHEMA, uma das funções pedagógicas mais importantes é a escolha dos problemas a serem passados ao aluno; ver seção 4.2. O uso básico das informações sobre o estado de conhecimento do aprendiz em relação ao *Curriculum*, conforme dito no capítulo anterior, deve ser ajudar na escolha de problemas. Neste capítulo mostramos como definir alguns conceitos relativos à escolha de problemas usando as informações sobre o estado de conhecimento dos aprendizes. Os conceitos principais são as *sessões de resolução de problemas* e as *funções de seleção de problemas.*

Em torno destes conceitos organizamos o capítulo em três seções principais. Na seção 6.2 apresentamos a noção de sessão de resolução de problemas. Em particular, mostramos como particionar *adequadamente* o Domínio de Problemas presente no Sistema Tutor dos agentes MATHEMA utilizando as informações sobre a *seqüência de aprendizagem* sendo seguida por um aprendiz. Onde *adequadamente* significa que os problemas a serem passados a um aprendiz em uma dada sessão de resolução, quando provindos de um conjunto de problemas resultante do particionamento, respeitam a seqüência de aprendizagem sendo seguida pelo aprendiz e não correm o risco de se repetir. Na seção 6.3 passamos então a considerar como selecionar um problema, o mais adequado, tendo em mãos um conjunto de problemas adequados; falamos da noção de função de seleção de problemas. Em particular, mostramos a especificação de uma

função de seleção de problemas utilizando a *sobreposição difusa*, parte do Modelo do Aprendiz especificado no capítulo anterior. Por fim, na seção 6.4, unimos os resultados apresentados nas duas seções anteriores em uma especificação para todo o processo de seleção de problemas proposto.

6.2 Sessões de Resolução de Problemas

A representação por sobreposição com incerteza proposta no capítulo anterior (Figura 5.7) é composta por uma seqüência de aprendizagem *md* (linha 24.0) e por um mapeamento *mc* entre unidades de conhecimento e números reais entre 0 e 1 (linha 24.1). A seqüência de aprendizagem *md* é o elemento chave na definição de sessões de resolução de problemas.

Uma *sessão de resolução de problemas* é uma sucessão de problemas que o Sistema Tutor dos agentes MATHEMA propõe a um aprendiz em uma dada unidade pedagógica ao longo de uma seqüência de aprendizagem. Uma sucessão onde não deve haver problemas repetidos e que deve envolver apenas problemas associados a uma dada unidade pedagógica ao longo de uma seqüência de aprendizagem.

Unidades de Problemas

Os problemas associados a uma dada unidade pedagógica n ao longo de uma seqüência de aprendizagem *md* (escrita md.up(n)) são problemas pertencentes ao Domínio de Problemas *dpr* do Sistema Tutor dos agentes MATHEMA (Figura 5.6, linha 22.1), e que são determinados por uma função *unidade-problema* especificada na Figura 6.1, linha 56. Ao conjunto de tais problemas denominamos *unidade de problemas*: linha 57.

A definição das unidades de problemas é tal que a cada unidade pedagógica $md.up(n)$ de uma seqüência de aprendizagem corresponde um subconjunto de problemas do Domínio de Problemas (linhas 56.0, 56.1 e 56.2) tal que os problemas p do subconjunto envolvem unidades de conhecimento $p.ucs$ da unidade pedagógica $md.up(n)$ (linha 56.3) juntamente com possíveis unidades de conhecimento de outras unidades pedagógicas que estejam antes na seqüência (linhas 56.4 e 56.5). Isto nos garante que os problemas que o Sistema Tutor dos agentes MATHEMA pode propor ao aprendiz tendo por base unidades de conhecimentos estão de acordo com (respeitam) a seqüência de aprendizagem sendo seguida pelo aluno. Um exemplo ilustra o que queremos dizer.

Consideremos o *Curriculum* para o subdomínio Estudo de Retas no contexto da Geometria Analítica restrito ao (profundidade) Plano Cartesiano conforme apresentado na seção 2.4.2, Figura 2.6. Duas possíveis seqüências de aprendizagem $ma(Luciano).md$

```
functions

56.0  unidade-problema (n : ℕ, md : SequenciaUP, dpr : Problema-set) ps : Problema-set
 .1   pre  n ≤ len md.up
 .2   post ps = {p | p ∈ dpr ·
 .3              p.ucs ∩ md.up (n) ≠ {} ∧
 .4              let crp = ⋃ md.cur.prt in
 .5              p.ucs ∩ crp ⊆ ⋃elems md.up (1, ... , n)}

types

57.0  UndProblema = Problema-set
 .1   inv ps ≜
 .2       ∃ n : ℕ, md : SequenciaUP, dpr : Problema-set ·
 .3           post-unidade-problema (n, md, dpr, ps)
```

Figura 6.1: Especificação VDM-SL: Unidades de Problemas.

e $ma(Jose).md$ sobre este *Curriculum* são mostradas na Figura 6.2. Consideremos agora um problema p pertencente ao Domínio de Problemas do Sistema Tutor tendo o seguinte enunciado:

Problema: Seja r a reta que passa pelo ponto $A(0,0)$ e é paralela à reta $s: y = 2x+1$. Qual a distância do ponto $(-1,-1)$ à reta r ?

Esse problema envolve unidades de conhecimento pertencentes às unidades pedagógicas Coeficiente Angular, Equação da Reta, Posições Relativas entre Retas e Distâncias entre Retas.

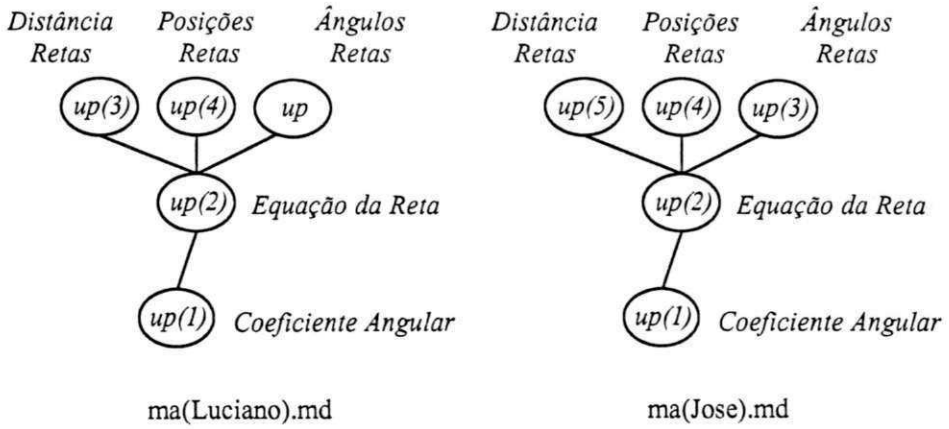


Figura 6.2: Seqüências de aprendizagem.

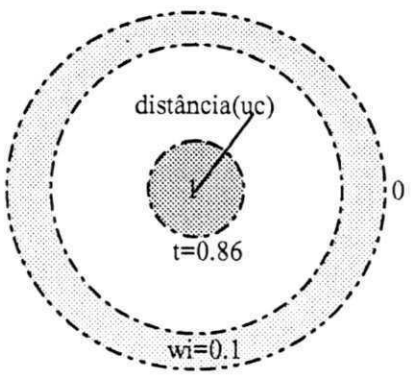


Figura 6.4: Sobreposição Difusa e Distância.

Os parâmetros wi e t dividem assim as unidades de conhecimento em uma sobreposição difusa mc em três intervalos; Figura 6.4. No primeiro, $\{uc \mid uc \in \mathsf{dom}\ mc \cdot 0 \leq mc(uc) \leq wi\}$, encontram-se as unidades de conhecimento que o SMA "está certo" de que o aluno não sabe. No segundo, $\{uc \mid uc \in \mathsf{dom}\ mc \cdot wi < mc(uc) < t\}$, encontram-se as unidades de conhecimento que o SMA "não tem certeza" se já foram assimiladas pelo aluno ou não. E, no terceiro, $\{uc \mid uc \in \mathsf{dom}\ mc \cdot t \leq mc(uc) \leq 1\}$, encontram-se as unidades de conhecimento que o SMA "está certo" de que o aluno já assimilou.

O objetivo do Sistema Tutor dos agentes MATHEMA ao passar problemas para os aprendizes deve ser então fazer com que as unidades de conhecimento, inicialmente no primeiro intervalo, passem pelo segundo intervalo e finalmente atinjam o terceiro intervalo. Nesse processo os valores associados a cada unidade de conhecimento podem ser vistos como *distâncias*; distâncias que separam cada unidade de conhecimento de um valor onde o aprendiz é dito, com 100% de certeza, ter aprendido a unidade de conhecimento.

Distâncias

Inicialmente uma unidade de conhecimento é colocada em uma sobreposição difusa a uma distância de $(1 - 0.1)$. Quando uma unidade de conhecimento atinge o limiar de assimilação sua distância é $(1 - 0.86)$. Uma unidade de conhecimento em relação a qual não restam dúvidas quanto ao fato de que o aprendiz não a sabe tem distância igual a $(1 - 0)$. Uma unidade de conhecimento em relação a qual o SMA garante com 100% de certeza de que ela foi assimilada pelo aprendiz tem distância igual a $(1 - 1)$. De forma geral para qualquer unidade de conhecimento $uc \in \mathsf{dom}\ mc$ sua distância é $d = 1 - mc(uc)$.

A noção de distância é especificada formalmente na Figura 6.5.

functions

60.0 $distancia : UndConhecimento \times UndConhecimento \xrightarrow{m} LI \rightarrow \mathbb{R}$

.1 $distancia\,(uc, mc) \triangleq$

.2 $\quad 1 - mc\,(uc)$

.3 pre $uc \in \text{dom } mc$

Figura 6.5: Especificação VDM-SL: Distância de uma Unidade de Conhecimento.

Uma extensão útil na noção de distância é a noção de *distância média.* A distância média é a noção de distância aplicada a subconjuntos de unidades de conhecimento em uma sobreposição difusa. Ou seja, um valor que nos mostra o quanto falta para que o SMA ateste com 100% de confiança que um conjunto de unidades de conhecimento foi assimilado pelo aprendiz.

A distância média $\overline{d}$ é calculada como uma média aritmética dos valores associados a cada unidade de conhecimento uc_i de um conjunto ucs, card $ucs = n$, em uma sobreposição difusa mc. Ou seja,

$$\begin{aligned}\overline{d} &= \frac{(1 - mc(uc_1)) + ... + (1 - mc(uc_n))}{n} \\ &= \frac{n - \sum_{i=1}^{n} mc(uc_i)}{n} \\ &= 1 - \sum_{i=1}^{n} \frac{mc(uc_i)}{n}\end{aligned}$$

Esta última expressão é utilizada na especificação de uma função que calcula a distância média de um conjunto de unidades de conhecimento ucs com relação a uma sobreposição difusa mc mostrada na Figura 6.6. Ver linha 62.3.

Os detalhes da especificação da função de cálculo da distância média são pouco importantes. O mais importante é que a partir da definição de distância média podemos especificar uma função de seleção de problemas; uma função de seleção cujo principal meio de decidir que problema passar ao aprendiz é um processo de encurtar distâncias médias escolhendo problemas que envolvam unidades de conhecimento mais distantes e, de preferência, os que envolvam o maior número possível dessas unidades de conhecimento.

A função *selecao-problema*

A função de seleção de problemas que agora passamos a discutir (Figura 6.7, linha 63) tem como entradas uma sobreposição difusa mc, parte integrante da representação por sobreposição com incerteza do Modelo do Aprendiz (Figura 5.7, linha 24.1), e um

functions

61.0 $somatorio : UndConhecimento \xrightarrow{m} LI \rightarrow \mathbb{R}$

.1 $somatorio\,(mc) \triangleq$

.2 let $uc \in$ dom mc in

.3 if card dom $mc = 1$

.4 then $mc\,(uc)$

.5 else $mc\,(uc) + somatorio\,(\{uc\} \lhd mc)$

.6 pre dom $mc \neq \{\}$;

62.0 $Distancia : UndConhecimento\text{-set} \times UndConhecimento \xrightarrow{m} LI \rightarrow \mathbb{R}$

.1 $Distancia\,(ucs, mc) \triangleq$

.2 let $m = ucs \lhd mc$ in

.3 $1 - somatorio\,(m)/$card dom m

.4 pre $ucs \subseteq$ dom mc

Figura 6.6: Especificação VDM-SL: Distância de um Conjunto de Unidades de Conhecimento.

conjunto de problemas ps. A saída é um problema selecionado a partir de ps utilizando a seguinte estratégia:

1. selecionar dentre um conjunto de problemas de entrada aqueles cujo conjunto de unidades de conhecimento envolvido tenham as maiores distâncias médias; (linhas 63.4 a 63.7);

2. dentre os problemas selecionados na etapa posterior escolher aqueles que envolvem o maior número possível de unidades de conhecimento (linhas 63.8 a 63.10);

3. por fim, dentre os problemas escolhidos no passo anterior selecionar aleatoriamente um problema (linha 63.11).

Uma precondição da função de seleção é que o conjunto de problemas de entrada seja não vazio (linha 63.1); quer dizer, que haja problemas dentre os quais um deve ser selecinado. Outra precondição é que todo problema do conjunto de problemas de entrada envolva um conjunto de unidades pedagógicas contido no domínio da sobreposição difusa de entrada (linhas 63.2 e 63.3); quer dizer, que os problemas dentre os quais um vai ser selecionado possam ser utilizados para encurtar distâncias médias com relação à sobreposição difusa de entrada.

```
functions

63.0  selecao-problema (mc : UndConhecimento -m-> LI, ps : Problema-set) p : Problema
 .1   pre  ps ≠ {} ∧
 .2       ∀ p ∈ ps ·
 .3           p.ucs ⊆ dom mc
 .4   post let MaisDistante = {p | p ∈ ps ·
 .5                                ¬ ∃ p' ∈ ps ·
 .6                                    Distancia (p'.ucs, mc) >
 .7                                    Distancia (p.ucs, mc)} in
 .8        let MaisUC = {p | p ∈ MaisDistante ·
 .9                           ¬ ∃ p' ∈ MaisDistante ·
 .10                              card p'.ucs > card p.ucs} in
 .11       p ∈ MaisUC
```

Figura 6.7: Especificação VDM-SL: Função de Seleção de Problema.

A segunda precondição nos mostra ainda a idéia central da função de seleção proposta: fazer com que o aprendiz a cada problema resolvido, possa aprender conceitos novos. Quer dizer, unidades de conhecimento que o SMA estima ainda não fazerem plenamente parte do estado cognitivo do aluno. Além de apresentar conceitos novos a cada problema a função de seleção tenta envolver em uma mesma situação de resolução de problemas o maior número possível de conceitos. Isto para tentar fazer com que o aluno tenha contato com o maior número possível de conhecimento em menos tempo/problemas possível.

Esta última característica da função de seleção que propomos talvez seja questionável. Talvez uma estratégia de envolver o mínimo possível de novas unidades de conhecimento em cada passo de aprendizagem seja pedagogicamente mais aconselhável. Não o sabemos. No entanto, como arrolaremos na conclusão, Capítulo 7, esperamos em trabalhos futuros estudar uma solução pedagogicamente mais aconselhável a esta situação.

6.4 Juntando as Partes

Tendo mostrado como particionar o Domínio de Problemas presente no Sistema Tutor dos agentes MATHEMA utilizando uma seqüência de aprendizagem *md* e, apresentado uma função de seleção de problemas utilizando uma sobreposição difusa *mc*, nos resta agora juntar estes dois resultados em uma especificação para todo o processo de seleção

de problemas. Esta especificação é mostrada na Figura 6.8.

A idéia é definir operações para o estabelecimento de sessões de resolução de problemas. Imaginamos duas operações principais, *INICIAR-SESSAO* (linha 65) e *PROXIMO-PROBLEMA* (linha 66), cujo estado (linha 64) é formado por um Modelo do Aprendiz (linha 64.1), um Domínio de Problemas (linha 64.2) e uma memória auxiliar (linha 64.3) que armazena os problemas que ainda não foram passados a aprendizes em sessões de resolução de problemas.

A operação *INICIAR-SESSAO*, como o nome pode sugerir, tem como propósito iniciar uma sessão de resolução de problemas para um aluno de nome a (linha 65.0) previamente representado no Modelo do Aprendiz (linha 65.4). O que é feito da seguinte forma. Primeiro, verificar qual a atual unidade pedagógica da seqüência de aprendizagem do aluno de nome a (linha 65.9). Depois, determinar a unidade de problemas para esta unidade pedagógica da seqüência de aprendizagem (linhas 65.10 e 65.11). Por fim, armazenar a unidade de problemas como os problemas a serem passados ao aprendiz na atual sessão de resolução de problemas (linha 65.12).

Notemos que a operação *INICIAR-SESSAO* deve ser realizada uma única vez depois que um nova unidade pedagógica é concatenada a uma seqüência de aprendizagem. Senão problemas repetidos serão passados ao aprendiz. Este fato é capturado pelas linhas 65.5 a 65.8 da precondição da operação *INICIAR-SESSAO*.

A operação *PROXIMO-PROBLEMA* tem como objetivo decidir qual o próximo problema a ser passado a um aprendiz de nome a na atual sessão de resolução de problemas (linha 66.0). O que é feito da seguinte forma. Antes de mais nada, verificar se os problemas que ainda não foram passados, ou seja, se os problemas do conjunto $ps(a)$, ainda correspondem à atual unidade pedagógica da seqüência de aprendizagem (linhas 66.5 a 66.8). Esta precondição pode não ser verdadeira caso a operação *INICIAR-SESSAO* não tenha sido executada depois de uma nova unidade pedagógica ter sido concatenada à seqüência de aprendizagem. Notemos que a cada nova unidade pedagógica deve corresponder uma sessão de resolução de problemas.

Dado que a precondição acima seja verdadeira o próximo passo é usar a função de seleção discutida na seção anterior para escolher um problema do conjunto $ps(a)$ (linha 66.9). Por fim, tendo escolhido um problema, este deve ser retirado do conjunto $ps(a)$ (linha 66.10). Em uma situação extrema todos os problemas de uma unidade de probemas podem ser passados ao aprendiz. Isto faz com que o conjunto $ps(a)$ torne-se vazio. Neste caso a operação *SELECAO-PROBLEMA* não pode mais ser executada. Este fato é capturado pelas linhas 66.3 e 66.4 da precondição da operação *SELECAO-PROBLEMA*.

tema Tutor dos agentes MATHEMA de modo a estabelecer sessões de resolução e finalizamos apresentando uma função de seleção de problemas que propõe problemas a um aprendiz levando em consideração seu nível de conhecimento.

7.2 Resultados atingidos

Os resultados que atingimos com nossa pesquisa e que foram relatados nesta dissertação podem ser dispostos em duas categorias segundo nossos objetivos iniciais:

(1) Direções gerais:

- Chegamos à conclusão que um Modelo do Aprendiz para os agentes MATHEMA deve ser individual e persistente (seção 4.3.1) e que pelo menos três tipos de informações mostram-se úteis. Informações sobre o estado de conhecimento de um aprendiz em relação ao domínio de aprendizagem, seus planos e estratégias ao resolver problemas e informações gerais sobre suas preferências (seções 4.3.2 e 4.3.3).
- Identificamos uma coleção de idéias e técnicas computacionais que podem ser empregadas no projeto e implementação de Sistemas de Modelagem do Aprendiz para os agentes MATHEMA (Figura 4.4).

(2) Sistema de Modelagem do Aprendiz:

- Especificamos um Sistema de Modelagem do Aprendiz para os agentes MATHEMA com as seguintes características principais:
  - Representação : sobreposição com representação de incerteza; (Figura 5.7)
  - Aquisição : análise dos passos de resolução; (Figura 5.11)
  - Manutenção : manutenção de sobreposição difusa; (Figura 5.13)
- Mostramos como as informações presentes no Modelo do Aprendiz podem ser utilizadas durante a escolha de problemas de problemas a serem passados aos aprendizes:
  - mostramos como particionar o Domínio de Problemas presente em um agente MATHEMA em unidades de problemas que respeitam a seqüência de aprendizagem sendo seguida por um dado aprendiz; (Figura 6.1)
  - definimos uma função de escolha de problemas a partir das unidades de problemas; (Figura 6.7)

7.3 Trabalhos futuros

Observando os resultados acima vislumbramos muitas linhas de trabalhos futuros. Algumas delas já comentadas de passagem ao longo do texto. A seguir arrolamos as linhas trabalhos que consideramos mais promissoras e que a médio e longo prazo esperamos realizar.

Primeiro, implementar o Sistema de Modelagem do Aprendiz especificado. Nesta implementação planejamos utilizar a própria linguagem VDM-SL conforme comentamos ao fim do Capítulo 5. Com esta implementação intencionamos realizar testes para validar o Sistema de Modelagem do Aprendiz proposto.

Segundo, especificar e implementar uma interface de resolução de problemas que seja adequada ao processo de aquisição por análise de passos de resolução. Ver seção 5.4.2.

Terceiro, tentar novos métodos de representação, aquisição e manutenção para o Sistema de Modelagem do Aprendiz dos agentes MATHEMA. Em especial tentar um método probabilístico para representar e tratar incertezas. Ver seção 5.5.1.

Quarto, definir novas formas de utilizar as informações do modelo do aprendiz. Conforme comentamos ao final da seção 6.3 a função de seleção de problemas que propomos talvez seja questionável de um ponto de vista pedagógico.

Por fim, ampliar nosso estudo sobre modelagem do aprendiz considerando agora a possibilidade de modelagem cooperativa do aprendiz. Conforme dissemos na Introdução nosso trabalho assumiu que cada agente MATHEMA realiza a modelagem do aprendiz independentemente de outros agentes MATHEMA. Este, no entanto, nem sempre é o caso: os agentes MATHEMA são capazes de se comunicar entre si e assim não é difícil imaginar que "trocando idéias" possam chegar a uma visão mais acurada do aprendiz.

Apêndice A

Sintaxe VDM-SL

VDM é um acrônimo de "*Vienna Development Method*": uma coleção de técnicas para a especificação formal e desenvolvimento de sistemas computacionais. Ele consiste em uma linguagem de especificação chamada de VDM-SL; regras para reificação de dados e decomposição de operações que permitem estabelecer ligações entre especificações abstratas de requisitos e especificações detalhadas que surgem nas etapas de projeto e implementação; e uma teoria de prova na qual argumentos rigorosos podem ser construídos sobre as propriedades dos sistemas especificados e correção das decisões de projeto.

A linguagem de especificação VDM-SL suporta a criação de descrições formais de sistemas computacionais através de definições de *tipos de dados*, *funções* e *operações* [Jon90, Gro98].

A.1 Tipos de Dados

De uma forma análoga às linguagens de programação tradicionais é possível definir tipos de dados na linguagem VDM-SL e dar-lhes nomes apropriados. Um definição de tipos de dados tem a seguinte forma:

types

67.0 $Anome = \mathsf{char}^+$

Onde *Anome* é o *tipo* que definimos e char^+ é uma *expressão de definição de tipo.* Expressões de definição de tipos são *tipos primitivos* ou *tipos compostos.*

Tipos Primitivos

Os tipos primitivos principais são *tipos numéricos*, *valores lógicos* e o *tipo caracter*. A simbologia básica associada aos tipos primitivos, seus principais valores e operações que podem ser realizadas com seus valores encontra-se resumida nas tabelas a seguir.

Tipos Numéricos	
$tipo = \mathbb{N}$	$\{0, 1, 2, ...\}$
$tipo = \mathbb{N}_1$	$\{1, 2, ...\}$
$tipo = \mathbb{Z}$	$\{..., -1, 0, 1, ...\}$
$tipo = \mathbb{Q}$	Números racionais
$tipo = \mathbb{R}$	Números reais
$+, -, *, /, <$	operadores (infixos) aritméticos tradicionais
$\uparrow$	(infixo) potência
abs	(prefixo) valor absoluto
mod	(infixo) módulo

Valores Lógicos	
$tipo = \mathbb{B}$	{true, false}
$\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$	operadores (infixos) lógicos tradicionais
$\forall x \in S \cdot E$	quantificador universal
$\exists x \in S \cdot E$	quantificador existencial

Tipo Caracter	
$tipo =$ char	$\{'a', 'b', ..., '0', '1', ...\}$

Tipos Compostos

Há sete tipos compostos: *conjuntos*, *seqüências*, *mapeamentos*, *produtos*, *objetos compostos*, *uniões* e *opcionais*. A simbologia básica associada a definição de tipos compostos, aos principais valores e às principais operações que podem ser realizadas com valores desses tipos está resumida nas tabelas a seguir.

Conjuntos	
$tipo = T$-set	conjunto finito objetos tipo T
$\{t_1, t_2, ..., t_n\}$	enumeração de conjunto
$\{\}$	conjunto vazio
$\{t \mid t \in c \cdot p(x)\}$	especificação de conjunto
$t \in c, t \notin c$	relações de pertinência
$c_1 \subseteq c_2$	inclusão de conjunto (subconjunto de)
$c_1 \subset c_2$	inclusão estrita (subconjunto próprio)
$c_1 \cup c_2$	união de conjuntos
$c_1 \cap c_2$	interseção de conjuntos
$c_1 \setminus c_2$	diferença de conjuntos
$\bigcup\{c_1, ..., c_n\}$	$c_1 \cup c_2 \cup ... \cup c_n$
card c	cardinalidade de um conjunto

Seqüências	
$tipo = T^*$	seqüência finita objetos tipo T
$tipo = T^+$	seqüência finita, não vazia, objetos tipo T
$[t_1, t_2, ..., t_n]$	enumeração de seqüência
$[]$	seqüência vazia
$[s(i) \mid i \in I \cdot p(i)]$	especificação de seqüência
$s(i, ..., j)$	subseqüência
$s_1 \frown s_2$	concatenação de seqüências
conc $\{s_1, ..., s_n\}$	$s_1 \frown s_2 \frown ... \frown s_n$
hd s	cabeça; hd $[a, b, c, d] = a$
tl s	cauda; tl $[a, b, c, d] = [b, c, d]$
len s	tamanho seqüência
inds s	índices; inds $s = \{1, ..., \text{len } s\}$
elems s	elementos; elems $s = [s(i) \mid i \in \text{inds } s]$

Mapeamentos	
$tipo = D \xrightarrow{m} R$	mapeamento finito objetos tipo D em R
$\{d_1 \mapsto r_1, d_2 \mapsto r_2, ..., d_n \mapsto r_n\}$	enumeração de mapeamento
$\{\mapsto\}$	mapeamento vazio
$m(d_i)$	aplicação; $m(d_i) = r_i$
dom m	domínio; dom $m = \{d_1, ..., d_n\}$
rng m	imagem; rng $m = \{r_1, ..., r_n\}$
$c \lhd m$	restrição; $c \lhd m = \{d \mapsto m(d) \mid d \in c \cap \text{dom } m\}$
$c \lhd\!\!\!\!- m$	deleção; $c \lhd\!\!\!\!- m = \{d \mapsto m(d) \mid d \in \text{dom } m \setminus c\}$
$m_1 \dagger m_2$	sobrescrita; $\{a \mapsto 1, b \mapsto 2\} \dagger \{a \mapsto 3\} = \{a \mapsto 3, b \mapsto 2\}$

Mapeamentos são associações entre dois conjuntos de valores; de forma geral podem ser pensados como funções parciais $m : D \to R$.

Produtos Cartesiano	
$tipo = T_1 \times T_2$	produto cartesiano finito entre tipos T_1 e T_2
mk-(t_1, t_2)	construtor de pares ordenados

Produtos cartesianos podem envolver mais de dois tipos. Neste caso os valores mk-$(t_1, ..., t_n)$ são chamados de *tuplas*.

Objetos Compostos	
$tipo :: t_1 : T_1$ $t_2 : T_2$ ... $t_n : T_n$	objeto composto de objetos tipos $T_1, T_2, ..., T_n$
mk-$tipo(t_1, t_2, ..., t_n)$	construtor de objetos compostos
$oc.t_i$	seleção de subobjeto
is-$tipo(oc)$	expressão *é-um*

Objetos compostos são parecidos com produtos cartesianos. A diferença básica entre os dois é que nos objetos compostos os componentes podem ser nomeados e selecionados diretamente. Objetos compostos correspondem aos *registros* ou *estruturas* das linguagem de programação tradicionais.

Uniões e Opcionais	
$tipo = T_1 \mid T_2 \mid ... \mid T_n$	união de objetos tipos $T_1, T_2, ... T_n$
$tipo = [T]$	opcional; $[T] = T \mid$ nil
nil	objeto omitido

Uniões correspondem a idéia de união de conjuntos; i.e., o tipo união contém todos os valores de cada tipo componente. Em uma união de objetos compostos valores são distinguidos por meio de expressões is-$T_i(u)$.

Invariantes

Se um tipo de dados definido como tipos tipos primitivos ou tipos compostos não deve conter certos valores, é possível restringir o conjunto de valores de um tipo por meio de um *invariante*. Ou seja, um predicado que estabelece uma propriedade que deve valer para todo objeto do tipo definido. Uma definição de tipos de dados com invariante tem a seguinte forma:

types

7.0 $Representacao :: md : MD$

.1 $\qquad\qquad\qquad mc : Sentenca \xrightarrow{m} LI$

.2 inv mk-$Representacao\,(md, mc) \triangleq$

.3 $\quad$ dom $mc = md$;

Onde mk-$Representacao(md, mc)$ é um *padrão* que deve casar com os valores pertencentes ao tipo *Representacao*, e dom $mc = md$ é uma expressão lógica, envolvendo alguns ou todos os identificadores no padrão.

A.2 Funções e Operações

Uma especificação formal de um sistema computacional feita por meio da linguagem VDM-SL é um *modelo do sistema*; um modelo constituído de tipos de dados e objetos, que representam entradas, saídas e *estado do sistema*, e de *funções* e *operações*, que representam as operações disponíveis no sistema.

Funções e operações podem ser definidas de forma direta ou especificadas de forma implícita.

Definição Direta

Uma definição direta de função ou operação mostra *como* o resultado é calculado por uma função ou operação.

Uma definição direta de função tem a seguinte forma:

functions

31.0 $retr\text{-}Sentenca : UndConhecimento \to Sentenca$

.1 $retr\text{-}Sentenca\,(\mathsf{mk}\text{-}Regra\,(se, entao)) \triangleq$

.2 $\quad \mathsf{conc}\,[se\,(i) \mid i \in \mathsf{inds}\,se] \frown entao$

Onde $\mathsf{conc}\,[se\,(i) \mid i \in \mathsf{inds}\,se] \frown entao$ é uma expressão que mostra como um objeto tipo *Sentenca* é obtido apartir de outro objeto $\mathsf{mk}\text{-}Regra(se, entao)$.

Especificação Implícita

Uma especificação implícita de função ou operação estabelece, através de pré ou pós-condições, *o que* deve ser computado por uma função ou operação.

Uma especificação implícita de função tem a seguinte forma:

functions

14.0 $aquisicao\,(obs : Observacao, rep : Representacao)\ aif : Ainfo$

.1 pre true

.2 post $aif = \{s \mid s \in rep.md \cdot$

.3 $\qquad \exists p \in \mathsf{dom}\ obs \cdot$

.4 $\qquad\quad obs\,(p) = s\}$

Precondições são expressões lógicas que podem envolver os parâmetros de entrada da função. *Pós-condições* são expressões lógicas que podem envolver tanto valores de entrada como o valor de saída da função. São dessa forma predicados:

$$\mathsf{pre}\text{-}aquisicao : Observacao \times Representacao \to \mathbb{B}$$
$$\mathsf{post}\text{-}aquisicao : Observacao \times Representacao \times Ainfo \to \mathbb{B}$$

Uma especificação implícita de operação tem a seguinte forma:

55.0 $INCLUSAO\text{-}1\,(a : Anome)$

.1 ext wr $ma : MA\text{-}1$

.2 $\quad$ rd $cur : Curriculum$

.3 pre $a \notin \mathsf{dom}\ ma$

```
.4  post let md = mk-SequenciaUP ([], cur) in
.5       let mc = {↦} in
.6       ma (a) = mk-Representacao-1 (md, mc)
```

Por convenção, os nomes de operações são escritos em maiúsculas. A primeira linha da especificação de uma operação é similar àquela para funções. A segunda parte determina que objetos a operação tem acesso externo (ext): nomes de objetos são precedidos por uma indicação de se o acesso é de leitura e escrita (wr) ou apenas de leitura (rd); o nome de cada objeto é seguido por seu tipo.

Pré e pós-condições são como antes com a diferença que passam a envolver os objetos de acesso externo. Em uma pós-condicação o valor de um objeto ext rd $oe : T$ antes da execução da operação é especificado como $\overleftarrow{oe}$; o valor após a execução é oe.

Estado

A coleção de objetos externos acessados por uma operação formam o seu estado. A coleção de todos os objetos externos acessados por todas as operações de uma especificação formam o estado da especificação. Notemos que operações têm estado, funções não.

O estado de uma especificação é definido como um objeto composto e pode ter um invariante:

```
51.0  state sma-1 of
  .1      cur : Curriculum
  .2      ma : MA-1
  .3      inv mk-sma-1 (cur, ma) ≜ ∀ a ∈ dom ma ·
  .4              ma (a).md.cur = cur
  .5  end
```

O papel de um invariante em estados pode ser visualizado considerando-os como pré e pós-condições globais. Um invariante em um estado é uma expressão que pode ser pensada como tendo sido juntada às pré e pós-condições de todas as operações da especificação.

Apêndice B

Demonstrações

B.1 Capítulo 5

Proposição 5.3.2.1

(i) A função *retr-Representacao* : *Representacao-1* $\rightarrow$ *Representacao* é total

(ii) $\forall\, r : \mathit{Representacao} \cdot \exists\, r_1 : \mathit{Representacao\text{-}1} \cdot \mathit{retr\text{-}Representacao}(r_1) = r$

Demonstração

(i) A função *retr-Representacao* (Figura 5.7) atribui a cada r_1 = mk-*Representacao-1*(md, mc) (linha 26.1) um r = mk-*Representacao*(md', mc') (linha 26.4). Onde md' = *retr-MD*(md) e $mc' = \{\mathit{retr\text{-}Sentenca}\,(uc) \mapsto mc\,(uc) \mid uc \in \mathsf{dom}\ mc\}$. Assim *retr-Representacao* é total se as funções *retr-MD* (linha 25) e *retr-Sentenca* (linha 31, Figura 5.8) também são totais. Essas funções, a exemplo de *retr-Representacao*, são funções "retrieve" e mostram como reificar os tipos de dados *MD* e *Sentenca* na especificação da Figura 5.3.

Mostrar que *retr-MD* é total reduz-se em última análise a mostrar que *retr-Sentenca* é total. Notemos que *retr-MD* (linha 25) transforma *md* : *SequenciaUP* (linha 25.1) em um conjunto de sentenças pertencentes a *MD* (linha 25.2); sentenças obtidas pela transformação das unidades de conhecimento pertencentes a algum elemento da seqüência de unidades pedagógicas; transformação via função *retr-Sentenca*. A seguir mostramos como provar que a função *retr-Sentenca* é total.

(ii) Uma argumentação análoga à apresentada acima nos mostra como demonstrar a parte (ii) da proposição 5.3.2.1. Essa argumentação nos leva a reduzir o problema de provar $\forall\, r : \mathit{Representacao} \cdot \exists\, r_1 : \mathit{Representacao\text{-}1} \cdot \mathit{retr\text{-}Representacao}(r_1) = r$

ao problema de provar $\forall\, s : Sentenca \cdot \exists\, uc : UndConhecimento \cdot retr\text{-}Sentenca(uc) = s$. Um problema que discutimos a seguir.

No esquema de prova acima omitimos um detalhe. md' e mc' na linha 26.4 devem ser tais que possamos criar uma representação via mk-$Representacao(md', mc')$. Ou seja, md' e mc' devem ser tais que dom $mc' = md'$, o invariante de *Representacao* (linha 7.3), seja verdadeiro. Isto é satisfeito através do invariante de *Representacao-1* (linhas 24.2 e 24.3). Notemos que por este invariante dom $mc = \bigcup$ elems $md.up$; ou ainda, $retr\text{-}Sentenca(\text{dom } mc) = retr\text{-}Sentenca(\bigcup \text{elems } md.up)$. Ora, $retr\text{-}Sentenca(\text{dom } mc) = \text{dom } mc'$, pela linha 26.3. E, $retr\text{-}Sentenca(\bigcup \text{elems } md.up) = md'$, pela linha 26.2. Assim dom $mc' = md'$ como se queria mostrar.

Proposição 5.3.3.1

(i) A função $retr\text{-}Sentenca : UndConhecimento \rightarrow Sentenca$ é total

(ii) $\forall\, s : Sentenca \cdot \exists\, uc : UndConhecimento \cdot retr\text{-}Sentenca(uc) = s$

Demonstração

(i) A função *retr-Sentenca* (Figura 5.8) é total pois para todo $uc =$ mk-$Regra(se, entao)$ (linha 31.1) atribui uma seqüência de caracteres formada pela concatenação de todas as condições (parte se de uma regra) juntamente com a ação (parte então de uma regra) (linha 31.2).

(ii) Seja $s : Sentenca$ arbitrário e $uc =$ mk-$Regra([], s)$. Temos então, $retr\text{-}Sentenca(uc) =$ conc $[] \frown s = s$. Como s é arbitrário podemos concluir $\forall\, s : Sentenca \cdot \exists\, uc : UndConhecimento \cdot retr\text{-}Sentenca(uc) = s$.

Proposição 5.4.1.1

(i) A função $retr\text{-}Observacao : RslProblema \rightarrow Observacao$ é total

(ii) $\forall\, obs : Observacao \cdot \exists\, rsl : RslProblema \cdot retr\text{-}Observacao(rsl) = obs$

Demonstração

(i) A função *retr-Observacao* (Figura 5.10) associa a cada *rsl*:*RslProblema* (linha 37.1) um mapeamento (linha 37.2); um mapeamento, entre *Passo* obtido pela aplicação da função *retr-Passo* (linha 36) a elementos de *rsl* linha (37.2) e *Sentenca* obtido pela aplicação da função *retr-Sentenca* a regras presentes em cada elemento de *rsl* (linhas 37.2 a 37.5). Assim *retr-Observacao* é total se *retr-Passo* e *retr-Sentenca* também o forem. Acima demonstramos que *retr-Sentenca* é total. Por fim, mostrar que *retr-Passo* é total é uma tarefa fácil; podemos ver diretamente este fato a partir de sua definição, linhas 36.1 e 36.2.

(ii) Seja *obs* : *Observacao* arbitrário. Seja ainda *rsl* : *RslProblema* tal que $obs = \{o \mapsto s\} \Leftrightarrow \mathsf{elems}\ rsl = \{\mathsf{mk}\text{-}Aplicacao(\text{-}, \mathsf{mk}\text{-}Regra([], s), o)\}$. *rsl* de fato existe: nada há (nenhum invariante) que impeça $\mathsf{mk}\text{-}Aplicacao(\text{-},\text{-},\text{-})$ e $\mathsf{mk}\text{-}Regra(\text{-},\text{-},\text{-})$. Agora, $retr\text{-}Observacao(rsl) = obs$ pois:

1. pela definição de *retr-Observacao* (linhas 37.1 a 37.5) e de *rsl* a partir de *obs* temos que $retr\text{-}Observacao(rsl) = \{retr\text{-}Passo(\mathsf{mk}\text{-}Aplicacao(\text{-}, \mathsf{mk}\text{-}Regra([], s), o)) \mapsto retr\text{-}Sentenca(\mathsf{mk}\text{-}Regra([], s))\}$;
2. pela definição de *retr-Passo* (linhas 36.1 e 36.2) temos que $retr\text{-}Passo(\mathsf{mk}\text{-}Aplicacao(\text{-}, \mathsf{mk}\text{-}Regra([], s), o)) = o$;
3. pela definição de *retr-Sentenca* (Figura 5.8, linhas 31.1 e 31.2) temos que $retr\text{-}Sentenca(\mathsf{mk}\text{-}Regra([], s)) = s$;
4. por fim, juntando 1., 2. e 3. temos que $retr\text{-}Observacao(rsl) = \{o \mapsto s\} = obs$.

Como por suposição *obs* é arbitrário podemos concluir $\forall\, obs : Observacao \cdot \exists\, rsl : RslProblema \cdot retr\text{-}Observacao(rsl) = obs$.

Proposição 5.4.1.2

(i) A função $retr\text{-}Ainfo : Regras \rightarrow Ainfo$ é total

(ii) $\forall\, i : Ainfo \cdot \exists\, rs : Regras \cdot retr\text{-}Ainfo(rs) = i$

Demonstraçao

(i) Para cada $\mathsf{mk}\text{-}Regras(err\text{-}2, err\text{-}1, crr\text{-}1)$ (Figura 5.10, linha 38.1), a função *retr-Ainfo* associa um conjunto de sentenças $\{retr\text{-}Sentenca\,(r) \mid r \in \bigcup\{err\text{-}2, err\text{-}1, crr\text{-}1\}\}$ (linha 38.2). Como *retr-Sentenca* é total (proposicao 5.3.3.1) concluímos que *retr-Ainfo* também é total.

(ii) Seja i : *Ainfo* arbitrário. Seja também rs = mk-*Regras*$(x, \{\}, \{\})$ onde $x = \{r \mid \forall s \in i \cdot retr\text{-}Sentenca(r) = s\}$. rs existe: na proposicao 5.3.3.1 mostramos que $\forall s : Sentenca \cdot \exists r : Regra \cdot retr\text{-}Sentenca(r) = s$. Existe e é por definição tal que $retr\text{-}Ainfo(rs) = \{retr\text{-}Sentenca(r) \mid r \in x\}$. Donde podemos concluir $retr\text{-}Ainfo(rs) = i$ pela definição de x a partir de i . Por suposição i é arbitrário; logo, $\forall i : Ainfo \cdot \exists rs : Regras \cdot retr\text{-}Ainfo(rs) = i$, como se queria mostrar.

Proposição 5.4.2.1

(i) $\forall rsl : RslProblema, rep_1 : Representacao\text{-}1 \cdot$
pre-*aquisicao*$(retr\text{-}Observacao(rsl), retr\text{-}Representacao(rep_1)) \Rightarrow$
pre-*analise-passos-resolucao*(rsl, rep_1)

(ii) $\forall rsl : RslProblema, rep_1 : Representacao\text{-}1, rs : Regras \cdot$
pre-*aquisicao*$(retr\text{-}Observacao(rsl), retr\text{-}Representacao(rep_1)) \wedge$
post-*analise-passos-resolucao*$(rsl, rep_1, rs) \Rightarrow$
post-*aquisicao*$(retr\text{-}Observacao(rsl), retr\text{-}Representacao(rep_1), retr\text{-}Ainfo(rs))$

Demonstração

(i) Observando a especificação de *analise-passos-resolucao* (Figura 5.11, linha 39) vemos que pre-*analise-passos-resolucao*(-,-) = true. Assim, independentemente de pre-*aquisicao*(-,-), a parte (i) da proposição 5.4.2.1 é satisfeita.

(ii) A demonstração da parte (ii) requer uma argumentação mais elaborada da qual mostramos apenas a linha mestra. Notemos inicialmente que (ii) tem a seguinte estrutura: $\forall x \cdot A(x) \wedge B(x) \Rightarrow C(x)$. Observando a especificação da função *aquisicao* (Figura 5.4) vemos que $A(x) \leftarrow$ pre-*aquisicao*(-,-) = true [1]. Assim resta-nos mostrar $\forall x \cdot B(x) \Rightarrow C(x)$. Façamos $B(x) \leftarrow$ post-*analise-passos-resolucao*(rsl, rep_1, rs) = true para rsl, rep_1 e rs arbitrários. Há então que se mostrar que $C(x) \leftarrow$ post-*aquisicao*$(retr\text{-}Observacao(rsl), retr\text{-}Representacao(rep_1), retr\text{-}Regras(rs))$ = true.

Pela especificação de *analise-passos-resolucao* (linha 39), $B(x)$ = true temos que:

1. existem regras $\exists r \in$ dom $rep_1.mc \cdot$ que são isoladas pelo processo de aquisição, ou seja, $r \in \bigcup\{rs.err\text{-}2, rs.err\text{-}1, rs.crr\text{-}1\}$;

[1] $A(x) \leftarrow$ ... leia-se com "a $A(x)$ é atribuído ...".

2. ainda, estas regras são tais que há passos de resolução $\exists\, p \in \mathsf{elems}\ rsl \cdot$ onde elas foram aplicadas ou deixaram de ser aplicadas; quer dizer, $p.r \vee (p.r = \mathsf{nil} \wedge r \in \mathsf{dom}\ p.ra)$;

3. pois bem, 1. nos leva a concluir que as sentenças $s \in retr\text{-}Ainfo(rs)$ (ver Figura 5.10, linha 38) pertencem ao $\mathsf{dom}\ retr\text{-}Representacao(rep_1).mc = retr\text{-}Representacao(rep_1).md$ (ver Figura 5.7, linha 26).

4. ainda, 2. nos leva a concluir que as mesmas sentenças $s \in retr\text{-}Ainfo(rs)$ estão relacionadas com passos observados, ou seja, $\exists\, p \in \mathsf{dom}\ retr\text{-}Observacao(rsl) \cdot retr\text{-}Observacao(rsl)(o) = s$ (ver Figura 5.10, linha 37).

Juntando as duas últimas conclusões com a especificação de *aquisicao* (Figura 5.4, linha 14) podemos mostrar que $C(x) \leftarrow \mathsf{post}\text{-}aquisicao(retr\text{-}Observacao(rsl), retr\text{-}Representacao(rep_1), retr\text{-}Regras(rs)) = \mathsf{true}$; "*quod erat demonstrandum*".

Proposição 5.5.1.1

(i) $\forall\, rs : Regras, rep_1 : Representacao\text{-}1 \cdot$
$\mathsf{pre}\text{-}manutencao(retr\text{-}Ainfo(rs), retr\text{-}Representacao(rep_1)) \Rightarrow$
$\mathsf{pre}\text{-}manut\text{-}sobreposicao\text{-}difusa(rs, rep_1)$

(ii) $\forall\, rs : Regras, rep_1 : Representacao\text{-}1, rep_1' : Representacao\text{-}1 \cdot$
$\mathsf{pre}\text{-}manutencao(retr\text{-}Ainfo(rs), retr\text{-}Representacao(rep_1)) \wedge$
$\mathsf{post}\text{-}manut\text{-}sobreposicao\text{-}difusa(rs, rep_1, rep_1') \Rightarrow$
$\mathsf{post}\text{-}aquisicao(retr\text{-}Ainfo(rs), retr\text{-}Representacao(rep_1), retr\text{-}Representacao(rep_1'))$

Demonstração

(i) Observando a linha 46.1 da especificação da função *manut-sobreposicao-difusa* (Figura 5.13) vemos que $\mathsf{pre}\text{-}manut\text{-}sobreposicao\text{-}difusa(\text{-},\text{-}) = \mathsf{true}$. Assim, independentemente de $\mathsf{pre}\text{-}aquisicao(\text{-},\text{-})$, a parte (i) da proposição 5.5.1.1 é satisfeita.

(ii) Já a parte (ii) da proposição 5.5.1.1 requer uma argumentação análoga à empregada na parte (ii) da proposição 5.5.1.1. Deve-se então mostrar que $\mathsf{post}\text{-}manutecao(retr\text{-}Ainfo(rs), retr\text{-}Representacao(rep_1), retr\text{-}Representacao(rep_1')) =$

true para valores rs, rep_1 e rep'_1 arbitrários tais que post-*manut-sobreposicao-difusa*(rs, rep_1, rep'_1) = true.

Para valores rs, rep_1 e rep'_1 arbitrários tais que post-*manut-sobreposicao-difusa*(rs, rep_1, rep'_1) = true concluímos que:

1. $rep'_1.mc = rep.mc \dagger m$,
2. onde $m = m' \dagger m'' \dagger m'''$ (pelas linhas 46.12, 46.9, 46.6 e 46.3) sendo dom $m' = rs.crr\text{-}1 \cap rep.mc$ (pelas linhas 46.4 e 46.5), dom $m'' = rs.err\text{-}1 \cap rep.mc$ (pelas linhas 46.7 e 46.8) e dom $m''' = rs.err\text{-}2 \cap rep.mc$ (pelas linhas 46.10 e 46.11);
3. de 2. concluimos que dom $m = \bigcup\{rs.crr\text{-}1, rs.err\text{-}1, rs.err\text{-}2\} \cap rep.mc$;
4. agora, vendo 1. e 3. via funções "*retrieve*" *retr-Ainfo* e *retr-Representacao*-1 obtemos *retr-Representacao*-1$(rep'_1).mc$ = *retr-Representacao*-1$(rep_1).mc \dagger m*$, para algum mapeamento $m* : Sentenca \mapsto LI$,
5. tal que dom $m* = retr\text{-}Ainfo(rs) \cap retr\text{-}Representacao\text{-}1(rep_1)$; relações que comparadas com as definições das funções "*retrieve*" nas Figuras 5.7, 5.10 mostram-se verdadeiras.

Por fim, juntando as conclusões 4. e 5. e comparando com a especificação de *manutencao* (Figura 5.5), podemos concluir que post-*manutecao*$(retr\text{-}Ainfo(rs), retr\text{-}Representacao(rep_1), retr\text{-}Representacao(rep'_1))$ = true para valores rs, rep_1 e rep'_1 arbitrários tais que post-*manut-sobreposicao-difusa*(rs, rep_1, rep'_1) = true.

Proposição 5.6.1.1

(i) A função $retr\text{-}MA : MA\text{-}1 \rightarrow MA$ é total

(ii) $\forall\, ma : MA \cdot \exists\, ma_1 : MA\text{-}1 \cdot retr\text{-}MA(ma_1) = ma$

Demonstração

(i) Mostrar que *retr-MA* é total reduz-se em última análise a mostrar que *retr-Representacao* é total. Notemos que *retr-MA* (Figura 5.15, linha 52) parte de um $ma : MA$ (Figura 5.2, linha 2) e cria um $ma_1 : MA$-1 preservando os aprendizes em ma e modificando a representação associada a cada aprendiz por meio da função *retr-Representacao*. O fato de que *retr-Representacao* é total foi demonstrado anteriormente neste apêndice.

(ii) Também já foi demonstrado que $\forall\, r : Representacao \cdot \exists\, r_1 : Representacao\text{-}1 \cdot retr\text{-}Representacao(r_1) = r$. Dessa forma, para um $ma : MA$ arbitrário, $\forall\, a \in \mathsf{dom}\ ma \cdot \exists\, r_1 : Representacao\text{-}1 \cdot retr\text{-}Representacao(r_1) = ma(a)$. Fazendo então $r_1 = ma_1(a)$, para algum $ma_1 : MA\text{-}1$ tal que $\mathsf{dom}\ ma = \mathsf{dom}\ ma_1$, e comparando estes fatos com a definição de *retr-MA* podemos concluir que: $\exists\, ma_1 : MA\text{-}1 \cdot retr\text{-}MA(ma_1) = ma$. Como partimos da hipótese de que ma era arbitrário demonstramos a parte (ii) da proposição 5.6.1.1.

B.2 Capítulo 6

Proposição 6.2.0.1

Sejam $cur : Curriculum, dpr : Problema\text{-}\mathsf{set}$, respectivamente, o *Curriculum* e o Domínio de Problemas do Sistema Tutor de um agente MATHEMA (Figura 5.6, linhas 22.0 e 22.1) e uma seqüência de aprendizagem $md = \mathsf{mk}\text{-}SequenciaUP(up, cur)$ (Figura 5.7, linha 23) tal que $\mathsf{elems}\ up = cur.prt$.

O conjunto

$$
\begin{array}{ll}
58.0 & P\,(md, dpr) \triangleq \{ps \mid ps : UndProblema \cdot \\
.1 & \qquad ps \neq \{\} \wedge \\
.2 & \qquad \exists\, n \in \mathsf{inds}\ md.up \cdot \\
.3 & \qquad\quad \mathsf{post}\text{-}unidade\text{-}problema(n, md, dpr, ps)\}
\end{array}
$$

é uma partição de dpr. Quer dizer:

(i) $\{\} \notin P(md, dpr)$

(ii) $\forall\, ps_1, ps_2 \in P(md, dpr) \cdot$
$(ps_1 \neq ps_2) \Rightarrow (ps_1 \cap ps_2 = \{\})$

(iii) $\bigcup P(md, dpr) = dpr$

Demonstração

O item (i) segue de forma direta da definição de $P(md, dpr)$; ver linha 58.1.

O item (ii) ; sejam $ps_1, ps_2 \in P(md, dpr)$ abitrários; suponhamos ainda que $ps_1 \neq ps_2$. Então, pela defição de $P(md, dpr)$, devem existir $n_1 \neq n_2$ tais que $\mathsf{post}\text{-}unidade\text{-}problema(n_1, md, dpr, ps_1)$, $\mathsf{post}\text{-}unidade\text{-}problema(n_2, md, dpr, ps_2)$. Usando n_1 e n_2 devemos mostrar que $ps_1 \cap ps_2 = \{\}$.

Suponhamos que existe um $p \in ps_1 \cap ps_2$. Pela especificação da função *unidade-problema*, linha 56.3, (1) $p.ucs \cap md.up(n_1) \neq \{\}$ e (2) $p.ucs \cap md.up(n_2) \neq \{\}$. Pela linha 56.5 (a) $p.ucs \cap crp \subseteq \bigcup \text{elems } md.up(1, ..., n_1)$ e (b) $p.ucs \cap crp \subseteq \bigcup \text{elems } md.up(1, ..., n_2)$. Ora, $n_1 \neq n_2$; digamos que $n_1 < n_2$. Por (a) vemos que todo conhecimento necessário para resolver p está contido nas unidades pedagógicas de $md.up(1, ..., n_1)$; por (2), no entanto, vemos que parte do conhecimento necessário para resolver p está na unidade pedagógica $md.up(n_2)$. Uma contradição uma vez que fizemos $n_1 < n_2$. Um argumento análogo a este fazendo $n_2 < n_1$ leva também a uma contradição. Como partimos da hipótese de que existia $p \in ps_1 \cap ps_2$ podemos concluir que $ps_1 \cap ps_2 = \{\}$, como se queria mostrar.

O item (iii) ; todo problema pertencente a $P(md, dpr)$ pelas definições de $P(md, dpr)$, linha 58.2, e *unidade-problema*, linha 56.2, também pertence a dpr. Quer dizer : $P(md, dpr) \subseteq dpr$.

Imaginemos agora um $p \in dpr$ arbitário. Pela definição das Bases de Conhecimento do Sistema Tutor dos agentes MATHEMA (Figura 5.6, linhas 22.6-22.8), $p.ucs \cap crp \neq \{\}$, onde, $crp = \bigcup cur.prt$. Uma das suposições iniciais da proposição é que $cur.prt = \text{elems } md.up$. Donde $crp = \bigcup cur.prt = \bigcup \text{elems } md.up$. Assim $p.ucs \cap crp \subseteq \bigcup \text{elems } md.up$. Este último fato comparado com as linhas 56.3 e 56.5 da especificação da função *unidade-problema* nos leva a ter certeza de que $\exists\, n \in \text{inds } md.up, ps : Problema\text{-set} \cdot$ tais que post-$unidade\text{-}problema(n, md, dpr, ps)$ e $p \in ps$. Como $n \in \text{inds } md.up$, $ps \in P(md, dpr)$. Logo $p \in \bigcup P(md, dpr)$. Quer dizer: $dpr \subseteq P(md, dpr)$.

Bibliografia

[ABH+95] D. J. Andrews, H. Bruun, B. S. Hansen, P. G. Larsen, N. Plat, et. al. *Information technology - programming languages, their environments and system software interfaces - vienna development method-specification language part 1: Base language.* ISO, 1995. Draft International Standard: 13817-1.

[ACC+93] J. R. Anderson, F. G. Conrad, A. T. Corbett, J. M. Finchamand D. Hoffman, e Q. Wu. *Computer programming and transfer*, cap 10. Lawrence Erlbaum, 1993.

[ACKP95] J. Anderson, A. Corbett, K. Koedinger, e R. Pelletier. Cognitive tutors: lessons learned. *The Journal of the Learning Sciences*, 4(2):167–207, 1995.

[AS95] Fabio N. Akhras e John Self. A process-oriented perspective on analysing learner-environment interactions in constructive learning. In *Proceedings of the 6th Brazilian Symposium on Computing in Education (SBIE'95)*, Florianópolis, Brasil, 1995. SBC.

[AZN98] David W. Albrecht, Ingrid Zukerman, e Anne E. Nicholson. Bayesian models for keyhole plan recognition in an adventure game. *User Modeling and User-Adapted Interaction*, 8(1+2), 1998.

[Bar92] P. Barker. An object oriented approach to hypermedia authoring. In M. Giardina, editor, *Interactive Multimedia Learning Environments*, pp 132–152. Springer-Verlag, Berlin, 1992.

[Bea94] Ian Beaumont. User modeling in the interactive anatomy tutoring system Anatom-Tutor. *User Modeling and User-Adapted Interaction*, 4(1):21–45, 1994.

[BK86] Raj K. Bhatnagar e Laveen N. Kanal. Handling uncertain information: a review of numeric and non-numeric methods. In L. N. Kanal e J. F. Lem-

mer (eds). *Uncertainty in Artificial Intelligence*, pp 3–26. Elsevier Science Publishers, North-Holland, 1986.

[BM92] P. Baffes e R. Mooney. Using theory revision to model students and acquire stereotypical errors. In *Proc. Of the Fourteenth Annual Conference of the Cognitive Science Society*, pp 617–622, Bloomington, IN, 1992.

[BSW97] Joseph Beck, Mia Stern, e Beverly Park Woolf. Using the student model to control problem difficulty. In Anthony Jameson, Cécile Paris, e Carlo Tasso, (eds). *User Modeling: Proceedings of the Sixth International Conference, UM97*, pp 277–288. Springer Wien New York, Vienna, New York, 1997. Disponvel em http://um.org.

[Buc92] Paulo Bucchi. *Matemática: ensino de 2o. grau*, volume único. Editora Moderna, São Paulo, 1 edição, 1992.

[Bur82] R. R. Burton. Diagnosing bugs in a simple procedural skill. In D. H. Sleeman e J. S. Brown, (eds). *Intelligent Tutoring Systems*, pp 157–184. Academic Press, London, 1982.

[CA95] Albert T. Corbett e John R. Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction*, 4(4):253–278, 1995.

[Car70] J. R. Carbonell. AI in CAI: an artificial intelligence approach to computer-assisted instruction. *IEEE Transactions on Man Machine Systems*, 11(4):190–202, 1970.

[CB90] T. Chan e A. B. Baskin. Learning companion systems. In C. Frasson e G. Gauthier, (eds). *Intelligent Tutoring Systems*. Alex Publishing Corp., New Jersey, 1990.

[CG77] B. Carr e I. Goldstein. Overlays: a theory of modelling for computer aided instruction. Relatório técnico, AI Memo 406, MIT, Cambridge, 1977.

[Cha91] E. Charniak. Bayesian networks without tears. *AI Magazine*, 12(4):50–63, 1991.

[Cha95] Tak-Wai Chan. A tutorial on social learning systems. In T. Chan e J. Self, (eds). *Emerging Computer Technologies in Education*, pp 71–96. AACE, Charlottesville, 1995.

[Cla83] W. J. Clancey. Guidon. *Journal of Computer-Based Instruction*, 10(1):8–14, 1983.

[Cla88] William J. Clancey. The role of qualitative models in instruction. In John Self, editor. *Artificial Intelligence and Human Learning*, cap 3, pp 49–68. Chapman and Hall Ltd, London, 1988.

[Cla95] W. J. Clancey. A tutorial on situated learning. In T. Chan e J. Self, (eds). *Emerging Computer Technologies in Education*, pp 71–96. AACE, Charlottesville, 1995.

[CMRS95] Antonella Carbonaro, Vittorio Maniezzo, Marco Roccetti, e Paola Salomoni. Modelling the student in Pitagora 2.0. *User Modeling and User-Adapted Interaction*, 4(4):233–251, 1995.

[Cos97] Evandro Barros Costa. *Um modelo de ambiente interativo de aprendizagem baseado numa arquitetura multi-agentes.* Tese de Doutorado, Universidade Federal da Paraíba, Campina Grande, PB, dezembro, 1997.

[CR92] A. T. Corbett e Anderson J. R. Student modeling and mastery learning in a computer-based programming tutor. In C. Frasson, G. Gauthier, e G. I. McCalla, (eds). *Intelligent Tutoring Systems: Proceedings of the Second International Conference, ITS'92*, pp 413–420, Berlin, 1992. Springer.

[CV96] C. Conati e K. VanLehn. Pola: A student modeling framework for probabilistic on-line assessment of problem solving performance. In *Proceedings of the 5th International Conference on User Modeling*, pp 75–82, Kailua-Kona, HI, 1996.

[Dan97] Carlos B. Dantas. *Probabilidade: um curso introdutório.* EDUSP, São Paulo, 1997.

[DBBO94] P. Dillenbourg, M. Baker, A. Blaye, e C. O'Malley. The evolution of research on collaborative learning, 1994. http://tecfa.unige.ch/tecfa-research/lhm/ESF-Chap5.text.

[DMS94] P. Dillenbourg, P. Mendelsohn, e D. Schneider. The distribuition of pedagogical roles in a multi-agent learning environment. In R. Lewis e P. Mendelsohn, (eds). *Lessons from Learning*, pp 199–216. North-Holland, Amsterdam, 1994.

[DP90] B. A. Davey e H. A. Priestley. *Introduction to lattices and order.* Cambridge University Press, Cambridge, 1990.

[DS92a] T. Del Soldato. Detecting and reacting to the learner's motivational state. In *Proceedings of ITS'92*, pp 567–574. Springer-Verlag, 1992.

[DS92b] Pierre Dillenbourg e John Self. A framework for learner modelling. *Interactive Learning Environments*, 2:111–37, 1992.

[EC88] Mark Elsom-Cook. Guided discovery tutoring and bounded user modelling. In John Self, editor. *Artificial Intelligence and Human Learning*, cap 10, pp 165–178. Chapman and Hall Ltd, London, 1988.

[Fin89] Timothy W. Finin. Gums - a general user modeling shell. In A. Kobsa e W. Wahslter, (eds). *User Models in Dialog Systems*, cap 15, pp 411–430. Springer-Verlag, Berlin, 1989.

[FKV94] Martin D. Fraser, Kuldeep Kumar, e Vijay K. Vaishnavi. Strategies for incorporating formal specifications in software development. *CACM*, 37(10):74–85, 1994.

[GDB92] Antonio Gisolfi, Antonina Dattolo, e Walter Balzano. A fuzzy approach to student modeling. *Computers Educ.*, 19(4):329–334, 1992.

[GDS95] D. W. Gurer, M. DesJardins, e M. Schlager. Representing a student's learning states and transitions. In *Working Notes of the AAAI-95 Spring Symposium on Representing Mental States and Mechanisms.* AAAI Press, Março 1995.

[Gen82] M. R. Genesereth. The role of plans in intelligent teaching systems. In D. H. Sleeman, editor, *Intelligent Tutoring Systems*, cap 7, pp 137–155. Academic Press, New York, 1982.

[Gia92] M. Giardina, editor. *Interactive Multimedia Learning Environments.* Springer-Verlag, Berlin, 1992.

[GK95] J. E. Greer e G. M. Koehn. The peculiarities of plan recognition for intelligent tutoring systems. In *IJCAI Plan Recognition Workshop*, Montreal, Agosto 1995.

[GMM89] J. Greer, M. Mark, e G. McCalla. Incorporating granularity-based recognition into scent. In D. Bierman, J. Breuker, e J. Sandberg, (eds). *Proc.*

Of the 4th International Conferencee on AI and Education, pp 107–115, Amsterdam, 1989.

[Gol82] I. P. Goldstein. The genetic graph: a representation for the evolution of procedural knowledge. In D. Sleeman e J. S. Brown, (eds). *Intelligent Tutoring Systems*, cap 3, pp 51–78. Academic Press, New York, 1982.

[Gro98] The VDM Tool Group. *Vdm-sl toolbox user manual.* Relatório técnico, IFAD, Outubro 1998.

[GS88] David Gilmore e John Self. The application of machine learning to intelligent tutoring systems. In John Self, editor, *Artificial Intelligence and Human Learning*, cap 11, pp 179–196. Chapman and Hall Ltd, London, 1988.

[GT94] P. Giangrandi e C. Tasso. Truth maintenance techniques for modelling students behaviour. Relatório técnico, Laboratorio di Intelligenza Artificiale, Universita di Udine, 1994.

[GT96] P. Giangrandi e C. Tasso. Modeling the temporal evolution of student's knowledge. In P. Brna, A. Paiva, e J. Self, (eds). *Proceedings of the European Conference on Artificial Intelligence in Education*, pp 184–190, 1996.

[GT97] P. Giangrandi e C. Tasso. Managing temporal knowledge in student modeling. In Anthony Jameson, Cécile Paris, e Carlo Tasso, (eds). *User Modeling: Proceedings of the Sixth International Conference, UM97*, pp 415–426. Springer Wien New York, Vienna, New York, 1997. Disponível em http://um.org.

[HDJG94] P. Holt, S. Dubs, M. Jones, e J. Greer. The state of student modelling. In Jim E. Greer e Gordon I. McCalla, (eds). *Student Modelling: the Key to Individualized Knowledge-Based Instruction*, NATO ASI Series F: Computer and Systems Sciences, vol 125, cap 1, pp 3–35. Springer-Verlag, Berlin, 1994.

[HMGN91] Xueming Huang, Gordon I. McCalla, Jim E. Greer, e Eric Neufeld. Revising deductive knowledge and stereotypical knowledge in a student model. *User Modeling and User-Adapted Interaction*, 1(1):87–115, 1991.

[Jam96] Anthony Jameson. Numerical uncertainty management in user and student modeling: An overview of systems and issues. *User Modeling and User-Adapted Interaction*, 5(3-4):193–251, 1996.

[JM93] A. Jones e N. Mercer. Theories of learning and information technology. In P. Scrimshaw, editor, *Language, Classrooms and Computers*, pp 11–26. Routledge, London, 1993.

[Jon89] Karen Sparck Jones. Realism about user modeling. In A. Kobsa e W. Wahslter, (eds). *User Models in Dialog Systems*, cap 12, pp 341–363. Springer-Verlag, Berlin, 1989.

[Jon90] Cliff B. Jones. *Systematic Software Development Using VDM.* Prentice-Hall International, Englewood Cliffs, New Jersey, 2 edição 1990.

[JW92] M. Jones e P. Winne, (eds). *Adaptive learning environments: foundations and frontiers.* Springer-Verlag, Berlin, 1992.

[Kas89] R. Kass. Student modeling in intelligent tutoring systems - implications for user modeling. In A. Kobsa e W. Wahslter, (eds). *User Models in Dialog Systems*, cap 14, pp 386–410. Springer-Verlag, Berlin, 1989.

[KF88] George J. Klir e Tina A. Folger. *Fuzzy sets, uncertainty and information.* Prentice Hall, Englewood Cliffs, New Jersey, 1988.

[KG94] G. Koehn e J. E. Greer. Recognizing plans in instructional systems using granularity. In *Proceedings of the International Conference on User Modelling*, pp 133–138, Cape Cod, MA, Agosto 1994.

[KIM92] Y. Kono, M. Ikeda, e R. Mizoguchi. To contradict is human - student modeling of inconsistency. In C. Frasson, G. Gauthier, e G. I. McCalla, (eds). *Intelligent Tutoring Systems: Proceedings of the Second International Conference, ITS'92*, Berlin, 1992. Springer.

[KLEG94] Sandra Katz, Alan Lesgold, Gary Eggan, e Maria Gordin. Modeling the student in Sherlock II. In Jim E. Greer e Gordon I. McCalla, (eds). *Student modelling: the key to individualized knowledge-based instruction*, NATO ASI Series F: Computer and Systems Sciences, vol 125, pp 99–125. Springer-Verlag, Berlin, 1994.

[Koe94] G. Koehn. Granularity-based plan recognition. Tese de Mestrado, Departamento de Ciência da Computação, Universidade de Saskatchewan, 1994. ARIES Relatório técnico 94-5.

[LD93a] S. P. Lajoie e S. J. Derry. *Computers as cognitive tools.* Lawrence Erlbaum Associates, Hillsdale, 1993.

[LD93b] B. Loftin e C. Dede. Described in surreal science. *Scientific American*, p 103, fevereiro 1993.

[LF98] S. Labidi e J. S. Ferreira. Technology-assisted instruction applied to co-operative learning: the Shiecc project. In *Proceedings of the IEEE International Conference Frontiers in Education FIE' 98*, Tempe, Arizona, Novembro 1998.

[LHB+96] P. G. Larsen, B. S. Hansen, H. Brunn, N. Plat, H. Toetenel, D. J. Andrews, J. Dawes, G. Parkin, et. al. *Information technology - programming languages, their environments and system software interfaces - vienna development method - specification language - part 1: Base language.* Dezembro 1996.

[LL89] S. P. Lajoie e A. Lesgold. Apprenticeship training in the workplace: A computer coached practiceenvironment as a new form of apprenticeship. *Machine Mediated Lerning*, 3(1):7–28, 1989.

[LOS84] P. Langley, S. Ohlsson, e A. Sage. Machine learning approach to student modeling. Relatório técnico CMU-RI-TR-84-7, Robotics Institute, Carnegie-Mellon University, Pittsburgh, Pennsylvania, 1984.

[Mar95] June Marquis. Erros comuns em álgebra. In Arthur F. Coxford and Albert P. Shulte, (eds). *As Idéias da Álgebra*, pp 234–236. Atual Editora, São Paulo, 1995.

[McC92] G. I. McCalla. The centrality of student modelling to intelligent tutoring. In E. Costa, editor, *New Directions for Intelligent Tutoring Systems*, volume 91, *NATO ASI Series*, pp 107–131. Springer-Verlag, Berlin, 1992.

[MG94] G. I. McCalla e J. E. Greer. Granularity-based reasoning and belief revision in student models. In Jim E. Greer e Gordon I. McCalla, (eds). *Student Modelling: the Key to Individualized Knowledge-Based Instruction*, NATO ASI Series F: Computer and Systems Sciences, vol 125, cap 2, pp 39–62. Springer-Verlag, Berlin, 1994.

[Mur97a] Elizabeth Murphy. *Learning environments: readings and reflections.* http://www.stemnet.nf.ca/ elmurphy/emurphy/learning1.html, 1997.

[Mur97b] W. Murray. *Intelligent tools and instructional simulations - the desktop associate, final report.* Relatório técnico, Teknowledge Corporation, Outubro 1997.

[MV95] J. D. Martin e K. VanLehn. A bayesian approach to cognitive assessment. In P. D. Nichols, S. F. Chipman, e R. L. Brennan, (eds). *Cognitively Diagnostic Assessment*, pp 141–165. Erlbaum, Hillsdale, NJ, 1995.

[Nea90] R. E. Neapolitan. *Probabilistic Reasoning in Expert Systems: theory and algorithms.* Wiley, New York, 1990.

[Ohl94] Stellan Ohlsson. Constraint-based student modeling. In Jim E. Greer e Gordon I. McCalla, (eds). *Student Modelling: the Key to Individualized Knowledge-Based Instruction*, NATO ASI Series F: Computer and Systems Sciences, vol 125, cap 7, pp 167–190. Springer-Verlag, Berlin, 1994.

[Pai95] A. Paiva. *About user and learner modeling - an overview.* http://cbl.leeds.ac.uk/amp/personal.html, 1995.

[Pap80] S. Papert. *Mindstorms: children, computers, and powerful ideas.* Basic Books, New York, 1980.

[Pea88] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: networks of plausibleinference.* Morgan Kaufmann, San Mateo, 1988.

[PPS87] OK-Choon Park, R. S. Perez, e R. J. Seidel. Intelligent cai: old wine in new bottles, or a new vintage? In G. Kearsley, editor, *Artificial Intelligence and Instruction: Applications and Methods*, cap 2, pp 11–45. Addison-Wesley, Reading, 1987.

[PS94] A. Paiva e J. Self. A learner model reason maintenance system. In A. Cohn, editor, *ECAI94. 11th European Conference on Artificial Intelligence*, pp 178–181. John Wiley & Sons, Ltd., 1994.

[PS95] A. Paiva e J. Self. Tagus – a user and learner modeling workbench. *User Modeling and User-Adapted Interaction*, 4(3):197–226, 1995.

[PSH94] A. Paiva, J. Self, e R. Hartley. On the dynamics of learner models. In A. Cohn, editor, *ECAI94. 11th European Conference on Artificial Intelligence*, pp 163–167. John Wiley & Sons, Ltd., 1994.

[PSH95] A. Paiva, J. Self, e R. Hartley. Externalising learner models. In J. Greer, editor, *International Conference on Artificial Intelligence in Education*, Washington, DC, 1995.

[RAF85] B. J. Reiser, J. R. Anderson, e R. G. Farrell. Dynamic student modelling in an intelligent tutor for lisp programming. In *Proceedings IJCAI-85*, pp 8–14, 1985.

[Rag96] Eva L. Ragnemalm. Student diagnosis in practice: Bridging a gap. *User Modeling and User-Adapted Interaction*, 5(2):93–116, 1996.

[Rey96] J. Reye. A belief net backbone for student modelling. In C. Frasson, G. Gauthier, e A. Lesgold, (eds). *Proceedings ITS '96*, pp 596–604, Montreal, Junho 1996. Springer.

[Ric79] E. Rich. User modelling via stereotypes. *Cognitive Science*, 3:329–354, 1979.

[Ric89] E. Rich. Stereotypes and user modeling. In A. Kobsa e W. Wahslter, (eds). *User Models in Dialog Systems*, cap 2, pp 35–51. Springer-Verlag, Berlin, 1989.

[Rie92] L. P. Rieber. Computer-based microwords: a bridge between construtivism and direct instruction. *Educational Technology Research and Development*, 40(1):93–106, 1992.

[Rie94] L. P. Rieber. *Computers, Graphics and learning.* Wm. D. Brown Communications, Inc., Dubuque, Iowa, 1994.

[RN95] S. Russell e P. Norvig. *Artificial Intelligence: a modern approach.* Prentice-Hall, New Jersey, 1995.

[SE89] R. C. Schanck e D. J. Edelson. A role for ai in education: using technology to reshape education. *International Journal of Artificial Intelligence in Education*, 1(2):3–20, 1989.

[Sel88] John Self. Student models: what use are they? In P. Ercoli e R. Lewis, (eds). *Artificial Intelligence Tools in Education*, pp 73–86. Elsevier Science Publishers, North-Holland, 1988.

[Sel90] John Self. Bypassing the intractable problem of student modelling. In C. Frasson e G. Gauthier, (eds). *Intelligent Tutoring Systems: at the Crossroads of Artificial Intelligence and Education*, pp 107–23. Ablex, Norwood, 1990.

[Sel94a] John Self. Formal approaches to student modelling. In Jim E. Greer e Gordon I. McCalla, (eds). *Student Modelling: the Key to Individualized Knowledge-Based Instruction*, NATO ASI Series F: Computer and Systems Sciences, vol 125, cap 12, pp 295–252. Springer-Verlag, Berlin, 1994.

[Sel94b] John Self. The role of student models in learning environments. *Transactions of the Institute of Eletronic, Information and Communication Engineers*, E77-D(1):3–8, 1994.

[Sel95a] John Self. Computational mathetics: towards a science of learning systems design. http://cbl.leeds.ac.uk/jas, 1995.

[Sel95b] John Self. Dormorbile: a vehicle for metacognition. In T. Chan e J. Self, (eds). *Emerging Computer Technologies in Education*. AACE, Charlottesville, 1995.

[Sha76] G. Shafer. *A mathematical theory of evidence*. Princeton University Press, 1976.

[Shu95] Valerie J. Shute. Smart: Student modeling approach for responsive tutoring. *User Modeling and User-Adapted Interaction*, 5(1):1–44, 1995.

[Sil99] J. C. Silva. Aquisição de conhecimento e manutenção para uma sociedade de agentes tutores artificiais. Tese de Mestrado, COPIN, UFPB - Campus II, Campina Grande, PB, fevereiro 1999.

[Sle82] D. Sleeman. Assessing aspects of competence in basic algebra. In D. H. Sleeman, editor, *Intelligent Tutoring Systems*, cap 9, pp 185–199. Academic Press, New York, 1982.

[SP95] Valerie Shute e Joseph Psotka. Intelligent tutoring systems: past, present, and future. In D. Jonassen, editor, *Handbook of Research on Educational Communications and Technology*. Scholastic Publications, 1995.

[SS81] D. H. Sleeman e M. J. Smith. Modelling students' problem solving. *Artificial Intelligence*, 16(2):171–187, maio 1981.

[SS96] R Sison e M. Shimura. The application of machine learning to student modeling: a survey andcompartative analysis. Relatório té, Dept. de Ciência da Computação, Tokyo Inst. of Technology, 1996.

[Ste95] M. Stefik. *Introduction to knowledge systems.* Morgan Kaufmann, San Francisco, 1995.

[TF93] N. Tokuda e A. Fukuda. A probabilistic inference scheme for hierarchical buggy models. *International Journal of Man-Machine Studies*, 38:857–872, 1993.

[Tho87] P. W. Thompson. Mathematical microworlds and intelligent computer-assisted instruction. In G. Kearsley, editor, *Artificial Intelligence and Instruction: Applications and Methods*, cap 5, pp 83–109. Addison-Wesley, Reading, 1987.

[Van96] K. VanLehn. Conceptual and meta learning during coached problem solving. In C. Frasson, G. Gauthier, e A. Lesgold, (eds). *Proceedings of the 3rd International Conference on Intelligent Tutoring Systems ITS'96*, pp 29–47, Berlin, 1996. Springer.

[Vas91] Julita Vassileva. A classification and syntesis of student modelling techniques in intelligent computer-assisted instruction. In D.H. Norrie e H.-W. Six, (eds). *Computer Assisted Learning*, pp 202–213. Springer-Verlag, 1991.

[Ver94] M. F. Verdejo. Building a student model for an intelligent tutoring system. In Jim E. Greer e Gordon I. McCalla, (eds). *Student Modelling: the Key to Individualized Knowledge-Based Instruction*, NATO ASI Series F: Computer and Systems Sciences, vol 125, cap 6, pp 147–164. Springer-Verlag, Berlin, 1994.

[Vil92] M. Villano. Probabilistic student models: bayesian belief networks and knowledge space theory. In C. Frasson, G. Gauthier, e G. I. McCalla, (eds). *Intelligent Tutoring Systems: Proceedings of the Second International Conference, ITS'92*, pp 491–498, Berlin, 1992. Springer.

[VL87] K. Van Lehn. Learning one subprocedure per lesson. *Artificial Intelligence*, 31(1):1–40, 1987.

[Was96] Barbara Wasson. Instructional planning and contemporary theories of learning: is thisa self-contradiction? In P. Brna, A. Paiva, e J. Self, (eds). *Proceedings of the European Conference on Artificial Intelligence in Education*, pp 23–30, Lisbon, 1996. Colibri.

[Wen87] Etienne Wenger. *Artificial intelligence and tutoring systems.* Morgan Kaufmann Publishers, Los Altos, 1987.

[Wil92] K. Wilson. Discussion on two multimedia R & D projects. In M. Giardina, editor, *Interactive Multimedia Learning Environments*, pp 186–196. Springer-Verlag, Berlin, 1992.

[Wil96] B Wilson, editor. *Constructivist learning environments: case studies in instructional design.* Educational Technology Publications, New Jersey, 1996.

[Win93] W. Winn. A constructivist critique of the assumptions of instructional design. In T. M. Duffy, J. Lowyck, e D. H. Jonassen, (eds). *Designing Environments for Constructive Learning*, NATO ASI Series F., pp 139–160. Springer-Verlag, Heidelberg, 1993.

[Woo88] Mark R. Woodroffe. Plan recognition and intelligent tutoring systems. In John Self, editor, *Artificial Intelligence and Human Learning*, cap 13, pp 212–225. Chapman and Hall Ltd, London, 1988.

[Woo92] B. Woolf. Towards a computational model of tutoring. In M. Jones e P. Winne, (eds). *Adaptive Learning Environments: Foudations and Frontiers*, pp 209–232. Springer-Verlag, Berlin, 1992.

[Zad88] Lofti A. Zadeh. Fuzzy logic. *IEEE Computer Mag.*, pp 83–93, Abril 1988.