UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Um Sistema Tutor Multi-Agentes no Domínio da Lógica

Dissertação submetida à Universidade Federal de Santa Catarina
como requisito parcial à obtenção do grau de

Mestre em Engenharia Elétrica

por

Elisa Flemming

Florianópolis, abril de 1998

Um Sistema Tutor Multi-Agentes no Domínio da Lógica

Elisa Flemming

Esta dissertação foi julgada para a obtenção do título de **Mestre em Engenharia**, modalidade **Engenharia Elétrica**, área de concentração **Sistemas de Controle, Automação e Informática Industrial**, e aprovada em sua forma final pelo curso de Pós-Graduação da Universidade Federal de Santa Catarina.

Florianópolis, 29 de abril de 1998.

Prof. Guilherme Bittencourt
Orientador

Prof. Adroaldo Raizer
Coordenador do Curso de Pós-Graduação

BANCA EXAMINADORA

Prof. Guilherme Bittencourt
Orientador

Evandro de Barros Costa

Profª Sandra Aparecida Sandri

Um sintoma da morte de nossos sonhos são nossas certezas. Porque não queremos olhar a vida como uma grande aventura a ser vivida, passamos a nos julgar sábios no pouco que pedimos da existência. E não percebemos a imensa Alegria que está no coração de quem está lutando.
O caminho da Sabedoria é não ter medo de errar!

(Paulo Coelho)

Aos meus pais e meu
marido Renato.

AGRADECIMENTOS

Ao meu orientador Guilherme Bittencourt que, sempre presente e sorrindo, guiou meus estudos com companheirismo e dedicação.

Ao Evandro de Barros Costa que, mesmo longe, contribuiu com suas opiniões e sugestões. À professora Sandra Sandri pelos comentários e sugestões apresentados quando da análise do documento parcial.

À CAPES pelo apoio financeiro e ao LCMI pela estrutura e oportunidade oferecidas. Em especial aos professores Edson de Pieri e Eugênio Castelan que sempre se mostraram prontos a ajudar.

Ao meu pai, irmãos e sogros pelo apoio e compreensão. Em especial à Diva, minha mãe, que sempre serviu como exemplo, e me incentiva na procura do meu aperfeiçoamento profissional. Ao Renato, meu marido que, mesmo nas horas mais difíceis, jamais deixou de dar seu amparo e de demonstrar o seu amor.

Aos amigos de todas as horas, que compreenderam as ausências e sempre estiveram ao meu lado para ajudar com o que fosse necessário.

À Deus e todos aqueles que, apesar de não estarem fisicamente presentes, nunca me abandonaram...

Resumo

O presente trabalho apresenta um sistema tutor multi-agentes no domínio da lógica. Do ponto de vista conceitual o sistema tutor proposto é baseado no MATHEMA, que é um sistema que modela um ambiente interativo de ensino e aprendizagem baseado numa arquitetura multi-agentes. No domínio em questão, o interesse está no envolvimento de aprendizes humanos em resolução de problemas, auxiliados por uma sociedade de agentes computacionais. Cada um dos agentes possui conhecimentos limitados, mas é capaz de apresentar um comportamento cooperativo em relação aos demais agentes, de forma que toda a sociedade coopere na instrução do aprendiz. Um protótipo do sistema foi implementado em Common Lisp, utilizando o sistema FASE para suportar os sistemas especialistas associados a cada agente.

Abstract

This work presents a multi-agent tutor system in the domain of logic. Conceptually, the proposed tutor system is based on the MATHEMA system, an environment to model learning/teaching activities through a multi-agent architecture. In the environment, the human apprentice performs problem solving tasks, with the help of a society of computational agents. Each agent has a limited amount of domain knowledge and problem solving skills, but it is able to behave in a cooperative way with respect to the other agents. In such a way that all the agents in the society cooperate in the apprentice instruction. A prototype of the system was implemented in Common Lisp, using the FASE system to support the expert-systems that provide the cognitive capabilities to the computational agents.

Sumário

Lista de Figuras

Lista de Tabelas

CAPÍTULO 1
Introdução

À medida que a tecnologia vem se desenvolvendo ao longo dos tempos, mudanças radicais em vários setores da sociedade são cada vez mais notórias. Os meios de produção e de serviço estão passando por profundas mudanças, caracterizadas por uma supervalorização do conhecimento [DRU93].

Em 1937, Alan Turing publicava idéias sobre algoritmos e John von Neumann fazia planos para uma nova máquina de calcular que pela primeira vez realizaria, além de cálculos matemáticos, o processamento lógico de informações. Foram destas idéias e planos que nasceu o computador moderno. Alan Turing e sua equipe construíram em 1940, para o serviço de inteligência britânico, um dos primeiros computadores operacionais. Era chamado de *Heath Robinson*, utilizava tecnologia de relés e foi utilizado pelos ingleses na Segunda Grande Guerra [BIT96].

A tecnologia dos computadores, com o decorrer do tempo, vai sendo desenvolvida e assim, o computador passa a ser comercializado. Depois da Segunda Grande Guerra, iniciou-se a transferência da tecnologia dos computadores dos laboratórios militares para a sociedade civil, e desta forma caracterizou-se a disseminação da eletrônica. O avanço tecnológico trouxe uma queda nos preços das calculadoras, computadores e outras

tecnologias de informação e comunicação, e com isto vários setores da sociedade passaram a ter acesso a estas tecnologias, que hoje invadem nosso dia-a-dia.

Nos tempos atuais, não se consegue mais imaginar a sociedade sem a presença dos computadores, isto porque eles penetraram nas indústrias, no comércio, na área financeira, na educação e até mesmo nas residências das pessoas. A comunicação, por sua vez, se dá de uma forma bem mais rápida e ágil, se compararmos com os tempos anteriores, e desta forma o acesso às informações também se torna mais fácil. Um exemplo disto é a criação da INTERNET (rede mundial de computadores) que permite a veiculação de informações através de vários países do mundo. Vive-se hoje o que se chama de Revolução da Informação ou Revolução da Automação [FER94].

Essas mudanças implicam em uma alteração de postura dos profissionais em geral e, portanto, requer o repensar dos processos educacionais. Organismos e comissões internacionais no domínio da educação vêm reconhecendo a necessidade de novos objetivos que respondam à transição de uma sociedade predominantemente industrial para outra que tem como característica principal a informação. A educação não pode mais ser baseada na instrução que o professor passa ao aluno, mas na construção do conhecimento pelo aluno e no desenvolvimento de competências como aprender a buscar a informação, compreendê-la e saber utilizá-la na resolução de problemas [VAL95].

O grande desafio hoje é aliar a tecnologia computacional com o ensino em si, de forma que se utilize esta tecnologia com o objetivo de melhorar o processo ensino-aprendizagem, e não simplesmente com o objetivo de inserir uma técnica de ensino alternativa.

Estes objetivos podem ser alcançados, através da sub-área de informática na educação, influenciada pelas áreas de Inteligência Artificial, Psicologia Cognitiva e Educação, culminando com o desenvolvimento de *software* educacional [KEA87]. Nesse sentido, na década de 70, surgiram os denominados Sistemas Tutores Inteligentes (STI's) que têm se mostrado interessantes quando utilizados no ensino [BAR82] e [WEG87], visto que

buscam, a partir da simulação das capacidades cognitivas do aluno, basear suas decisões pedagógicas de forma individualizada, e que atenda às necessidades de cada usuário.

A questão central e desafiadora na pesquisa em STI's diz respeito a esta adaptabilidade do sistema às necessidades individuais de cada aprendiz, visto que, cada um deles possui características próprias e deve ser modelado de maneira singular. Um problema que surge neste contexto é a complexidade do conhecimento que o sistema deveria possuir para poder promover esta interação individualizada, dentro de uma perspectiva de aprendizagem cooperativa.

Atualmente, a área de STI começa a incorporar técnicas da Inteligência Artificial Distribuída [DLC89], [DR94], através de uma abordagem de Sistemas Multi-agentes [JEN93],[JS92], na concepção de seus sistemas [VIG96],[COT97].

Este trabalho situa-se dentro da área de informática na educação, mais precisamente no desenvolvimento de sistemas tutores inteligentes, e tem como objetivo o desenvolvimento de um sistema tutor multi-agentes orientado ao domínio da lógica. Do ponto de vista conceitual, tomou-se por base, para a definição do sistema, o modelo teórico de ambiente interativo de aprendizagem com um enfoque multi-agentes denominado MATHEMA [COT97].

O MATHEMA é um ambiente integrado proposto com a finalidade de contribuir, principalmente, em dois aspectos problemáticos na construção de um STI: definição de um modelo do conhecimento do domínio que se preocupa com aspectos ligados à sua computabilidade e uma modelagem do estudante a partir da solução multi-agentes, apresentando uma alternativa ao diagnóstico distribuído.

Para definir o sistema tutor em lógica foi necessário, primeiramente, um trabalho de adequação ao modelo MATHEMA, considerando as peculiaridades do domínio em questão. Em seguida, foram tratadas as questões relacionadas às interações entre os agentes tutores, e entre estes e um aprendiz humano.

Para implementar o sistema tutor multi-agentes foram utilizados a linguagem de programação Common Lisp [SJ84] e o sistema FASE [BM93], que é uma ferramenta para construção de arcabouços para sistemas especialistas. O modelo MATHEMA é capaz de suportar uma rede de agentes tutores e especialistas humanos podendo fornecer serviços a distância para um aprendiz. A capacidade cognitiva de cada agente da sociedade está associada a um conjunto de bases de conhecimento, consistindo em fatos representados sob a forma de *frames* e regras simbólicas, associadas aos diferentes tipos de conhecimento: sobre o domínio e sobre as estratégias de ensino e de cooperação.

A dissertação está organizada em sete capítulos, sendo que cada capítulo constitui uma parte essencial do estudo feito para a sua elaboração.

O Capítulo 2 apresenta algumas questões referentes ao uso da informática na educação, procurando gerar reflexões sobre o processo ensino-aprendizagem quando o computador também é parte integrante deste.

O Capítulo 3 trata das noções fundamentais de suporte ao entendimento do sistema tutor multi-agentes proposto. Apresenta noções à respeito da Inteligência Artificial, apresentando os sistemas especialistas e o sistema FASE utilizado na implementação, e por fim, discute alguns aspectos da Inteligência Artificial Distribuída, enfatizando a abordagem de Sistemas Multi-Agentes.

O Capítulo 4 fornece uma visão geral à respeito dos Sistemas Tutores Inteligentes partindo da definição de suas características gerais, passando por um breve histórico e por fim apresentando uma arquitetura geral de um STI.

O Capítulo 5 descreve o ambiente MATHEMA, levando-se em consideração principalmente os aspectos que interessam a este trabalho. Parte-se da definição da arquitetura geral do ambiente, discute-se o modelo do domínio adotado e por fim apresenta-se a arquitetura dos agentes tutores.

No Capítulo 6 descreve-se o Sistema Tutor Multi-Agentes desenvolvido no domínio da lógica. Para tal, apresenta-se o modelo do domínio utilizado, a arquitetura geral dos agentes e a descrição do funcionamento das interações entre os agentes na sociedade.

O Capítulo 7 apresenta as conclusões finais e as perspectivas de trabalhos futuros.

No apêndice tem-se algumas definições que dizem respeito ao domínio de estudo escolhido para a implementação do sistema tutor: a lógica clássica. Desta forma, não haverão formalismos durante a apresentação do domínio, visto que o mesmo estará contemplado no apêndice ao final deste documento.

CAPÍTULO 2
Informática na Educação

Neste capítulo são discutidos alguns pontos a respeito do uso da informática na educação. Para tal, inicialmente considera-se o processo de inserção do computador na escola como uma reflexão, no sentido em que a máquina por si só não causa mudanças no processo ensino-aprendizagem. Ao contrário, as mudanças surgem a partir da reflexão e da posterior transformação do ambiente de aprendizagem, de forma que o aluno tenha oportunidade de vivenciar e desenvolver as habilidades exigidas pela atual sociedade de conhecimento.

Em seguida, são discutidas as visões à respeito do processo de ensino-aprendizagem e da aquisição do conhecimento, levando-se em conta a presença do computador dentro deste processo.

Por fim destacam-se algumas modalidades existentes de *software* educacional, apontando suas características básicas, bem como alguns exemplos possíveis de uso.

2.1. Considerações Iniciais

Nos anos 70, o ensino apoiado por computadores prometia mudanças profundas nas formas de se conceber e administrar o processo de ensino-aprendizagem [SAN93]. Estas

promessas levaram a um repensar dos conceitos da aprendizagem e provocaram uma revolução quando se descobre que o computador pode ser usado para educar, criando uma nova *mídia educacional* [VAL95].

Balacheff [BAL94] coloca que uma das principais razões por trás da introdução dos computadores em sala de aula foi a possibilidade de os alunos poderem interagir e aprender com o *software*, quase que independentemente dos professores. É necessário, de qualquer forma, que se tenha claro que esta nova mídia educacional não substitui o professor, mas sim possibilita uma mudança dos papéis deste e do seu aluno. Nesse caso, o professor mudaria, de repassador de conhecimento, para criador de ambientes de aprendizagem e facilitador do processo de aquisição de conhecimento, e o aluno, de mero memorizador passivo de conhecimento, pode tornar-se agente da aprendizagem, através da procura e uso do conhecimento de forma independente.

Valente [VAL93] diz que para a implantação do computador na educação são necessários basicamente quatro elementos: o computador, o *software* educacional, o professor capacitado para usar o computador como mídia educacional e o aluno.

Um *software* educacional é um programa que visa atender necessidades definidas no processo ensino-aprendizagem e possui objetivos pedagógicos. Todo o *software* que pode ser inserido num contexto e numa situação de ensino-aprendizagem, na qual exista uma metodologia orientadora do processo, pode ser considerado educacional [VIG96].

A tarefa do professor é de ajudar a desenvolver o pensamento do aluno, sua capacidade de analisar e generalizar os fenômenos da realidade. O professor não pode se limitar a organizar ações previstas para a execução destas tarefas, mas sim deve propiciar aos alunos os métodos necessários, que tenham características pedagógicas e levem ao aparecimento de atividades inovadoras.

Dentro deste contexto, é preciso que o professor seja sensibilizado para o uso crítico da informática na educação, de forma que não se torne apenas um mero repetidor de

experiências ou operador de *software* educativo [MOR86]. É fundamental que os professores tenham consciência da mudança que a inserção do computador causa no processo ensino-aprendizagem, para que possam criar atividades inovadoras que tenham uma base pedagógica e tecnológica, partindo para uma etapa de repensar o *ato de ensinar*.

Como foi comentado inicialmente, o uso dos computadores na educação mostra que desde o seu aparecimento, já se reconhecia seu enorme potencial para o uso educacional. Alguns pesquisadores reagiam com entusiasmo, outros com apreensão, e as primeiras tentativas de criação de *software* educacional tiveram resultados diversos: alguns projetos revelavam problemas com relação ao desempenho das máquinas e outros apontavam promessas pedagógicas magníficas [WEG87].

A concepção de um ambiente de aprendizagem assistido por computador envolve, direta ou indiretamente, uma visão sobre o processo ensino-aprendizagem. É preciso que haja uma reflexão neste sentido desde o início da concepção, quando o *software* ainda se encontra em desenvolvimento.

Não é mais razoável se admitir, nos dias de hoje, um *software* educacional que use diversos recursos tecnológicos como som, animação, cores sem que contemple pontos como conteúdo e prática pedagógica.

Isto se justifica quando se pensa que, dependendo do tipo de *software* usado, bem como do envolvimento do professor na interação aluno-computador, a compreensão do aluno após a utilização do software pode variar, ou até mesmo não acontecer. Por outro lado, existe *software* que faz com que o aluno busque informações, consiga processá-las e utilizá-las na resolução de problemas, permitindo a compreensão do que faz e a construção do seu próprio conhecimento.

Piaget [PIA78] faz uma distinção entre o *fazer* e o *compreender*, colocando que a criança é capaz de realizar ações complexas e ter um êxito precoce sem, no entanto, ter a compreensão conceitualizada daquilo que realizou. Ela pode saber fazer, obter sucesso mas

não compreender os conceitos envolvidos na atividade que realizou. A passagem do saber fazer para o compreender se dá devido à tomada de consciência, que consiste na transformação do esquema das ações – que permitem o fazer, em noções e operações – que constituem a conceitualização.

Conclui-se então que um determinado *software* não pode ser analisado independentemente do seu uso, visto que não é este que permite ao aluno entender ou não um determinado conteúdo. A compreensão é fruto de como o *software* é utilizado e de como o aluno está sendo desafiado na atividade de usá-lo.

Outra questão interessante é que a ação do professor também é diferenciada de acordo com o *software* utilizado. Em alguns têm-se características que facilitam a atuação do professor na interação aluno-computador, permitindo ao professor entender mais facilmente o que o aluno está pensando ou fazendo. Os que não possuem estas características exigem uma atuação maior do professor no sentido de conseguir com que o aluno passe do nível do fazer para o compreender.

2.2. Visões sobre Natureza do Conhecimento e Aprendizagem

Existem vários modelos com propostas para lidar com a questão da aprendizagem. Neste sentido, Moreira [MOR86] apresenta três visões sobre educação, ou mais especificamente sobre a natureza do conhecimento e da aprendizagem: visão empirista, visão racionalista e visão construtivista. A seguir apresentam-se algumas características para estas visões.

2.2.1. Visão Empirista

A visão empirista baseia-se na corrente psicológica que defende a idéia de que todo conhecimento provém da experiência. Em virtude de sua base epistemológica, a visão empirista forma o corpo do que se chama *Associacionismo*, cuja expressão mais importante

é o behaviorismo. Na perspectiva behaviorista, a Psicologia é definida como a "ciência do comportamento", passível de tratamento empírico, e o comportamento entendido como produto das pressões do ambiente, significando o conjunto de reações que podem ser medidas, previstas e controladas.

Seguindo este raciocínio, a aprendizagem passa a ser considerada a "mudança de comportamento" resultante do treino ou da experiência, e o conhecimento é uma cadeia de idéias atomisticamente formada a partir do registro dos fatos e se reduz a uma simples cópia do real.

A visão empirista foi a primeira linha de pensamento que surgiu dentro do contexto do computador no ensino, e por isto tem suas origens nas primeiras tentativas da utilização de máquinas na aprendizagem de qualquer conhecimento.

Em 1924 Dr. Sidney Pressey inventou uma máquina para corrigir testes de múltipla escolha, e em 1950 Skinner, professor de psicologia de Harvard, utilizando-se desta idéia propôs uma máquina para ensinar usando o conceito de *instrução programada*.

Este conceito de instrução programada consiste em dividir o material a ser ensinado em pequenas unidades, logicamente encadeadas, chamadas módulos. Cada módulo termina com um teste que o aluno deve responder preenchendo espaços em branco ou escolhendo a resposta certa entre algumas alternativas apresentadas. O aluno lê o fato ou o conceito e é imediatamente questionado, se a resposta está correta, passa para outro módulo, senão a resposta certa é fornecida e o aluno é convidado a rever módulos anteriores ou, ainda, a realizar outros módulos que sirvam como pré-requisito.

A instrução programada na forma impressa foi muito utilizada durante o final de 1950 e início dos anos 60. Entretanto havia muita dificuldade na produção do material instrucional e os materiais existentes não possuíam nenhuma padronização, dificultando sua disseminação.

Com o surgimento do computador, notou-se que os módulos do material instrucional poderiam ser apresentados com uma maior flexibilidade se fossem implementados no computador. Desta forma, diversos programas de instrução programada foram implementados e surgiram as instruções programadas por computador: *CAI - Computer-Aided Instruction.*

Nestes sistemas, o computador funcionava simplesmente como um dispositivo de apresentação do material preparado pelo professor. Em resumo, os sistemas CAI, também conhecidos como *AFO – Ad-hoc Frame Oriented*, não eram mais do que uma versão eletrônica dos livros clássicos de aprendizagem, e por isto conhecidos como *viradores de páginas eletrônicos.*

2.2.2. Visão Racionalista

Esta nova visão decorre do surgimento da *Gestalt* que é uma corrente psicológica que vai contra a visão empirista porque se opõe ao behaviorismo. A Gestalt pressupõe que todo conhecimento é anterior à experiência, e é fruto do exercício de estruturas racionais, pré-formadas no sujeito. O homem ao nascer, já apresentaria todas as possibilidades do conhecimento definidas por leis inerentes à razão humana.

Os defensores desta linha não se preocupam com a aprendizagem já que a experiência não contribui para a estruturação do conhecimento e a aprendizagem se reduz à solução de problemas ou *insights.*

Em termos da produção de *software* educacional, esta visão não produziu práticas nem efeitos significantemente diferentes dos apresentados na visão empirista.

2.2.3. Visão Construtivista

As duas visões anteriores resultam num silenciamento dos alunos, isolando-os e submetendo-os à autoridade do saber dos professores, dos textos, dos livros e das instruções

programadas. O resultado disto é o falso conhecimento e a subordinação do aluno aos esquemas dos professores [GIU85].

Na visão construtivista o problema da aprendizagem tem como pressuposto que o conhecimento não procede nem da experiência única dos objetos, nem de uma programação inata pré-formada no sujeito, mas de construções sucessivas com elaborações constantes de estruturas novas.

A prática pedagógica da transmissão e da memorização não serve mais, e nesta perspectiva o computador passa a ser utilizado como veículo de disseminação de informações de grande porte.

É dentro do contexto da visão construtivista que se enquadra o sistema tutor inteligente proposto no âmbito deste trabalho.

2.3. Tipos de *Software* Educacional

Hoje se fala em várias categorias de *software* educacional, que são classificados de acordo com a sua utilização e com algumas características de sua programação. A literatura sobre o uso de computadores no ensino apresenta várias classificações para o *software* educacional, ver por exemplo [CAM95] , [STA90]. No entanto vários autores situam-se num contexto mais abrangente usando a classificação proposta por Taylor em 1980 [CAM96] quando coloca a utilização do computador como *Tutor*, *Ferramenta* ou *Tutelado*.

Como *tutor* o computador desempenha o papel de professor, orientando os alunos para a aquisição de um novo conhecimento, como *ferramenta* o computador é utilizado para adquirir e manipular informações e como *tutelado* os alunos ensinam o computador.

Dentro desta classificação, algumas modalidades de uso do computador têm sido consagradas. A seguir apontam-se algumas características que diferenciam os tipos de

software ditos educacionais. Vale destacar, no entanto, que existe *software* que se enquadra em mais de uma destas categorias, dependendo de sua concepção. Como exemplo tem-se os CAI's que podem ser considerados como um *software* de exercício e prática ou como um tutorial.

2.3.1. Exercício e Prática

O *software* considerado como de exercício e prática é aquele que tem como objetivo a aquisição de uma habilidade ou a aplicação de um conteúdo já conhecido pelo aluno, mas não inteiramente dominado Tipicamente é utilizado para revisar conteúdos vistos em sala de aula, principalmente os que envolvem memorização e repetição.

É a forma mais tradicional na qual os computadores vêm sendo utilizados em educação, e é considerado um *software* dos mais fáceis de serem desenvolvidos e utilizados. Tem como características uma seleção aleatória de problemas e não possui julgamento das respostas erradas, ou seja, o aluno repete o exercício quantas vezes for necessário até que acerte.

2.3.2. Jogos

São considerados uma fonte de recreação para a aquisição de um determinado tipo de aprendizagem e possuem elementos de desafio ou competição. A pedagogia por trás desta abordagem é a de exploração auto-dirigida ao invés da instrução explícita e direta [VAL93].

Segundo pesquisadores da área, os jogos podem desenvolver várias características dentre elas o uso da negociação, da persuasão, da cooperação, do respeito à inteligência do adversário, assim como a visão do todo mais do que as partes. Em suma, pode-se esperar mais motivação por parte do aluno, através de algo como aprender brincando.

2.3.3. Simulação e Modelagem

A simulação é a representação do comportamento de um objeto real, de um sistema ou evento. É um modelo simbólico e representativo da realidade que deve ser utilizado a partir da caracterização dos aspectos essenciais do fenômeno. Este tipo de programa pode ser utilizado, por exemplo, após a aprendizagem de conceitos e princípios básicos do tema em questão.

O usuário, em alguns casos, é um elemento participante que controla o sistema, isto é, possui o controle das variáveis e dos parâmetros. O sistema gerenciador, então, ajusta a simulação baseando-se nas entradas dadas pelo usuário.

A modelagem computacional, por sua vez, é um sistema dinâmico no qual cada passo do cálculo pode ser definido de maneira elementar. Os modelos são construídos passo a passo, modificando-se os modelos anteriores. Por exemplo, na modelagem dinâmica identificam-se variáveis importantes que descrevem um sistema e especifica-se a variação delas ao longo do tempo. As regras para evolução do sistema são regras para o cálculo dos próximos valores de uma variável.

Tanto na simulação quanto na modelagem o aluno é alertado com mensagens de erro, quando for o caso, que indicam o caminho correto a ser seguido. Além desta característica, sistemas deste tipo têm a capacidade de armazenamento das respostas, a fim de se conhecer a estrutura do raciocínio do aluno diante do problema dado.

2.3.4. Micromundos

Os Micromundos foram propostos, inicialmente na década de 60, por Papert e sua equipe no Massachusets Institute of Technology (MIT). O projeto LOGO [PAP85] é um exemplo consagrado desta categoria, no qual um objeto representado por uma tartaruga interage com o aluno de forma a ajudá-lo na resolução de problemas.

A característica dos Micromundos em relação a sua proposta pedagógica, diz respeito à aprendizagem pela ação, numa perspectiva de construção do conhecimento.

Outro exemplo de Micromundo é o *Cabri-Géomètre*, um caderno de rascunho interativo em geometria, desenvolvido pelo LSDD-IMAG em Grenoble [BAU90].

2.3.5. Tutorial

O software do tipo tutorial é utilizado como apoio ou reforço para aulas ou então como uma ferramenta de revisão de atividades já vistas em sala de aula. Os tutoriais têm como objetivos genéricos a introdução de conceitos novos e a pretensão de se obter a aquisição de conceitos, princípios e/ou generalizações.

Em seu desenvolvimento, procura-se implementar a emissão de mensagens de erro que conduzam o aluno a uma resposta certa, quando for o caso.

2.3.6. Tutores Inteligentes

O objetivo dos tutores inteligentes é trazer maior flexibilidade e interatividade no domínio da tutoria. Os impasses provocados pelos sistemas CAI tradicionais levaram os programadores a introduzir técnicas da Inteligência Artificial nos sistemas de aprendizagem assistidos por computador, o que provocou um deslocamento metodológico qualitativamente radical na forma de programar sistemas educacionais: da programação de decisões à declaração do conhecimento.

A idéia geral consiste em aplicar técnicas e métodos de Inteligência Artificial em sistemas CAI, surgindo assim os sistemas *Intelligent Computer Assisted Instruction (ICAI)*, cujo nome mais utilizado na literatura é *Intelligent Tutorial System (ITS)* ou *Sistemas Tutores Inteligentes (STI)*.

A tendência hoje é a evolução dos sistemas tutores inteligentes para os Ambientes Interativos/Inteligentes de Aprendizagem, *Interactive/Intelligent Learning Environment (ILE)* ou ainda Sistemas Tutores Cooperativos. Estes sistemas podem ser entendidos como uma combinação de aspectos das categorias de tutores inteligentes e micromundos.

Os ILE's assim como os tutores inteligentes passaram a incorporar modelos de trabalho cooperativo, e desta forma os ambientes passam a ser concebidos com a abordagem de agentes [COT97].

2.4. Considerações Finais

Quando se faz um acompanhamento do uso do computador na educação, nota-se que sua utilização atualmente aponta para uma nova direção de ensino. O computador vem sendo utilizado de outras formas, agora como um instrumento de complementação e de aperfeiçoamento.

Derrubou-se a idéia de substituição do professor pelo computador e segundo Wenger [WEG87], isto se deve ao fato de que o ensino como um ato social envolve muitas outras dimensões além do processamento de informações para o qual os computadores são usados.

Neste sentido, o *software* educacional não pode mais ser concebido para se tornar "virador de páginas", ao contrário, passa a ser um elemento mais ativo no processo de interação com o aluno. Assim os sistemas tutores inteligentes possuem um grande potencial e por este motivo têm sido objeto de pesquisas constantes.

Por fim vale destacar a importância de se fazer uma reflexão a respeito do processo ensino-aprendizagem antes de se conceber um *software* educacional. Quando existe esta preocupação, busca-se levar em consideração os aspectos inerentes a este processo e que não podem ser desprezados quando se insere o computador no ensino.

Este trabalho trata do desenvolvimento de um sistema tutor inteligente, baseado no ambiente MATHEMA [COT97], que procura levar em consideração, quando da sua concepção, as características acima citadas.

CAPÍTULO 3
Inteligência Artificial

Neste capítulo são apresentadas considerações a respeito da Inteligência Artificial (IA), um ramo da ciência da computação que tem como objetivo central a criação de teorias e modelos para a capacidade cognitiva e a implementação de sistemas computacionais baseados nestes modelos.

Inicialmente, discute-se a respeito das linhas principais de pesquisa na área de IA, e em seguida, apresenta-se os Sistemas Especialistas (SE's). Descreve-se uma ferramenta que permite a construção de arcabouços de sistemas especialistas (o sistema FASE) e por fim tem-se algumas noções em Inteligência Artificial Distribuída (IAD).

Em particular, quando se trata da IAD será enfatizada uma abordagem de Sistemas Multi-Agentes (SMA) que será discutida de forma genérica, porém levando em conta os conceitos principais para que se possa compreender sua utilização no contexto deste trabalho.

3.1. Características Gerais da IA

Em 1956, durante uma conferência de verão em Dartmouth College, NH, USA, aparece pela primeira vez uma menção oficial à expressão *Inteligência Artificial*. Na proposta desta conferência, escrita por John McCarthy (Dartmouth), Marvin Minsky

(Hardward), Nathaniel Rochester (IBM) e Claude Shannon (Bell Laboratories), aparece a intenção de se estudar o tópico *inteligência artificial* [MCC79].

No entanto, nesta época o termo Inteligência Artificial (IA) gerou bastante polêmica, visto que, quando se tentava chegar a uma definição formal precisa para IA, era necessário definir a própria inteligência [BIT96].

De qualquer forma, chegou-se a algumas definições consideradas operacionais: *Uma máquina é inteligente se ela é capaz de solucionar uma classe de problemas que requerem inteligência para serem solucionados por seres humanos* [MH69] ou ainda *Inteligência Artificial é o estudo das faculdades mentais através do uso de modelos computacionais* [CM85].

Existem basicamente duas linhas de pesquisa destinadas à construção de sistemas inteligentes: a *Conexionista* e a *Simbólica*.

O objetivo da linha conexionista é o de modelar a inteligência humana através da simulação dos neurônios e de suas interligações. A linha conexionista deu origem à área de redes neurais ou redes neuronais artificiais.

A linha simbólica segue a tradição lógica e teve como defensores principais McCarthy e Newell [NEW80]. Inicialmente as pesquisas na área simbólica tinham como objetivo simular a inteligência humana em geral, e se concentravam no desenvolvimento de formalismos gerais capazes de resolver qualquer tipo de problema.

Como se trabalhava com sistemas gerais, tornou-se inviável a extensão para domínios de problemas reais, visto que não se imaginou que problemas reais tinham uma complexidade tão grande, e que freqüentemente o conhecimento disponível sobre o mundo real é incompleto e parcialmente incoerente.

Pelos problemas citados, os pesquisadores passaram a trabalhar com a IA buscando simular a inteligência humana em situações pré-determinadas e se concentravam em desenvolver formalismos de representação de conhecimento adaptados a cada tipo de problema que estivesse sendo analisado.

Na década de 70 a IA estava praticamente restrita ao ambiente acadêmico e os programas eram idealizados sem aplicações práticas ou fins comerciais. A década de 80 foi marcada pela disseminação dos *Sistemas Especialistas* (SE), que indicavam a primeira oportunidade real de comercialização de produtos da IA. Surgem também as *Ferramentas para a Construção de Sistemas Especialistas* ou os *Arcabouços de Sistemas Especialistas* (ASE).

No contexto deste trabalho, utilizou-se para a implementação do Sistema Tutor Inteligente proposto o sistema FASE, que é um ASE. Nas próximas seções serão apresentados, de forma geral, os Sistemas Especialistas e o sistema FASE.

3.2. Sistemas Especialistas

Sistemas Especialistas (SE's) são programas de computador que reproduzem o comportamento de especialistas humanos na solução de problemas do mundo real. Os primeiros SE's desenvolvidos com sucesso foram o DENDRAL [FBL71] e o MYCIN [SHO76]. A partir destes, os SE's passaram a ser desenvolvidos em vários domínios de conhecimento, por exemplo agricultura, engenharia, geologia, direito, matemática, medicina, aplicações militares, física, dentre outros.

A IA está num SE basicamente na forma como é representado o conhecimento sobre o domínio, ou seja, o comportamento do especialista ao resolver um problema. A parte mais difícil do desenvolvimento de um SE é exatamente a *aquisição de conhecimento*.

Os SE's utilizam o formato de regras de produção, expressões que associam a cada condição uma ação, e apresentam uma arquitetura com base em três módulos (ver figura 3.1):

- Base de Regras;
- Memória de Trabalho;
- Motor de Inferência.

A base de regras possui condições que representam *perguntas* à representação de conhecimento armazenado na memória de trabalho, que por sua vez pode conter qualquer tipo de estrutura de dados. Porém mais do que estruturas de dados, a memória de trabalho deve respeitar um método de representação de conhecimento, que é uma linguagem formal com uma descrição matemática de seu significado. Por exemplo a lógica de primeira ordem é um formalismo de representação de conhecimento bastante utilizado.

A base de regras e a memória de trabalho formam a base de conhecimento, módulo no qual se armazena todo o conhecimento sobre o domínio específico de cada SE.

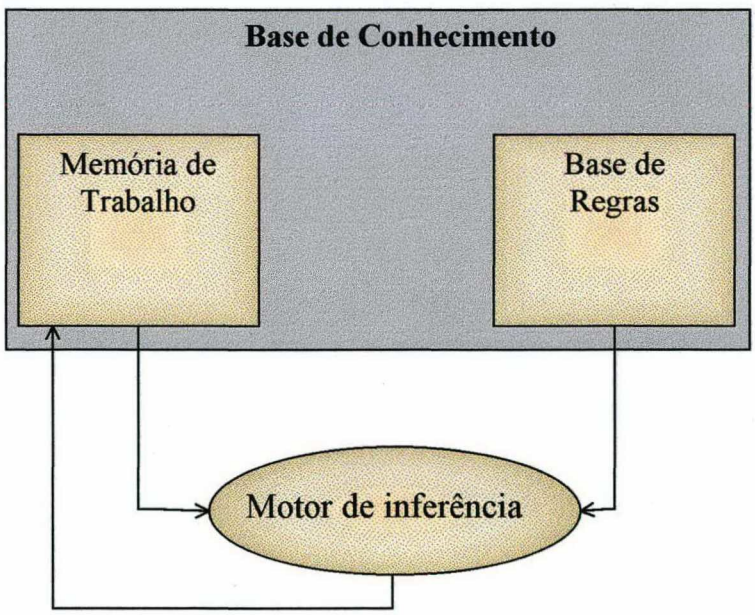


Figura 3.1: Arquitetura de um Sistema Especialista.

O motor de inferência é o módulo responsável pelo mecanismo de controle do sistema, e tem como objetivo avaliar e aplicar as regras de acordo com as informações contidas na memória de trabalho.

Uma das características que deve estar contida num SE é a existência de um mecanismo de raciocínio incerto que permita representar a incerteza a respeito do conhecimento do domínio, já que o conhecimento real é incerto.

Além disso, o conhecimento armazenado nas regras e na memória de trabalho deve ser obtido com pelo menos um especialista humano do domínio e deve ser representado de acordo com regras formais definidas para a codificação da base de regras. Sabe-se que o desempenho de um SE melhora na medida em que se consegue armazenar um maior conhecimento de uma melhor forma possível, ou seja, de forma que se tenha características ao mais próximo possível do conhecimento real.

Os SE's são desenvolvidos, na maioria das vezes, a partir de *Arcabouços de Sistemas Especialistas* (ASE's). Um ASE é uma ferramenta que suporta todas as funcionalidades de um SE, de forma que a função do programador é a de codificar o conhecimento especializado de acordo com a linguagem de representação de conhecimento escolhida.

3.3. O Sistema FASE

O sistema FASE (*Ferramenta para a construção de Arcabouços de Sistemas Especialistas*) [BM93], [BIT95] é uma ferramenta para a geração de arcabouços de sistemas especialistas baseados em regras. Foi implementado na linguagem de programação Common Lisp [SJ84] e os programas fonte são de domínio público.

O objetivo do sistema é integrar em uma única ferramenta diversas funcionalidades para representação de conhecimento, estratégias de controle, técnicas de resolução de

conflito e de raciocínio incerto. Estas funcionalidades podem ser combinadas para a construção de arcabouços de sistemas especialistas de diferentes níveis de complexidade.

A arquitetura do FASE pode ser representada como mostra a figura 3.2. O ambiente é modular, constituído por uma biblioteca de funções LISP dividida em 4 módulos, de acordo com as funcionalidades citadas anteriormente. Existe ainda um outro módulo chamado manipulador de regras, ao qual podem ser integrados os recursos disponíveis nos módulos da biblioteca de funções.

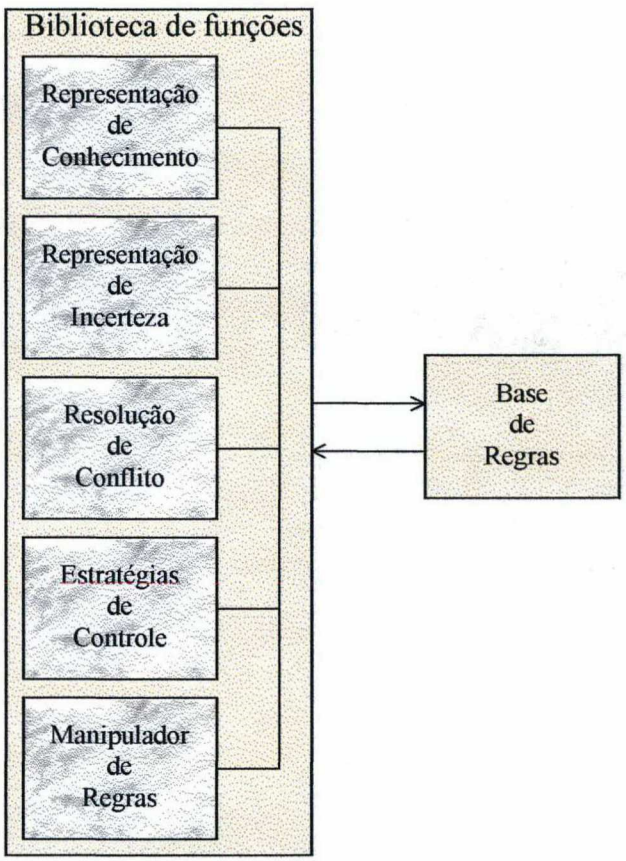


Figura 3.2: Arquitetura Geral do FASE.

Tem-se a seguir uma descrição de cada um dos módulos que formam a biblioteca de funções do FASE.

3.3.1. Manipulador de Regras

O formalismo implementado neste módulo define as interfaces entre ele e os demais módulos. A modularidade do ambiente decorre dos conceitos de *Padrão* e *Instância*.

Um padrão, do inglês *pattern*, é uma representação genérica de um conjunto de fragmentos de conhecimento que geralmente envolve variáveis.

Uma instância, por sua vez, é uma representação específica de um fragmento de conhecimento. A instância contém uma substituição de variáveis, o tempo de criação do fragmento de conhecimento que a gerou, um ou mais números representando medidas de incerteza, informação sobre a origem do fragmento de conhecimento e, eventualmente, outros atributos que sejam necessários a representações de conhecimento específicas.

Intuitivamente um padrão pode ser interpretado como uma "pergunta" à memória de trabalho e uma instância como uma "resposta" a esta pergunta. Além disso a combinação entre um padrão e uma instância é interpretada como um novo fragmento de conhecimento, no qual as variáveis presentes são substituídas de acordo com a substituição presente na instância.

As regras utilizadas possuem a seguinte sintaxe:

```
<rule> ::= ( <name> <alpha>   ({<pattern>})
                              ({<pattern>}))
```

onde: <name> é um símbolo correspondente ao nome da regra
<alpha> é o valor numérico associado ao coeficiente de certeza da regra
<pattern> é a sintaxe dos padrões que irá depender do método de representação de conhecimento utilizado.

3.3.2. Estratégias de Controle

Existem disponíveis no FASE dois tipos de estratégias de controle para encadeamento de regras: progressiva e regressiva.

Na estratégia progressiva, também chamada *encadeamento à frente* ou *dirigido por dados*, a parte esquerda da regra é comparada com a descrição da situação atual, contida na memória de trabalho. As regras que satisfazem a esta descrição têm sua parte direita executada, o que em geral significa a introdução de novos fatos na memória de trabalho.

Na estratégia regressiva, também dita *encadeamento para trás* ou *dirigido por objetivos*, o comportamento do sistema é controlado por uma lista de objetivos. Um objetivo pode ser satisfeito diretamente por um elemento da memória de trabalho, ou podem existir regras que permitam inferir algum dos objetivos correntes, isto é, contenham uma descrição deste objetivo em suas partes direitas. As regras que satisfazem esta condição têm suas partes esquerdas adicionadas à lista de objetivos correntes.

3.3.3. Resolução de Conflitos

Após a definição da estratégia de controle, o motor de inferência inicia o processo de busca que o guia na pesquisa à memória de trabalho e à base de regras. Ao término deste processo, dispõe de um conjunto de regras chamado *Conjunto de Conflito*. Caso este conjunto seja vazio, a execução é terminada, caso contrário serão necessário métodos que permitem escolher quais regras serão executadas e em que ordem.

O FASE dispõe dos seguintes métodos: ordem lexicográfica sobre os nomes das regras; regra mais recente e regra mais específica.

Pode-se utilizar mais de um dos métodos citados para resolver os conflitos, desde que exista uma relação de precedência entre eles.

3.3.4. Representação de Incerteza

O tratamento da incerteza é uma área que vem sendo bastante pesquisada em SE, visto que os domínios adequados à implementação se caracterizam por não serem modelados por uma teoria geral, o que implica em descrições incompletas, inexatas ou incertas. Os métodos para tratar este problema, de maneira geral, atribuem aos fatos e regras uma medida numérica que represente de alguma forma a confiança do especialista.

No FASE estão disponíveis os métodos do fator de certeza (conforme o modelo adotado no MYCIN) [SHO76] e a teoria de possibilidades [DP88].

3.3.5. Representação do Conhecimento

O módulo de representação de conhecimento consiste em uma base de conhecimento interna e uma linguagem de acesso que implementa um formalismo de representação de conhecimento.

Atualmente existem três linguagens de representação de conhecimento disponíveis no ambiente: lógica, redes semânticas e quadros. Estas linguagens foram escolhidas por serem as mais utilizadas para o desenvolvimento de SE. Cada linguagem de representação de conhecimento forma um módulo independente, que pode ser utilizado isoladamente para o desenvolvimento de sistemas baseados em conhecimento que não utilizem o paradigma de regras de produção.

A seguir tem-se uma descrição das linguagens de representação de conhecimento disponíveis no ambiente, bem como exemplos de sua utilização no FASE.

(a) Lógica

A lógica é uma das linguagens de representação de conhecimento mais utilizadas em SE, e as duas principais vantagens de sua utilização são: a expressividade inerente da linguagem e uma semântica bem definida e bastante estudada. Uma base de conhecimento

que utiliza este tipo de representação consiste em um conjunto de fórmulas lógicas que podem ser representadas, por exemplo, em expressões LISP.

A utilização da lógica é interessante quando se tem o conhecimento como uma coleção de fatos independentes e bastante desestruturados. O método de inferência pode ser uma simples unificação ou dedução automática.

Exemplo: Deseja-se representar as condições de funcionamento de um motor de automóvel a partir dos seguintes indicadores:

- ⇨ Alto consumo de combustível.
- ⇨ Baixo rendimento do motor.
- ⇨ Baixo consumo de óleo do cárter.
- ⇨ Viscosidade do óleo do cárter moderada.
- ⇨ Não apresenta centelhamento no escape.
- ⇨ Não apresenta ruído no motor.
- ⇨ A taxa de compressão do motor é moderada.
- ⇨ Não apresenta odor de combustível no escapamento.
- ⇨ Tomada de aceleração é normal.
- ⇨ Partida rápida.

A informação desejada assume a seguinte sintaxe, no FASE, quando se utiliza a lógica como linguagem de representação de conhecimento:

```
((logic (consumo_de_combustivel alto)
        (rendimento_do_motor baixo)
        (consumo_de_oleo_do_carter baixo)
        (viscosidade_do_oleo_carter moderada)
        (centelha_no_escape negativo)
        (ruido_no_motor negativo)
        (taxa_de_compressao_do_motor moderada)
```

```
(odor_de_combustivel_no_escape negativo)
(tomada_de_aceleraçao normal)
(partida_do_motor rapida)))
```

(b) Redes Semânticas

Uma rede semântica é um grafo orientado cujos nodos podem ser utilizados para representar conceitos primitivos, predicados, objetos, etc, e os arcos são utilizados para associar os nodos com relações binárias. Os arcos podem ser positivos ou negativos permitindo assim a representação explícita de exceções.

O mecanismo de inferência básico é a herança através dos arcos do grafo. Esta característica permite que propriedades de um nó sejam especificadas apenas uma vez, no mais alto nível de uma hierarquia de conceitos, sendo herdadas por todos os conceitos derivados, implicando um economia substancial de memória.

Exemplo: Seja a seguinte informação: "Clyde é um elefante real, a cor dos elefantes é cinza, no entanto os elefantes reais não são cinza". A figura 3.3 representa esta informação utilizando redes semânticas. A sintaxe utilizada no FASE é apresentada abaixo.

```
(snet (cor      (clyde) (real) (elefante) (cinza)
                (clyde real e-um)
                (real elefante e-um)
                (elefante cinza e-um)
                (real cinza (not . e-um))))
```

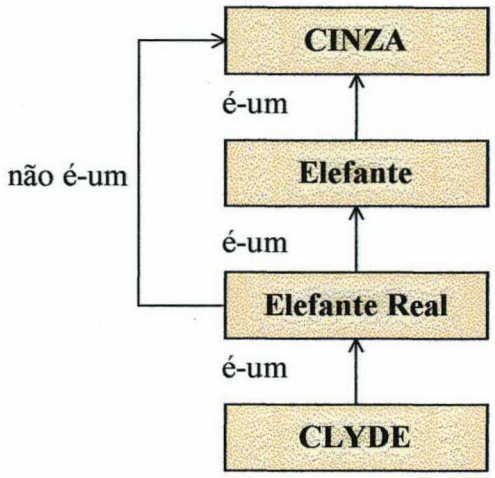


Figura 3.3: Rede semântica referente ao exemplo do elefante Clyde.

(c) Quadros

Os quadros (do inglês *frames*) foram introduzidos para permitir a expressão das estruturas internas dos objetos, mantendo a possibilidade de representar herança de propriedades, como fazem as redes semânticas.

Um quadro consiste em um conjunto de atributos, sendo que um valor pode ser associado a cada atributo. Este valor pode ser um ponteiro para um outro quadro ou qualquer informação primitiva sobre o conceito representado pelo quadro.

Existem três mecanismos de inferência integrados no formalismo: herança de valores de atributos através da hierarquia, valores previamente escolhidos podem ser utilizados quando não há informação disponível e ligação procedural para permitir a execução de uma função externa ao formalismo.

Exemplo: Deseja-se representar imóveis utilizando-se quadros. A figura 3.4 representa a estrutura do imóvel.

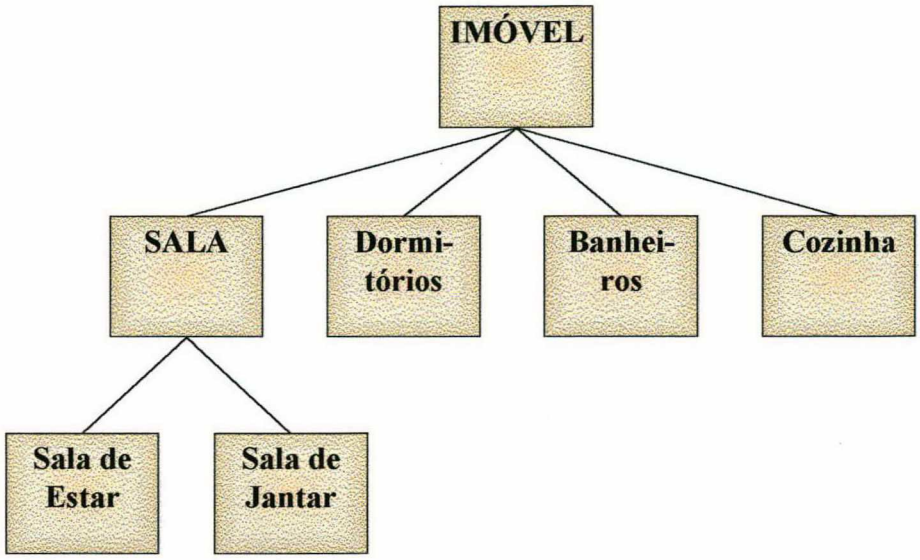


Figura 3.4: Estrutura de quadros para imóveis

A sintaxe utilizada no FASE seria a seguinte:

```
((frame    (imovel root)
           (comodo imovel
                (numero_de_paredes 4)
                (formato retangular)
                (altura 3)
                (area_minima 8)
                (volume_minimo 24))
           (sala comodo
                (mobilia (sofa,mesa,cadeira))
                (finalidade convivencia))
           (dormitorio comodo
                (mobilia (cama,guarda_roupas))
                (finalidade dormir))
           (banheiro comodo
```

```
        (mobilia (pia,vaso,chuveiro,chuveiro))
        (finalidade higiene_pessoal))
(cozinha comodo
        (mobilia (fogao,geladeira,pia,freezer))
        (finalidade preparo_refeicoes))
(sala_de_estar sala
        (mobilia (sofa,poltrona,cadeira))
        (finalidade convivencia))
(sala_de_jantar sala
        (mobilia (mesa,cadeira,balcao))
        (finalidade servir_refeicoes))))
```

3.4. Inteligência Artificial Distribuída

A Inteligência Artificial Distribuída (IAD) é um dos ramos recentes da Inteligência Artificial (IA) que tem sido foco de atenção dos pesquisadores e apresenta um enorme potencial de aplicações [DLC89]. A IAD se preocupa com o estudo e o desenvolvimento de sistemas compostos por coleções descentralizadas de componentes que interagem entre si.

Os componentes dos sistemas de IAD, normalmente, são chamados *agentes*, um grupo de entidades inteligentes que interagem por cooperação, coexistência ou competição uns com os outros de forma que possam resolver problemas, executar tarefas, etc. Neste nível, um sistema passa a formar uma sociedade chamada *sociedade de agentes*.

Alguns pesquisadores, como Nilsson, acreditam que a IAD tem um papel fundamental para o entendimento da IA. Segundo Nilsson, um sistema pode ser tão complicado e conter tanto conhecimento embutido que a partição deste em diferentes entidades cooperativas resultará num aumento da eficiência - que diz respeito a modularidade, flexibilidade e resposta ao longo do tempo [NIL80].

Por razões históricas, a IAD se dividiu em duas abordagens: Solução Distribuída de Problemas (SDP) e Sistemas Multi-Agentes (SMA). A seguir apresenta-se uma breve discussão a respeito destas abordagens.

3.4.1. Solução Distribuída de Problemas

Os primeiros sistemas de IAD que surgiram foram os chamados sistemas de Solução Distribuída de Problemas (SDP). Eles tinham como preocupação inicial a construção de sistemas distribuídos para solucionar os problemas de IA de alta complexidade ou de natureza distribuída.

Pode-se dizer que um objetivo geral da SDP é a utilização da robustez e da capacidade de processamento, oferecidas pela tecnologia de redes, para atacar problemas naturalmente distribuídos ou bastante complexos. Desta forma, desenvolve técnicas de raciocínio e representação de conhecimento que propiciem aos solucionadores de problemas (chamados *agentes* ou *nós*) a cooperação para que possam resolver o problema distribuído em questão.

A SDP define sistemas distribuídos fracamente acoplados, compostos por agentes (ou nós) semi-autônomos que realizam sofisticadas técnicas de solução de problemas e que interagem entre si para solucionar um único problema [COS97].

Os agentes envolvidos em SDP são programados para cooperar, dividir tarefas, comunicar-se de forma que consigam encontrar a solução de um problema. No entanto, a experiência mostrou que não é simples estabelecer esta integração em uma coleção de indivíduos, e por isto houve a necessidade de um estudo das pressuposições básicas sobre agentes para que se garanta a possibilidade de ação cooperativa em sociedade. Assim, o agente tornou-se o foco das pesquisas, surgindo uma outra abordagem chamada Sistemas Multi-Agentes (SMA). Os SMA são apresentados a seguir.

3.4.2. Sistemas Multi-Agentes

A abordagem através de Sistemas Multi-Agentes (SMA) tem como foco de estudo o agente e sua perspectiva social, ou seja, qual a sua capacidade de relacionar-se com os demais agentes, visando uma ação comum.

O objetivo geral dos SMA é a coordenação de um comportamento inteligente entre os vários agentes da comunidade, de forma que cada agente coordene seus conhecimentos, seus objetivos, suas habilidades e planos para juntamente com os outros agentes tomarem uma ação para resolverem problemas.

Assim como os agentes da SDP (também ditos nós) os agentes de SMA podem compartilhar recursos, tais como conhecimentos e resultados parciais. No entanto, nos SMA, os agentes devem além disso raciocinar sobre o seu processo de coordenação. Assim, a coordenação nos SMA é um aspecto fundamental já que se esta não existir, qualquer benefício de interação inexiste e o comportamento do grupo de agentes pode tornar-se caótico.

Ferber [FG91] coloca que um agente nos SMA é uma entidade real, ou virtual, capaz de agir sobre ela mesma e sobre seu ambiente, que em um universo multi-agente pode comunicar-se com outros agentes. Gasser [FG91] ressalta o aspecto *identidade* do agente, afirmando que um agente pode ser considerado como um meio que produz um certo número de ações a partir dos conhecimentos e mecanismos internos que lhe são próprios.

Os SMA baseiam-se na idéia dada pela metáfora de inteligência, a *comunidade inteligente* [BIT96], ou seja, a base para a inteligência do sistema é o comportamento social. Os agentes, que são os membros de uma comunidade inteligente, podem variar sua complexidade em termos computacionais, de forma que existem os chamados *Agentes Reativos*, que são os mais simples, e os *Agentes Cognitivos ou Inteligentes*, que exibem comportamentos mais complexos. Algumas característica interessantes dos tipos de agentes na comunidade inteligente serão apresentadas a seguir.

(a) Agentes Reativos

Os Agentes Reativos são baseados em modelos de organização biológica sendo que seu funcionamento segue o modelo *Estímulo-Resposta*.

As sociedades de agentes reativos normalmente são formadas por um grande número de agentes extremamente simples, que não apresentam memória, não planejam sua ação futura e não se comunicam com outros agentes.

Segundo Ekdahl [EAD95], os agentes reativos têm mostrado suas limitações, visto que além de não serem muito versáteis, possuem problemas para obter conhecimento que dependa de raciocínio, pois o raciocínio envolve elementos que não estão diretamente ligados à percepção.

(b) Agentes Cognitivos ou Inteligentes

Os Agentes Cognitivos ou Inteligentes (ou ainda deliberativos) são baseados em modelos de organizações sociais humanas. A grande vantagem da utilização dos agentes cognitivos é que estes são capazes de conversar, dialogar, negociar, buscar um consenso, etc.

Eles possuem habilidades e capacidades significativas que em grande parte das vezes diferem ou complementam as de outros agentes. Por outro lado têm dificuldade em construir conhecimento diretamente a partir do mundo real.

Desta forma, o modelo de comunicação passa a ser mais complexo, visto que os agentes passam a se comunicar através de linguagens e protocolos ou ainda por sofisticadas estratégias de negociação.

Alguns exemplos de aplicações dos agentes cognitivos são simulação de organizações, estruturas de mercado, sistemas de automação industrial, telecomunicações, robótica, sistemas tutores inteligentes, aplicações de entretenimento como jogos, realidades virtuais habitadas por agentes, etc.

Além dos agentes puramente cognitivos ou reativos, pode-se pensar em outra abordagem, que seria o uso de agentes autônomos baseados em mecanismos híbridos. Muitos pesquisadores têm sugerido que a síntese dos benefícios dos agentes reativos e dos cognitivos seria muito interessante, uma vez que o sistema reativo poderia tratar as tarefas de rotina e o sistema cognitivo poderia ser usado para tarefas mais avançadas [WAZ97].

Este trabalho tem como interesse os agentes cognitivos e por isto serão enfatizados de forma resumida, na próxima seção, os elementos básicos para caracterizar um Sistema Multi-Agentes Cognitivos.

3.4.3. Sistemas Multi-Agentes Cognitivos

Numa sociedade, os agentes se articulam para resolver um dado problema, e esta idéia de cooperação conduz a duas noções importantes: a de organização e a de controle.

Pode-se dizer que a organização diz respeito ao modo pelo qual um grupo de agentes se organiza para resolver um problema específico, que conforme o caso, pode ser decomposto em subproblemas que são repassados para os agentes aptos a resolvê-los.

Numa concepção de organização dinâmica, os agentes não possuem *a priori* um único problema a ser solucionado, já que os agentes pré-existem a eventuais problemas que possam ser solucionados pelo sistema. Surge com isso a noção de formação de coalizões dinâmicas entre agentes, com vistas a solucionar um dado problema.

Distingue-se dois modelos de organização dinâmica: a *rede contractual* e as *coalizões baseadas em dependência*. No primeiro modelo, quando um agente não consegue resolver

um problema sozinho, ele realiza uma busca por agentes da sociedade que possam lhe ajudar, sem saber qual deles tem a capacidade requerida, ou até mesmo se existe algum agente que tem o conhecimento para lhe ajudar. Por outro lado, no modelo das coalizões baseadas em dependência, quando o agente necessita uma cooperação, ele realiza uma busca inteligente, visto que possui informações a respeito das capacidades e das habilidades dos outros agentes da sociedade.

Com relação à noção de controle, numa sociedade de agentes pode haver o *controle centralizado*, no qual existe um agente controlador encarregado de coordenar todas as atividades de cooperação entre agentes, e o *controle distribuído* que, como o próprio nome sugere, distribui a tarefa de controle para cada um dos agentes que passam a ter um controle local.

CAPÍTULO 4

Sistemas Tutores Inteligentes

Neste capítulo, pretende-se fornecer uma visão geral a respeito dos Sistemas Tutores Inteligentes, que serão objeto de estudo nos próximos capítulos quando será discutida a arquitetura do sistema tutor adotado neste trabalho. Inicia-se descrevendo as características gerais de um Sistema Tutor Inteligente (STI), tendo como base as atuais concepções discutidas dentro da área. Em seguida, discute-se o contexto histórico, importante para o entendimento geral do funcionamento de um STI, e por fim analisa-se com detalhes a arquitetura básica de um STI.

4.1. Características Gerais de um Sistema Tutor Inteligente

Como foi discutido no capítulo 2, sabe-se que as modernas correntes da Educação vêm incentivando a troca do modelo de ensino tradicional, centrado no professor, pelo modelo centrado no aluno que se baseia no "*aprender a aprender*". Quando se buscava a simulação de algumas das capacidades cognitivas do aluno, para que se pudesse utilizar estes resultados como base das decisões pedagógicas a serem tomadas, vislumbrou-se a aplicação de técnicas e métodos de IA em sistemas CAI, surgindo assim os sistemas *Intelligent Computer-Assisted Instruction (ICAI)* mais conhecidos por *Intelligent Tutorial System (ITS)* ou ainda Sistemas Tutores Inteligentes (STI).

A partir de esforços empreendidos em torno de resultados oriundos das áreas de Inteligência Artificial (IA), Psicologia Cognitiva e Educação, os STI's surgiram na década de 70. Da IA eles utilizam métodos relacionados, por exemplo, à representação e manipulação do conhecimento sobre um dado domínio, chegando-se à definição de um modelo do domínio (modelo do "expert") [BB75]. Da Psicologia Cognitiva espera-se uma resposta para a questão ligada *a quem ensinar*, que diz respeito ao conhecimento que o sistema deve possuir sobre o estudante. Finalmente, a área de Educação encarrega-se de subsidiar mecanismos para o sistema trabalhar as questões sobre o *como e quando ensinar*.

A introdução das técnicas da IA provocou um deslocamento metodológico qualitativamente radical na forma de programar sistemas educacionais: da programação de decisões à declaração do conhecimento. Estas técnicas dão ao sistema características inteligentes e servem para analisar o estilo e a capacidade de aprendizagem do aluno. Por exemplo, o sistema deve detectar o momento de aprendizagem do estudante e, a partir desta informação, mudar as estratégias de tutoramento quando for o caso. Outro exemplo, seria a capacidade que o sistema tem de oferecer uma instrução especial para o aluno que está apresentando dificuldades a respeito do tema em questão.

Portanto os STI's modificam suas bases de conhecimento à medida que interagem com o aluno. Ao perceber as intervenções do aluno, aprendem e adaptam as estratégias de ensino de acordo com as ações do mesmo e então constróem um Modelo Cognitivo do aluno a partir desta interação. Assim são formuladas estratégias de ensino-aprendizagem adequadas ao aluno e à situação do momento [VIC90].

Outra característica de um STI é possuir uma grande base de conhecimento a respeito do tema a ser ensinado, construída por um especialista, a partir da qual o sistema interage com o aluno, simulando a figura de um professor humano. A modelagem cognitiva se torna constante e progressiva.

Em resumo, o objetivo fundamental do STI é proporcionar uma instrução adaptada ao aluno, tanto em conteúdo como na forma, tornando seu comportamento cada vez mais próximo ao de um professor humano.

Na próxima seção tem-se uma breve abordagem histórica dos STI's, interessante para que se entenda o desenvolvimento destes sistemas ao longo de suas três décadas de existência.

4.2. Histórico

Segundo Wenger [WEG87] as pesquisas na área de STI começaram de fato com o desenvolvimento do sistema clássico chamado SCHOLAR; este sistema foi projetado e construído pelo Professor Jaime Carbonell no *BBN-Bolt Beranek and Neuman Inc.*, um laboratório de pesquisa em Cambridge, Massachusetts, no início da década de 70. Desta forma o SCHOLAR representa um marco histórico para os STI's, não apenas por ter sido o precursor, mas também por servir de base para muitos trabalhos que se seguiram.

De forma geral, este sistema tem por objetivo o ensino de geografia da América do Sul. O sistema conduz um diálogo com o aluno, fazendo uso do método socrático como estratégia tutorial. Para representar o conhecimento, ele usa uma rede semântica (ver seção 3.3.5) em cujos nodos estão os objetos e conceitos geográficos. A tabela 4.1 mostra um exemplo de um diálogo entre o tutor e aluno.

Os objetos e os conceitos são organizados por uma hierarquia parcial, de forma que algumas inferências simples podem ser feitas pela propagação das propriedades hierárquicas. Por exemplo, basta saber que o Chile está na América do Sul e que Santiago está no Chile para concluir que Santiago está na América do Sul.

SCHOLAR:	O planalto central e leste da Argentina é ____.
ALUNO:	Chaco.
SCHOLAR:	Não. O planalto ao nordeste da Argentina é chamado Chaco. O planalto central e leste da Argentina é os Pampas.
ALUNO:	Chaco fica no Chile?
SCHOLAR:	Não. O Chaco fica na Argentina, Paraguai e Bolívia, não no Chile.
ALUNO:	Os Andes é uma região do Chile?
SCHOLAR:	Bom, você está quase certo. Mais especificamente, na região do Chile se encontra os Andes do Sul.

Tabela 4.1: Exemplo de um diálogo entre o SCHOLAR e um aluno

O SCHOLAR foi objeto de muitos estudos e outras versões foram construídas. No entanto a modelagem do aluno sempre foi um problema, visto que não se conseguia uma forma ideal ou completa para este. Então o sistema não possuía suficientemente desenvolvida a capacidade de fazer inferências sobre o comportamento do aluno de forma que pudesse ajustar as estratégias de ensino.

Collins fez um estudo sobre o método socrático, que parte de conhecimentos que o aluno já domina e ensina através de perguntas e diálogos, levando-o a tirar suas próprias conclusões. Analisando os resultados do SCHOLAR, apresentou um conjunto de regras que podem ser utilizadas para conduzir o diálogo nos STI's [BAR82]. Para testar este conjunto de regras tutoriais, criou o projeto WHY. Seu trabalho não foi muito bem reconhecido na literatura por se tratar de uma abordagem muito teórica e apresentar alguns problemas.

O projeto coordenado por Brown&Burton chamado SOPHIE – *SOPHisticated Instrucional Environment* [WEG87] é de uma certa forma, uma reminiscência do SCHOLAR. É baseado num tutor usado para ensinar meteorologia e funciona como um laboratório onde o aluno tem a chance de aplicar seus conhecimentos e receber um *feedback*, ou seja, uma avaliação de suas hipóteses. A partir do SOPHIE foram criados

outros projetos, porém este ambiente ainda é usado hoje numa versão de jogo, com um domínio referente a circuitos elétricos.

A partir destes primeiros estudos, os pesquisadores já se deparavam com as limitações destes sistemas e passaram então a trabalhar na tentativa de solucioná-las. Dentre estas tentativas, pode-se destacar o projeto desenvolvido com o auxílio de professores, que descreviam as estratégias que tomariam caso tivessem que orientar os alunos no terminal, enquanto os pesquisadores implementavam estas estratégias no sistema. Porém esta prática deu origem a diferentes bases de conhecimento, formadas por diferentes pontos de vista oriundos de diferentes especialistas.

Por outro lado a complexidade do domínio escolhido para o SOPHIE fez com que surgissem projetos separados para investigar vários tópicos relacionados, em um domínio específico. A partir daí começaram a surgir vários sistemas que se encaixavam na categoria de um STI e que tentavam reparar e melhorar os problemas encontrados até então.

Wenger [WEG87] apresenta um *survey* descrevendo detalhadamente os principais STI's desenvolvidos desde o SCHOLAR até meados de 1986. Em [SEL88] tem-se além de descrições dos sistemas, algumas perspectivas e direções futuras.

Atualmente, a idéia defendida é a de considerar o aprendiz como elemento ativo no processo de aprendizagem, levando os STI's para um estilo de interação mais flexível. Desta forma, novas abordagens surgem na concepção de ambientes computacionais de ensino e aprendizagem. Nesse sentido, os STI's tradicionais evoluem para os Ambientes Interativos/Inteligentes de Aprendizagem (*Interactive/Intelligent Learning Environment* - ILE) ou ainda Sistemas Tutores Cooperativos.

Um ILE pode ser caracterizado como uma categoria de software educacional na qual o aprendiz é colocado em uma situação de resolução de problemas. Durante esse processo, o tutor artificial assiste o aprendiz em sua tarefa e monitora o seu aprendizado [COT97].

A seguir tem-se uma descrição detalhada dos módulos básicos que formam a arquitetura geral de um STI.

4.3. Arquitetura Geral de um STI

Um STI possui uma estrutura básica, composta por quatro módulos, a saber:

- Módulo do Domínio – MD
- Módulo do Aprendiz – MA
- Módulo do Tutoramento – MT
- Módulo de Interface – MI

A figura 4.1 ilustra estrutura básica de um STI.

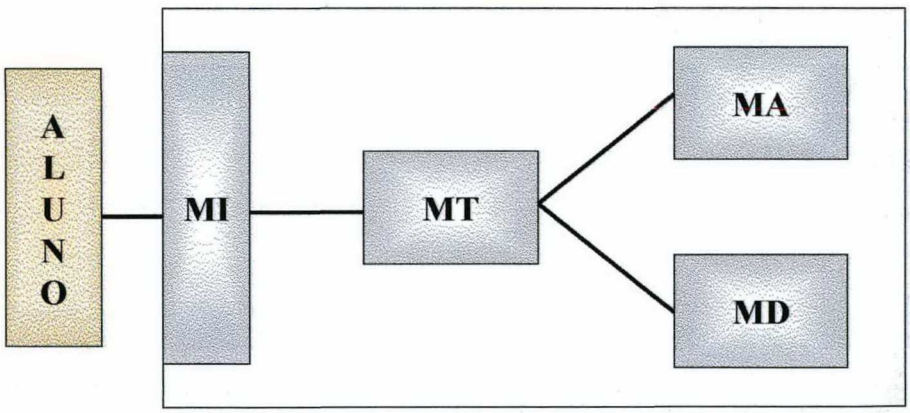


Figura 4.1: Arquitetura Geral de um STI

Estes módulos se inter-relacionam de forma a refletir o processo ensino-aprendizagem. Na estrutura básica existem conhecimentos especializados para se tratar questões como: *o que ensinar*, que é modelada no MD e representa a área de conhecimento a ser estruturada; *a quem ensinar*, que é resolvida quando o MA procura captar o atual estado de aprendizagem do estudante e *como ensinar*, respondida quando o MT faz o elo entre os módulos de forma a definir as estruturas didática e pedagógica a serem adotadas.

4.3.1. Módulo do Domínio

O módulo do domínio de ensino, também conhecido como módulo do modelo do perito ou do especialista, atua como fonte provedora de conhecimento sobre um domínio particular de ensino, a ser ministrado pelo tutor. Este conhecimento é formalizado em fatos e regras de forma que se gere questões, respostas, tarefas, etc. Ainda pode funcionar como um provedor de padrões para a avaliação do desempenho do aprendiz.

Um STI poderoso precisa ter um corpo de conhecimento do domínio abundante e robusto, no sentido de se ter um corpo vasto de conhecimento. Assim, a aquisição de conhecimento deve vir de diferentes peritos do domínio, o que torna esta tarefa bastante trabalhosa. Devido a este dificuldade, muitas vezes o MD é incompleto.

A natureza do conhecimento abordado, ou seja, a forma como o conhecimento é interpretado e manipulado pelo cérebro humano, é um dos fatores que influenciam o entendimento do conteúdo.

Existem algumas técnicas utilizadas para se tratar e representar este conhecimento, de forma que se consiga uma representação para o MD o mais próxima possível da realidade. A maioria dos STI's trabalham com o conhecimento do tipo procedural ou com o conhecimento do tipo declarativo, sendo que existem tentativas híbridas.

A seguir tem-se algumas características básicas dos tipos de conhecimento citados.

(a) Conhecimento procedural

O conhecimento procedural ou procedimental é aquele que está relacionado com tarefas que podem ser algoritmizadas, ou seja, tarefas que podem ter sua execução dividida em comandos básicos e uma estrutura de controle de fluxo destes comandos.

Em Marietto [MAR] apontam-se algumas características negativas quando se opta pela representação usando-se o conhecimento procedural:

- Execução repetida de algoritmos de busca seqüencial;
- Possibilidade de redundância de conhecimento através de duas ou mais regras que representem o mesmo conceito;
- Limitação de se trabalhar com conceitos imprecisos.

(b) Conhecimento Declarativo

O conhecimento declarativo, por sua vez, está relacionado com a manipulação de conjuntos de informações. Para a sua representação tem-se as redes semânticas e os *frames* (ver seção 3.3.5) com maior aplicabilidade.

O uso de redes semânticas possibilita a representação de forma eficiente do conhecimento hierárquico, além da busca simplificada na base de conhecimento, pois informações a respeito de um componente podem ser obtidas através dos nós do grafo. No entanto sua estrutura não consegue tratar de forma adequada as exceções, as crenças e os conceitos temporais.

Uma das vantagens do uso de *frames* é que um conceito representado pode ser utilizado em diferentes contextos e, através da utilização de mecanismos de herança, é possível compartilhar informações sem perda da independência do conhecimento.

Então o conhecimento nos STI's é predominantemente representado fazendo-se uso de regras de produção, *frames* e redes semânticas de acordo com o tipo de conhecimento abordado. Desta forma, existe uma ligação evidente entre um STI e um Sistema Especialista (SE) porque, na maior parte das vezes, tenta-se aproveitar a teoria utilizada na construção da base de conhecimentos de um SE para a construção do MD. No entanto esta adaptação não funciona bem quando se analisa o fato de que a representação do conhecimento deve atender à finalidade do aplicativo.

Uma das principais diferenças entre um STI e um SE diz respeito à capacidade de identificação e de adaptação à "linguagem" do usuário que os STI's possuem, sendo que o conhecimento deve ser representado da forma mais abrangente possível. Ao contrário, os SE são destinados apenas a um usuário especializado no assunto em questão.

4.3.2. Módulo do Aprendiz

A idéia de modelar o aprendiz não é nova e surgiu na tendência de tornar os STI's adaptáveis. O módulo do aprendiz, do estudante, ou do aluno, tem como papel a representação abstrata do estado de conhecimento e desempenho do estudante, durante a interação dinâmica com o mesmo.

Representa o aprendiz individual ou, mais precisamente, o estado cognitivo do conhecimento e do comportamento do aluno, tentando identificar o que ele sabe, o que não sabe, o que entendeu mal, etc. As técnicas de diagnóstico é que fazem este papel de determinar em qual estado mental, ou seja, o atual estado de conhecimento e comportamento, o estudante se encontra. A partir destas informações, em cada situação, o STI tem base para que possa escolher as estratégias de ensino adequadas a cada aluno.

O MA tem duas funções fundamentais: a da informação e da representação [WEG87]. Como fonte de informação, o MA infere aspectos não observáveis do comportamento do aprendiz visando atingir dois objetivos: produzir uma interpretação das ações do aluno e promover a construção do conhecimento que provocou esse comportamento.

Como fonte de representação, se constitui como um subconjunto do domínio do conhecimento a ser ensinado. O MA compara o estado do conhecimento do aprendiz com o conhecimento do MD para detectar vazios, erros, etc permitindo avaliações parciais do processo de aprendizagem.

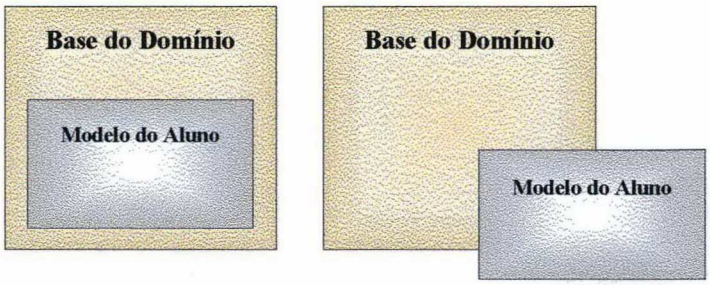
Este módulo é constituído por dados estáticos e dados dinâmicos [VIC90] que permitem ao tutor a comprovação de hipóteses a respeito do aluno. Tem-se como exemplos algumas técnicas para se determinar o MA:

- Incluir um reconhecimento de padrões aplicado à história das respostas fornecidas pelo aluno;
- Levar em conta as preferências do aluno, incluindo as coisas que ele sempre costuma esquecer quando interage com o tutor;
- Indicar os objetivos particulares de cada aluno.

Uma forma de se conceber o MA é representá-lo em relação à base de conhecimentos do domínio. A maioria dos STI's desenvolvidos até o momento utilizam dois tipos de modelos: *overlay* ou perturbação [MAR]. Por motivo de simplificação computacional, nenhum destes modelos seguem a observação teórica de que um STI deve ter o MA separado do MD. Então na prática, o MA é construído a partir do MD adicionando-se informações que os diferenciam.

Nos modelos *overlay* o conhecimento do aluno é representado como um subconjunto da base de conhecimento do sistema tutor (ver figura 4.2a), o que significa dizer que a representação de conhecimento utilizada no MA e na base do domínio é a mesma. Neste modelo, assume-se que os erros ou comportamentos anômalos do aluno são sempre devidos à ausência de alguma informação presente na base do domínio. Desta forma, sua capacidade de espelhar o conhecimento do aluno sobre um determinado tópico é limitada, visto que muitos comportamentos incorretos originam-se de concepções e idéias incorretas no raciocínio do aluno, e não simplesmente pela falta de informações.

O modelo de perturbação veio para aperfeiçoar o *overlay*. Neste modelo, existe a base do domínio e uma biblioteca, cujos membros são chamados de erros ou *bugs*, sendo que o MA inclui elementos da base do domínio e da biblioteca de erros (ver figura 4.2b). Este modelo permite um tratamento mais inteligente dos comportamentos incorretos do aluno, porém continuam limitados no que diz respeito à estruturação e gerenciamento de sua biblioteca de erros.



(a) (b)

Figura 4.2: Tipos de Módulo do Aprendiz

O MA tem grande importância para um STI e dependendo da sofisticação que se queira para este modelo, várias complicações surgem na sua implementação. Por isto, o MA vem sendo foco de pesquisadores na área, que têm procurado vencer as limitações observadas. Assim, surgem formas alternativas de se tratar o problema da modelagem que, por se mostrarem promissoras, vêm indicando a tendência de pesquisa na área de modelagem do estudante.

Algumas tendências na área podem ser enumeradas:

- desenvolvimento de teorias de aprendizagem "computáveis";
- desenvolvimento de ambientes de aprendizagem adaptativos;

- uso da teoria de linguagem natural (para auxiliar o estabelecimento dos estados mentais do aprendiz);
- uso de Lógica Fuzzy adequada para tratar com incertezas e imprecisão, características marcantes no processo de aprendizagem;
- uso de Agentes Inteligentes que permitem a representação e análise simultânea de vários pontos de vista;
- desenvolvimento de ferramentas de autoria e utilização de multi-agentes na estrutura de STI.

4.3.3. Módulo do Tutoramento

O módulo dc tutoramento também chamado de módulo pedagógico ou de estratégia de aprendizagem tem como função a decisão sobre quais as estratégias de ensino serão adotadas sobre o domínio de aplicação, a partir da interação com o MA.

As estratégias de ensino podem ser vistas como uma seqüência de ações que visam atingir determinados objetivos, definindo formas de aprender o material instrucional. Quanto ao que apresentar, as atividades pedagógicas variam desde simples dicas a explicações, sugestões, exercícios, teste de hipóteses do MA, etc.

O MT também determina a granularidade do *feedback*, ou seja, o grau de controle sobre o aprendiz ou o tamanho e conteúdo tanto das suas perguntas quanto das respostas do aprendiz. Dessa forma, um MT mal projetado pode destruir a motivação do aprendiz ou seu senso de descoberta, prejudicando a eficiência do processo de aprendizagem [CUR96]. Um dos grandes problemas do MT é a quase inexistência de conhecimento formal sobre a comunicação perito-aprendiz.

De acordo com a estratégia pedagógica adotada, define-se o tipo de ação tutorial. Portanto, dependendo do tipo de conhecimento em foco tem-se: tutor expositor e tutor procedural ou procedimental.

O tutor expositor, definido para conhecimento declarativo, tem como tarefa principal manter o foco do diálogo e a consistência ou a coerência do processo. Outra tarefa é a manutenção e a recuperação de todo o histórico do processo.

Por outro lado, a estratégia adequada para o conhecimento procedural define um tutor do tipo procedural ou procedimental. Além das tarefas citadas para o expositor, este precisa ordenar adequadamente os vários níveis de perícia e subperícias e selecionar perguntas ou exercícios que reflitam esta ordem.

4.3.4. Módulo de Interface

Uma boa interface é vital para o sucesso de qualquer sistema interativo e se tratando de STI, pode-se dizer que é nesta interação que se realizam as duas principais funções do STI: apresentação do material instrucional e monitoração do progresso do estudante através da recepção da resposta do aluno.

Com o avanço da tecnologia, as possibilidades na apresentação têm recebido um significativo avanço a partir da utilização de sistemas de hipertexto e hipermídia. Com a utilização destas ferramentas, têm-se criado aplicações bastante ricas em interatividade. Porém, convém destacar que não é isto que garante uma personalização no tratamento das diferenças individuais de cada aluno.

Geralmente a comunicação é estabelecida através de menus, gráficos, ícones, símbolos e frases, utilizando janelas e cores. Durante a sessão, tanto o tutor quanto o aluno podem mudar a forma de diálogo ou a apresentação.

A forma do aluno se comunicar com o tutor é feita através de restrições na linguagem a fim de viabilizar a comunicação. A correção ortográfica é feita da forma usual utilizando um dicionário de palavras, através do qual o tutor formula hipóteses sobre sua categoria sintática quando não encontra a palavra exatamente igual à digitada pelo aluno.

4.4. Considerações Finais

Pode-se concluir que um STI é um programa educacional do qual se espera uma capacidade de interação com qualquer tipo de usuário. Assim, o sistema promoveria ações de acordo com as características do aluno que utiliza o software.

No entanto, apesar das melhorias trazidas com a inserção da IA, os STI's ainda estão longe de poder tratar a complexidade inerente ao processo ensino-aprendizagem, principalmente no que diz respeito ao modelo do aprendiz.

A modelagem do estudante deveria ser capaz de rastrear o processo de aquisição de conhecimento por parte do aluno, identificando de forma dinâmica seu atual estado mental e, a partir deste diagnóstico, direcionar estratégias adequadas de tutoramento [MAR]. Entretanto, na prática, a modelagem do estudante está longe de ser o que se idealiza, e isto se deve principalmente à ausência de uma teoria capaz de explicar como se dá o processo de aquisição do conhecimento no contexto ensino-aprendizagem.

Os STI's não têm representado de forma adequada o processo de aprendizagem e por isto pesquisas estão sendo desenvolvidas com o objetivo de propor arquiteturas que os melhor representem [COT97].

No próximo capítulo tem-se uma descrição do ambiente MATHEMA, que serviu como base para o desenvolvimento do sistema tutor multi-agentes em lógica.

CAPÍTULO 5
O Ambiente MATHEMA

Neste capítulo será descrito o ambiente MATHEMA, baseando-se em [COT97] e levando-se em consideração os aspectos importantes para o desenvolvimento do STI em lógica. Inicialmente colocam-se alguns aspectos interessantes relativos a concepção deste modelo teórico, no sentido de situar o MATHEMA no contexto atual das pesquisas em STI's.

Apresenta-se a arquitetura geral do ambiente, bem como a definição do modelo do conhecimento do domínio adotado. Como se trata de uma abordagem multi-agentes, por fim apresenta-se a arquitetura de um agente tutor bem como as interações existentes entre estes agentes.

5.1. Considerações Iniciais

Como foi visto no Capítulo 4, quando se tratava dos Sistemas Tutores Inteligentes, existem algumas características que devem estar presentes nestes sistemas, de forma que se possa ter um ambiente que possibilite ao máximo uma interação com o aluno, sem que o processo ensino-aprendizagem seja prejudicado.

Além disso, foram colocados alguns pontos que vêm sendo objeto de pesquisas quando se pensa em conceber um Sistema Tutor Inteligente. Como exemplo, a modelagem do aprendiz deve ser efetiva, no sentido de que o sistema tenha um modelo do estado cognitivo do aluno a cada momento. Por outro lado, tem-se um compromisso com o modelo do domínio, que deve ser obtido da forma mais adequada para que contemple o conhecimento em questão. Por fim é preciso definir um modelo pedagógico que seja abrangente e que busque a qualidade na determinação da melhor estratégia a ser usada com o aluno no momento da interação [COT97].

Partindo-se de questionamentos a respeito destes pontos alvos de pesquisa, é que o ambiente MATHEMA, proposto no âmbito da tese de doutorado de Evandro de Barros Costa [COT97], foi concebido utilizando-se uma arquitetura multi-agentes.

Quando o modelo do domínio no MATHEMA foi definido, levou-se em consideração requisitos para sua qualidade, de forma que houvesse uma contribuição para que o STI se tornasse mais perceptivo, e assim o modelo do aprendiz também estaria sendo aperfeiçoado. Um estudo que levou em consideração questões sobre ganhos ao adotar-se uma abordagem de agentes, trouxe conclusões que evidenciam uma melhora na interação Sistema Tutor-Aprendiz quando se trabalha com esta abordagem [COT97]. Por isso o MATHEMA adota uma abordagem multi-agentes, que será entendida melhor no decorrer da descrição do ambiente.

O MATHEMA é um modelo de ambiente interativo de aprendizagem, concebido para possibilitar um ensino adaptativo e seus desdobramentos no processo de aprendizagem. Trata-se de um modelo teórico conceitual, que serviu como base para a implementação do sistema tutor proposto neste trabalho.

O ensino adaptativo é visto como conseqüência do processo de interações cooperativas envolvendo os seus componentes (*Aprendiz, Tutores*) em atividades baseadas em resolução de problemas. O sistema envolve o aprendiz na resolução de uma sucessão de problemas e passa a desempenhar um papel de tutor assistente. A noção de problema é colocada de

forma intuitiva, significando uma situação na qual, a partir de certas condições iniciais, tenta-se atingir um dado objetivo através de um conjunto de ações [HUN87] [NIL82] [PEA84].

A aprendizagem, por sua vez, é assumida como sendo favorecida e decorrente de atividades provenientes do ensino adaptativo, significando aquisição de conhecimento. Nesta perspectiva, o MATHEMA propõe-se a prover princípios e uma arquitetura alternativa, necessários para orientar o desenvolvimento de STIs particulares [CP96].

O MATHEMA tem como proposta central o engajamento de um aprendiz humano numa relação com uma sociedade de agentes tutores artificiais, visando envolvê-lo em situações de aprendizagem. Esses agentes tutores podem cooperar entre si ou com uma sociedade de especialistas humanos, em favor de promover a aquisição de conhecimento por parte do aprendiz. Representa um modelo de STI distribuído no qual os agentes tutores se comunicam por meio de troca de mensagens. Na próxima seção discute-se com mais detalhes a arquitetura geral do MATHEMA.

5.2. Arquitetura Geral do MATHEMA

A concepção do MATHEMA foi feita a partir de exigências de qualidade no processo de interação entre duas entidades: uma *máquina* desempenhando um papel de tutor e um *humano* comportando-se como um aprendiz. Como um recurso básico de apoio à administração da complexidade inerente ao processo ensino-aprendizagem, inicialmente elaborou-se um modelo de conhecimento multidimensional relacionado ao conhecimento do tutor sobre um dado domínio (esta questão será melhor discutida em 5.4).

A partir disso, definiu-se uma sociedade de agentes tutores especializados em operar sobre o conhecimento distribuído neste modelo e mais adiante a concepção de um modelo baseado numa arquitetura multi-agentes, considerando principalmente as atividades de

interação cooperativa envolvendo agora três entidades: um *aprendiz humano*, uma *sociedade de agentes tutores artificiais* e uma *sociedade de especialistas humanos*.

A arquitetura geral do MATHEMA pode ser visualizada na Figura 5.1, e os seus componentes estão descritos sucintamente a seguir.

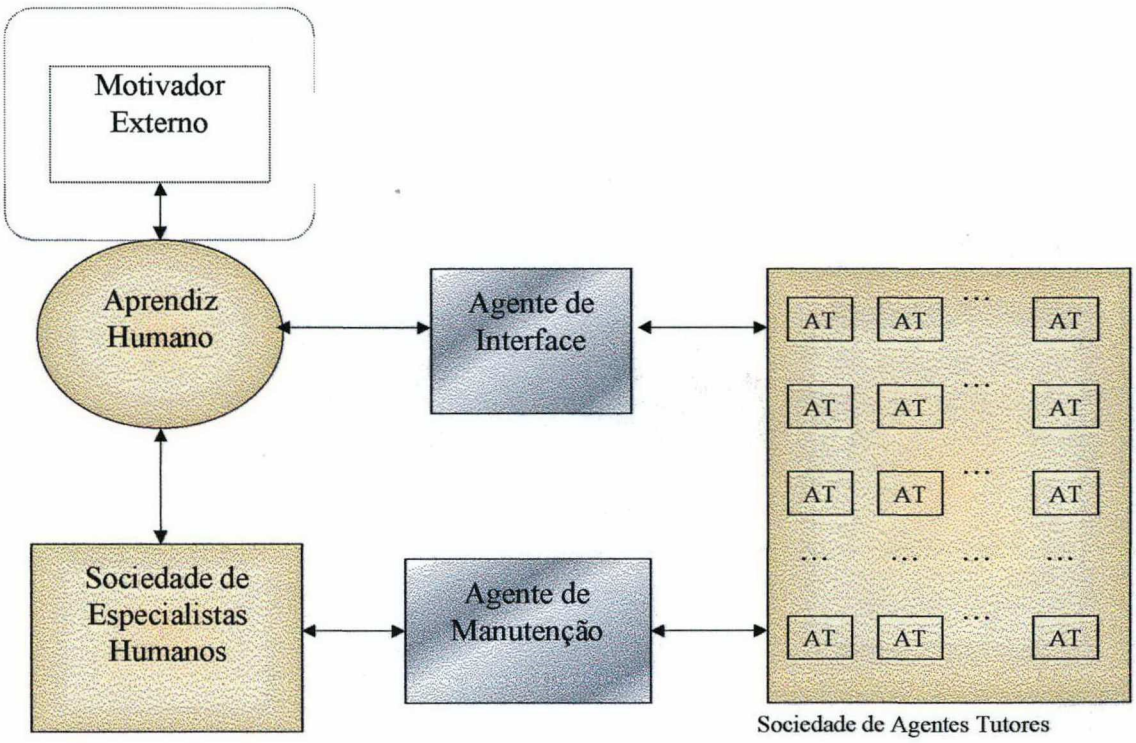


Figura 5.1: Arquitetura Geral do MATHEMA.

- *Aprendiz Humano* (AH): agente interessado em aprender algo sobre um dado domínio trabalhado no MATHEMA. Desempenha um papel de agente ativo quando envolvido em atividades baseadas em resolução de problemas, e fica sob a assistência especializada de um agente tutor, que por sua vez faz parte de uma sociedade de agentes tutores cooperantes.

- *Sociedade de Agentes Tutores Artificiais* (SATA): conjunto de agentes capazes de cooperar entre si a fim de promover a aprendizagem de um dado aprendiz quando se encontra numa atividade de resolução de problemas. Cada agente integrante da SATA é especializado em subdomínios relacionados a um dado domínio de conhecimento. Essa idéia foi inicialmente inspirada nas reflexões contidas em *Sociedade da Mente*, de Marvin Minsky [MIN85]. Minsky, um dos fundadores da IA, desenvolveu uma série de idéias a respeito da inteligência, todas baseadas em um mesmo princípio: a mente é organizada como uma coleção de pequenas unidades desprovidas de inteligência, e a inteligência emerge da combinação destas, cada uma responsável por um pequeno processo. Assim, a mente seria organizada como uma *sociedade* dessas pequenas unidades, que recebem o nome de *agentes*.

- *Sociedade de Especialistas Humanos* (SEH): funciona como uma fonte de conhecimento externa ao sistema e se comunica com a SATA através de manutenção. Existem duas funções principais da SEH: promover o processo de manutenção da SATA, que diz respeito à inclusão e exclusão de agentes, bem como alterações no conhecimento dos agentes e estar disponível para que, em casos em que a SATA não consiga resolver um determinado problema, possa auxiliar os agentes e assistir os aprendizes. Quando não é requisitada para uma das funções colocadas, a SEH analisa o desenvolvimento das interações entre o aprendiz e a SATA, avaliando o desempenho de seus agentes tutores e daí, sempre que necessário, promovendo-lhes melhorias.

- *Agente de Interface* (AI): é o agente responsável pela interação entre o aprendiz e a SATA. Ajuda o aprendiz no processo de escolha de um *agente supervisor* (este processo será melhor discutido na seção 5.3), de acordo com suas necessidades no momento, e por isto sabe da existência dos agentes na SATA bem como de suas capacidades.

- *Agente de Manutenção* (AM): é o agente responsável pela interação entre a SATA e a SEH. Sua função é a de prover esta interação, oferecendo meios necessários para a percepção, comunicação e manutenção da SATA.

- *Motivador Externo* (ME): representa as entidades humanas externas que podem motivar o aprendiz à utilização do MATHEMA. Exemplos de ME's são os professores, os colegas, etc.

5.3. Interações no MATHEMA

Depois de analisar a arquitetura geral do MATHEMA, é interessante discutir a respeito das interações existentes. No sistema há essencialmente três tipos de interações (que podem ser visualizadas na figura 5.2):

(i) a que ocorre entre o aprendiz humano e a SATA via um agente tutor;
(ii) a interação entre agentes tutores na SATA;
(iii) entre a SATA e a SEH.

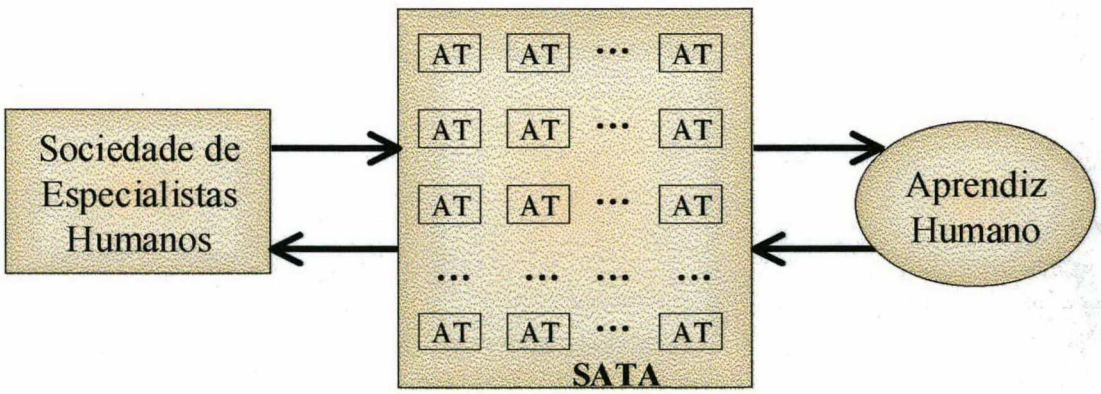


Figura 5.2: Interações que ocorrem no MATHEMA.

Para que se possa ter uma visão geral do funcionamento destas interações, analisa-se a seguir algumas situações que ocorrem quando se utiliza o MATHEMA.

O aprendiz humano pode ser incentivado por um motivador humano externo a usar o ambiente MATHEMA pela primeira vez. Quando este processo se inicia, começa a existir a interação *(i)* entre o aprendiz e o agente de interface (AI). O AI procura saber quais os objetivos do aprendiz, tentando ainda apresentar o ambiente levando em conta estes objetivos.

Depois dos objetivos estarem claros, escolhe-se um *supervisor* na SATA chamado *agente supervisor* que passa a ser responsável pelo aprendiz, oferecendo-lhe a orientação e o suporte pedagógico necessários para que obtenha o aprendizado.

Durante a interação entre o agente supervisor e o aprendiz, podem ocorrer três situações:

- Existe interação apenas entre o aprendiz e o seu agente supervisor, enquanto este agente consegue sozinho resolver as situações que se apresentam;

- Com o aumento da complexidade das situações no decorrer da interação, o aprendiz passa a exigir do *supervisor* a solução de problemas que ele sozinho não é capaz de fazer. Desta forma passa a requerer a participação de outros agentes que começam a cooperar de forma a atender às necessidades do aprendiz;

- Num último caso nem mesmo o auxílio dos agentes da SATA é suficiente para que se atenda o aprendiz, então o *agente supervisor* comunica ao aprendiz sobre a impossibilidade de atendê-lo naquele momento e busca ajuda da SEH usando o *agente de manutenção*.

Caso o aprendiz não esteja utilizando o MATHEMA pela primeira vez, as interações funcionarão basicamente da mesma forma, porém o AI já terá conhecimento do que já foi visto anteriormente com aquele aprendiz, ou seja, o sistema já possui um certo

modelamento para aquele aluno. A partir destas informações o AI dá continuidade ao processo oferecido pelo ambiente dentro do conteúdo em questão e da mesma forma, podem existir as interações já citadas.

5.4. Modelagem do Conhecimento sobre um Domínio

Como já foi dito anteriormente, a técnica utilizada para a modelagem do domínio no MATHEMA foi concebida com o intuito de amenizar a complexidade inerente ao processo ensino-aprendizagem. Desta forma tem-se um compromisso entre "riqueza" e estruturação apropriada do domínio [CP96], que resulta na disponibilidade de múltiplas visões sobre um mesmo conhecimento.

Em [COT97] apresenta-se um modelo estrutural para o domínio de conhecimento, no qual existe uma partição em subdomínios e uma posterior organização adequada para que haja o apoio necessário no processo de interação, que ocorre entre as entidades que compõem o ambiente.

A idéia básica é submeter um dado domínio de conhecimento a um particionamento em diferentes subdomínios, buscando um corpo de conhecimento que lhe seja subordinado no momento de sua operacionalização. Esta busca é orientada de maneira a alcançar um conhecimento com especialidades distribuídas em três dimensões de conhecimento [CP97]:

(i) uma para o *contexto* (C), que diz respeito às várias interpretações que podem ser dadas a um domínio de conhecimento a ser ensinado. É composta por diferentes pontos de vista sobre o domínio e constitui-se em representações ou abordagens diferentes de um mesmo objeto de conhecimento.

(ii) uma para *profundidade* (P), que é definida em relação a um contexto particular do domínio. Significa que um domínio pode ser constituído de vários níveis, pois um determinado conhecimento pode ser ensinado em níveis distintos que irão depender,

por exemplo, do grau de conhecimento do aluno sobre o assunto em questão e sobre as técnicas necessárias ao tratamento dos problemas do domínio.

(iii) uma para a *lateralidade* (L), que trata dos conhecimentos fortemente relacionados ao domínio de aplicação, vistos num contexto e profundidade fixados. Diz respeito aos conhecimentos afins de suporte a uma dada unidade de conhecimento, consistindo nos pré-requisitos da referida unidade.

Com esta visão multidimensional, existe a possibilidade de um domínio poder ser enfocado por uma visão contextual, sendo que esta pode vir acompanhada de várias alternativas de variação do ponto de vista de profundidade e lateralidade em relação a cada contexto escolhido no domínio.

A definição de um conhecimento sobre um domínio, a partir da visão multidimensional, é feita com a definição de pares $< C_i, P_{ij} >$,

onde C_i é um contexto particular e

P_{ij} é a j-ésima profundidade definida relativamente ao i-ésimo contexto C_i.

A cada um desses pares $< C_i, P_{ij} >$ associa-se um subdomínio d_{ij}. Daí, para cada par, deve-se identificar suas lateralidades L_{ijk}. A lateralidade pode ser definida como os "pré-requisitos" externos que cada d_{ij} necessita para seu entendimento, sendo que a cada L_{ijk} é associado um domínio d_{ijk}.

Assim, a definição de um determinado conhecimento sobre um domínio é dada pela união de subdomínios específicos com o conhecimento lateral adjacente, ou seja:

$d_{11} \cup d_{12} \cup \ldots \cup d_{21} \cup d_{22} \cup \ldots \cup$ lateralidades correspondentes.

Para ilustrar a noção de diferentes dimensões do conhecimento, tem-se um exemplo no domínio específico da Geometria [CBF97] [FBC97].

Exemplo: Domínio da Geometria Euclidiana Plana.

Supondo que o domínio seja o da Geometria Euclidiana Plana, especificamente o estudo dos triângulos, pode-se definir as dimensões como segue.

D = *Geometria Euclidiana Plana* – Estudo dos Triângulos.

⇨ *Contextos:*

C_1 = contexto métrico;

C_2 = contexto trigonométrico.

Cada contexto acima possui níveis de conhecimento (profundidades), desde o mais elementar até o mais complexo. Por exemplo, ao contexto C_1 podem ser associadas as seguintes profundidades:

⇨ *Profundidades:*

P_{11} = triângulos retângulos;

P_{12} = triângulos quaisquer.

Fixando-se o contexto C_1 juntamente com a profundidade P_{11}, as lateralidades associadas ao par $< C_1, P_{11} >$, podem ser:

⇨ *Lateralidades:*

L_{111} = produtos notáveis;

L_{112} = equações do 2° grau.

Estes exemplos de contexto e profundidades podem ser ampliados ou reorganizados de acordo com o objetivo didático que o especialista deseja perseguir.

Pode-se pensar em uma visão interna de cada subdomínio d_{ij} em D, na qual tem-se um domínio como constituído por um conjunto de entidades pedagógicas definidas em função de objetivos de ensino-aprendizagem específicos que estão associados a um *curriculum*. Ao *curriculum* associa-se um conjunto de unidades pedagógicas dispostas segundo uma ordem definida e relacionada ao pré-requisito e nível de dificuldade.

No capítulo 6, quando se fala a respeito do domínio da lógica, tem-se algumas considerações sobre como se definiu os domínios d_{ij} de acordo com a estrutura pedagógica adotada.

A adoção de SMA é decorrente principalmente das considerações feitas sobre o conhecimento do domínio. A sociedade é definida em função da organização e modelagem do domínio de conhecimento que conduz a dois tipos de agentes: *(i)* agentes constituídos através de pares $< C,P >$, significando um contexto e uma profundidade associada e *(ii)* agentes L que são associados a cada par $< C,P >$ [CPJ97]. Na próxima seção tem-se uma descrição mais detalhada da arquitetura geral destes agentes.

5.5. Arquitetura Geral de um Agente Tutor

Os agentes tutores artificiais que formam a SATA são capazes de, através de linguagens e protocolos, cooperarem entre si a fim de oferecer ao aprendiz condições mais efetivas no suporte às atividades de aprendizagem em situações de resolução de problemas.

Estes agentes são do tipo cognitivo (na seção 3.4.2 discutiu-se as características dos agentes cognitivos ou inteligentes) e são assumidos como benevolentes, ou seja, sempre

concordam em cooperar com os demais agentes quando forem solicitados. Eles possuem propriedades como [FG96]:

- *Autonomia*: os agentes têm os seus próprios objetivos, o que permite a eles lidar com estes sem a intervenção direta de outros agentes;
- *Orientação a objetivos*: diz respeito ao fato dos agentes exibirem um comportamento de acordo com suas capacidades;
- *Habilidade social*: possibilidade de um agente poder interagir com os demais e com o mundo externo à SATA.

Cada agente é uma entidade especializada em um domínio específico (conforme definido na seção 5.4), e desempenham papéis que incluem principalmente os de tutores inteligentes cooperativos no momento de suas interações com o aprendiz.

Costa [COT97] apresenta a arquitetura de um agente tutor oferecendo uma visão em dois níveis de abstração distintos: o *nível macro* e o *nível micro*, que serão descritos a seguir.

5.5.1. Arquitetura do Agente sob uma Visão Macro

Dentro desta visão macro, tem-se uma arquitetura geral de um agente tutor constituída por uma composição hierárquica de três sistemas, a saber:

- *Sistema Tutor*: é responsável pelas interações cooperativas/colaborativas entre o agente tutor e o aprendiz. Em suma cabe a ele a execução das atividades tutoriais, utilizando-se de um modelo do domínio, de um modelo pedagógico e de um modelo do aprendiz (assim como foi definido para os STIs em 4.3).

- *Sistema Social*: é responsável pelo comportamento cooperativo de um agente com os demais agentes e com a SEH, através do agente de manutenção. É composto por bases de conhecimento e mecanismos de raciocínio necessários para realizar tal comportamento cooperativo, refletindo assim o comportamento social do agente.

- *Sistema de Distribuição*: é responsável pelas atividades de envio e recebimento de mensagens através do meio de comunicação. Além disso gerencia internamente a distribuição das mensagens no agente tutor.

A figura 5.3 apresenta a arquitetura geral, a nível macro, de um agente tutor.

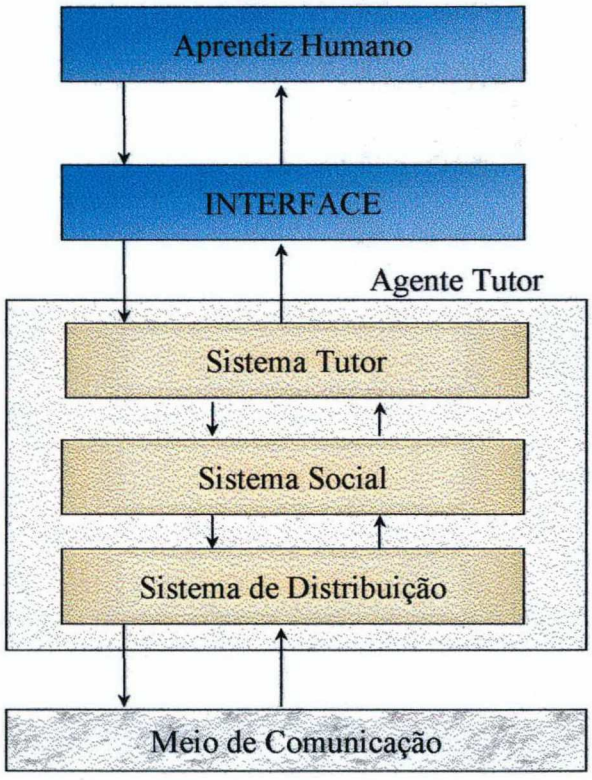


Figura 5.3:Arquitetura de um Agente Tutor: nível macro.

5.5.2. Arquitetura do Agente sob uma Visão Micro

Analisando a arquitetura do agente tutor sob uma visão micro, tem-se na verdade uma visão interior desta, na qual se descreve os módulos de cada um dos sistemas mencionados no item anterior. A figura 5.4 ilustra os componentes dos sistemas que compõem o agente.

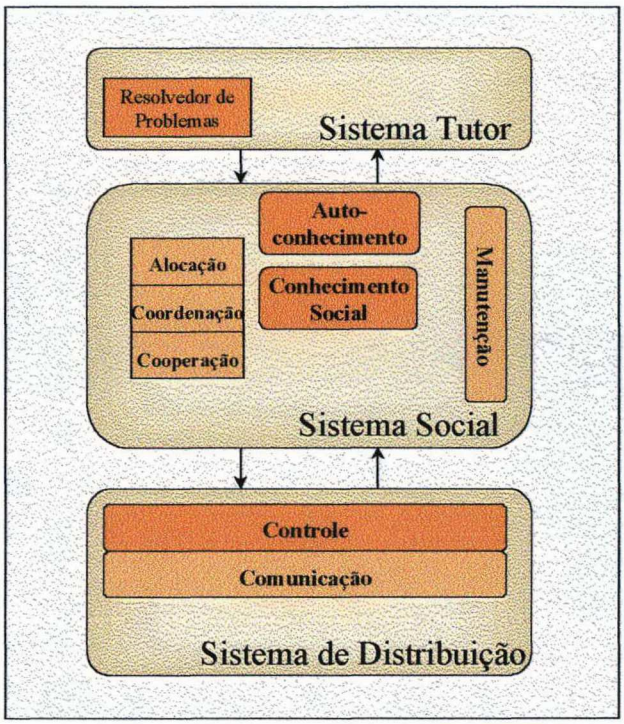


Figura 5.4:Arquitetura de um Agente Tutor: nível micro.

O sistema tutor é definido por um módulo principal chamado *Resolvedor de Problemas*, encarregado de resolver problemas, ou seja, de resolver tarefas que incluem a própria resolução de um problema, bem como o diagnóstico e a instrução. É dotado de mecanismos que proporcionam a identificação das tarefas que são solucionáveis pelo agente e as que necessitam de especialidades de outros agentes tutores. Uma de suas

funcionalidades é a decomposição de tarefas, interessante quando se tem uma tarefa complexa e que exige habilidades de outros agentes.

O sistema social é definido por seis módulos que possuem as seguintes funções:

- *Autoconhecimento* (AC): é a estrutura de conhecimento de suas próprias habilidades, onde se representa o que o agente sabe fazer. É uma base de conhecimento de apoio a atividade cooperativa e é usado quando um agente necessita decidir se possui conhecimento para resolver uma tarefa pedagógica. É definido da seguinte forma:

$$AC =< AgenteId, LH >$$

onde: *AgenteId* é um identificador único para o agente e
LH é uma lista indicando um conjunto finito de suas habilidades (*Hab*).

As habilidades, por sua vez, são definidas como:

$$Hab =< \text{Tipo} , \text{Cenário} , \text{Identificações} , \text{Representações} >$$

onde Tipo: é definido como sendo *Resolução*, *Diagnóstico* ou *Instrução*;
Cenário =< contexto , profundidade >;
Identificações =< $Ident_1, Ident_2, \ldots, Ident_i$ > é um conjunto no qual se atribui um nome para cada habilidade;
Representações =< *entrada*, *saída* >, onde se define o formato de entrada e saída pelo qual a habilidade pode ser acessada.

- *Conhecimento social* (CS): é a estrutura na qual o agente representa explicitamente o conhecimento sobre habilidades dos outros agentes tutores da sociedade. Quando o conhecimento do agente é insuficiente para resolver uma tarefa, o CS é acionado com o intuito de se identificar agentes que possam potencialmente engajar-se numa

cooperação a fim de resolver a tarefa em questão. Também trata-se de uma base de conhecimento e é definido pelo conjunto de AC's dos outros agentes tutores.

- *Alocação* (AL): é o módulo que é responsável pela seleção dos agentes, identificados como aptos a resolverem uma determinada tarefa, para a cooperação. A partir de uma lista de tarefas propostas pelo *Resolvedor*, o módulo AL opera sobre ela e produz como resultado uma lista de pares formados pela tarefa e sua correspondente lista de agentes aptos a executá-la (esta lista de agentes pode, eventualmente, ser vazia).

- *Coordenação* (Coord): realiza a sincronização das atividades cooperativas e opera sincronizado às execuções das subtarefas. É responsável pela interpretação da estrutura de tarefas (na qual se tem todas as tarefas e suas subtarefas, bem como a hierarquia para sua execução), determinando a cada momento a tarefa a ser repassada para o módulo de *cooperação*. Quando a tarefa não é decomposta em subtarefas, o Coord se torna dispensável, visto que ele se torna útil apenas para o tratamento das situações que envolvem tarefas decompostas.

- *Cooperação* (Coop): é o módulo responsável pelo gerenciamento dos diálogos cooperativos, ou pelas tentativas de assegurar a execução de uma tarefa. O Coop dispõe de um mecanismo para ativar uma instância de execução de um dado protocolo, no momento do encaminhamento da atividade cooperativa ao agente que vai cooperar. Em situações de conflito, ou seja, quando se tem mais de um agente apto a cooperar sobre uma dada tarefa, ele escolhe um dos agentes.

- *Manutenção* (Man): é o módulo responsável pelo gerenciamento dos diálogos de manutenção, ou seja a atualização dos CS's de forma que se garanta um funcionamento correto do AL. É importante que os agentes tutores da sociedade mantenham seus CS's atualizados, o que significa que devem estar cientes a respeito da entrada, saída ou atualização dos outros agentes da sociedade, para que não corram o risco de, por exemplo, contar com a cooperação de um agente que já não faz parte da sociedade.

O sistema de distribuição, por sua vez, é definido por dois módulos:

- *Controle*: módulo responsável pela interface entre o sistema social e o sistema de distribuição. Procura a ligação correta de mensagens às instâncias de diálogo, além de decidir sobre o destino adequado a uma determinada mensagem, dependendo do seu tipo (cooperação ou manutenção), utilizando-se de um mecanismo de apoio à tomada de decisão.

- *Comunicação*: responsável pelo interfaceamento entre o sistema de distribuição e o ambiente de comunicação, encarregando-se da distribuição e coleta das mensagens. Vale destacar que o referido ambiente de comunicação é comum a todos os agentes tutores da sociedade.

Existem outros módulos presentes quando se define um agente tutor que faz parte da SATA, porém destacou-se apenas os que foram julgados mais importantes dentro do contexto deste trabalho. Detalhes destas definições, bem como do funcionamento de cada um dos sistemas e seus módulos, podem ser encontrados em [COT97].

5.6. Interações entre Agentes Tutores

Na seção 5.3 discutiu-se de forma geral os tipos de interações existentes no ambiente MATHEMA. Agora, depois de analisar a arquitetura geral de um agente tutor, pode-se discutir com mais detalhes a interação *(ii)* entre agentes tutores na SATA.

No MATHEMA existe um módulo chamado *Protocolos* que é o responsável pela criação e ativação de um diálogo com base nos protocolos disponíveis. Os protocolos disponíveis definem um comportamento interativo, enquanto que sua instância é uma entidade com um contexto particular responsável pela sua execução. Existe um conjunto preestabelecido de protocolos que trata as situações de interação entre os agentes tutores na SATA.

A seguir apresentam-se as categorias de protocolos, definidas em [COT97], utilizadas no MATHEMA: os *protocolos de cooperação* e os *protocolos de manutenção*. Depois de apresentados os protocolos, comenta-se a respeito da linguagem de interação.

5.6.1. Protocolos de Cooperação

Os protocolos de cooperação regem as instâncias de protocolos que promovem as situações de cooperação e são ativados pelo módulo Coop. Com relação as situações de cooperação, existem dois tipos de diálogo: *mestre-escravo* e *licitação*.

Mestre-Escravo

O mestre-escravo ocorre quando o agente sabe com quem cooperar. Existem dois papéis definidos: o do mestre (contratante) e o do escravo (contratado). A tabela 5.1 apresenta, de forma resumida, os passos que ocorrem quando existe o mestre-escravo.

1. No agente mestre uma determinada instância de cooperação cria uma instância de diálogo relativa à situação de mestre-escravo. Este protocolo envia um PEDIDO para o agente que desempenhará o papel de escravo, sendo que a mensagem é enviada junto com a TAREFA a ser resolvida pelo agente que recebe o pedido.

2. No agente escravo outra instância irá tratar a situação de mestre-escravo, recebendo o pedido e repassando a mensagem, através do sistema de distribuição, para ser tratada adequadamente pelo sistema tutor.

3. O sistema tutor, por sua vez, tenta executar a tarefa para que disponha do RESULTADO obtido na execução e requerido pelo mestre, ou então de uma mensagem de ERRO que pode ter acontecido durante o processamento.

4. Depois disso, o escravo envia uma RESPOSTA para o mestre, sendo que nesta mensagem coloca-se o resultado ou o erro, quando for o caso.

5. Por fim, o mestre recebe a resposta e trata esta mensagem, encerrando-se assim o processo de cooperação.

Tabela 5.1: Passos resumidos para a situação de mestre-escravo

Dependendo da cooperação e de acordo com a tarefa, um mesmo agente pode exercer o papel de mestre e de escravo, de forma que uma tarefa possa ser realizada através de cooperações recursivas. Isto porque, dependendo do grau de complexidade da tarefa, esta precisa ser dividida em subtarefas e um mesmo agente pode não ter todas as habilidades requeridas para a resolução desta, e precise cooperar com o próprio agente que lhe pediu cooperação.

Licitação

É definida como um mecanismo que visa estabelecer uma interação de um agente com um grupo de agentes e é utilizada quando o agente não sabe com quem pode cooperar. A licitação irá ocorrer quando o módulo AL produz uma lista de seleção de agentes vazia, o que significa que o agente não identificou nenhum outro que poderia ajudá-lo em determinada situação. A tabela 5.2 apresenta os passos para o processo de licitação.

1. No lado gerente, uma determinada instância de cooperação irá criar uma instância de diálogo relativa à situação de licitação. Os parâmetros utilizados na criação são: a TAREFA para a qual se quer cooperação e uma lista com as IDENTIFICAÇÕES dos agentes que devem ser utilizados na licitação, caso esta lista seja vazia, todos os agentes da sociedade receberão a mensagem.

2. O protocolo envia um ANÚNCIO para os agentes, sendo que na mensagem enviada coloca-se a TAREFA a ser executada.

3. O sistema de distribuição de cada agente que recebe a mensagem, repassa o anúncio de forma que se consiga extrair a tarefa e repassá-la ao sistema tutor.

4. O sistema tutor analisa a tarefa para identificar quais das suas habilidades serão requeridas para a resolução desta. Quando a tarefa não puder ser resolvida pelo agente, tenta-se a divisão em subtarefas de forma que o agente identifique se pode ajudar pelo menos na solução de algumas destas subtarefas.

5. Caso haja decomposição da tarefa, o agente então quantifica sua resposta, dando um PARECER que é baseado na razão entre o número de subtarefas que o agente pode resolver sobre o número de subtarefas total, resultante da decomposição. Este fator será considerado como um indicativo de sucesso quando for maior do que zero. Caso não exista a decomposição, o agente colocará no parecer a possibilidade ou não de resolver aquela tarefa.

6. Por fim os agentes enviam PROPOSTAS, que são constituídas pela identificação do agente bem como pelo seu PARECER. O agente que pediu a cooperação fica recebendo estas propostas num determinado intervalo de tempo, e quando o limite de tempo expira, ordena a lista dos agentes que enviaram suas propostas com base no parecer dado por cada um.

Tabela 5.2: Passos resumidos para a situação de licitação

5.6.2. Protocolos de Manutenção

Os protocolos de manutenção regem as instâncias de protocolos que promovem as situações de manutenção e são ativados pelo módulo Man. A manutenção se faz necessária devido a importância de todos os agentes estarem sempre com os seus CS's atualizados. As

situações de manutenção que devem ser tratadas são: *entrada*, *saída* e *atualização* de um agente.

Entrada

Nesta situação de manutenção trata-se da entrada de agentes novos na SATA. Esta entrada deve consistir de uma apresentação, bem como de uma troca de informações a respeito da habilidade de cada agente, para que se atualizem os antigos agentes da SATA bem como o agente novo que está entrando. A tabela 5.3 apresenta os passos que consistem a entrada de um agente.

1. Quando um novo agente entra na sociedade, o sistema tutor pede ao sistema social, através do módulo de manutenção, que inicie o protocolo de entrada.
2. O agente que entrou envia uma APRESENTAÇÃO para todos os agentes da sociedade, colocando na mensagem o seu AC.
3. Cada agente da sociedade, por sua vez, atualiza o seu CS e responde enviando o seu próprio AC.
4. O agente que está entrando recebe, durante um certo intervalo de tempo, os AC's que estão sendo enviados pelos outros tutores da sociedade para cadastrá-los em seu CS.

Tabela 5.3: Passos resumidos para a situação de entrada

Saída

Com o mesmo objetivo da entrada, a situação de saída avisa à sociedade a remoção de um agente, de forma que todos os CS's sejam atualizados. A tabela 5.4 apresenta a situação de saída.

1. Quando um agente precisa sair da sociedade, antes de se efetuar a saída de fato, o sistema tutor pede ao sistema social, através do módulo manutenção, que inicie o protocolo de saída.
2. Cria-se uma instância de protocolo e é enviada uma mensagem de saída para todos os outros agentes da SATA.
3. Os agentes recebem a mensagem de saída e por fim atualizam o seu CS, retirando o modelo do agente que sai.

Tabela 5.4: Passos resumidos para a situação de saída

Atualização

A situação de atualização de um agente tutor é caracterizada através de uma saída seguida de uma reentrada. Desta forma, para entender seu funcionamento, basta seguir os passos da saída, seguidos dos passos correspondentes à entrada de um agente.

5.6.3. Linguagem de Interação

As interações entre os agentes ocorrem utilizando-se um mecanismo de comunicação, através de troca de mensagens. A estrutura das mensagens trocadas é fixa para que se facilite sua composição e interpretação. A figura 5.5 ilustra o formato de uma mensagem.

A mensagem é composta por três componentes mais gerais: *distribuição*, *social* e *tutor*. Cada um destes componentes é formado por atributos que possuem funções diferenciadas. A seguir tem-se a descrição de cada um destes parâmetros.

Figura 5.5: Formato da mensagem trocada entre agentes

Distribuição: nesta parte da mensagem são tratados os atributos de endereçamento do agente e identificação de diálogo, tanto para o emissor quanto para o receptor. É composta pelos seguintes campos:

- TAM: indica o *tamanho da mensagem*, para que o módulo comunicação tenha controle sobre o tamanho das mensagens recebidas;

- REM: deve ser preenchido com a identificação do agente tutor *remetente*. É importante esta informação para que o agente que receba a mensagem identifique seu remetente e quando for o caso, saiba para quem deve enviar uma resposta;

- DEST: é a identificação do agente tutor *destinatário*. Caso a mensagem seja destinada a mais de um agente, é necessário o preenchimento com mais de um identificador ou com a palavra chave TODOS, quando a mensagem for enviada para todos os agentes da sociedade;

- IDDLG_REM: é a *identificação da instância de diálogo do remetente* e deve ser preenchido com a identificação da instância de protocolo que remete a mensagem;

- IDDLG_DST: é a *identificação da instância de diálogo do destinatário*, preenchido com o número identificador da instância de protocolo do destinatário ao qual a mensagem também pertence.

Social: é a parte da mensagem composta por atributos úteis para a interpretação do conteúdo da mensagem. Possui os seguintes campos:

- PROT: tipo de *protocolo* utilizado na interação. Anteriormente foram discutidos estes protocolos que são mestre-escravo e licitação, referentes à cooperação e referentes à manutenção são a entrada, saída ou atualização de agentes;

- PRIM: são as *primitivas* utilizadas para contextualizar a interação dentro do protocolo. A tabela 5.5 apresenta estas primitivas bem como suas respectivas semânticas.

PRIMITIVA	SEMÂNTICA
PERGUNTA	Pedido de execução de uma tarefa.
RESPOSTA	Retorno de execução.
ANÚNCIO	Anúncio da tarefa.
PROPOSTA	Proposta de resolução de tarefa.
APRESENTAÇÃO	Apresentação dos agentes.
INFORMAÇÃO	Um agente informa que está saindo.

Tabela 5.5: Primitivas de Interação

A tabela 5.6. mostra a relação entre as situações existentes de acordo com os protocolos bem como a primitiva correspondente a cada um deles.

PROTOCOLO	SITUAÇÃO	PRIMITIVA
Cooperação	Mestre-Escravo	Pergunta
		Resposta
	Licitação	Anúncio
		Proposta
Manutenção	Entrada	Apresentação
	Saída	Informação

Tabela 5.6: Relação entre os protocolos, situações e primitivas

Tutor: nesta parte da mensagem se encontra o *conteúdo da aplicação*. Possui um único campo que é chamado CONTEÚDO.

O conteúdo da mensagem, dependendo da semântica, possui um formato especial com um significado particular. A tabela 5.7 mostra alguns exemplos para o preenchimento das interações nas situações existentes nas quatro situações: entrada, saída, mestre-escravo e licitação.

Tipo da Mensagem	REM	DEST	PROT	PRIM	CONTEÚDO
1	$Agente_i$	$Agente_j$	Cooperação	Pergunta	TAREFA
2	$Agente_j$	$Agente_i$	Cooperação	Resposta	ERRO
3	$Agente_j$	$Agente_i$	Cooperação	Resposta	RESPOSTA
4	$Agente_i$	GRUPO	Cooperação	Anúncio	HABILIDADE
5	GRUPO	$Agente_i$	Cooperação	Proposta	PARECER
6	$Agente_{novo}$	TODOS	Manutenção	Apresentação	Descrição do agente
7	$Agente_{saindo}$	TODOS	Manutenção	Informação	Identificação do agente

Tabela 5.7 :Resumo das interações de acordo com as situações existentes

Na tabela 5.7 os agentes estão colocados de forma generalizada (Agente$_i$, Agente$_j$, etc), e representam qualquer um dos agentes da SATA que estiver envolvido numa das interações exemplificadas.

Se houver uma mensagem do tipo 1, então poderão haver duas possibilidades de respostas: mensagem tipo 2 ou tipo 3. Se houver uma mensagem do tipo 4, haverá o tipo 5 como resposta, ou seja, o agente que pediu cooperação fica esperando do grupo, por um determinado tempo, uma resposta. Por fim, as mensagens do tipo 6 e 7 dizem respeito, respectivamente, a entrada e saída de agentes na SATA. Vale destacar que os números colocados para as mensagens (tipo 1, 2, etc) foram utilizados apenas para facilitar o entendimento da tabela 5.7, e não fazem parte da linguagem de interação.

CAPÍTULO 6

Sistema Tutor Multi-Agentes em Lógica

Este capítulo apresenta o sistema tutor multi-agentes desenvolvido no domínio da lógica, que teve como base o ambiente MATHEMA descrito no capítulo anterior. Para tal inicia-se definindo o modelo do domínio, a partir do modelo multidimensional proposto no MATHEMA. Após esta modelagem define-se a sociedade de agentes tutores artificiais (SATA).

Para concluir o capítulo, apresentam-se os aspectos gerais de implementação do sistema tutor, procurando descrever o agente tutor, bem como dar uma noção geral do funcionamento do sistema.

6.1. Considerações Iniciais

O sistema tutor multi-agentes proposto, do ponto de vista conceitual, teve por base o modelo teórico de ambiente interativo de aprendizagem denominado MATHEMA (descrito no capítulo 5), considerando-se principalmente seus princípios e a sua arquitetura.

O MATHEMA, além de definir uma arquitetura geral para sistema tutores, introduz uma estruturação para o conhecimento local de cada agente e para a organização destes agentes da sociedade em um espaço multidimensional.

Escolheu-se o domínio da lógica por se tratar de uma teoria matemática de grande importância, além de exercer influência na IA e nas ciências cognitivas em geral. É formal e permite explorar de forma abrangente o modelo proposto no ambiente MATHEMA. No apêndice tem-se algumas definições formalizadas de alguns conceitos da lógica que serão apenas citados neste capítulo.

Por outro lado, este domínio se mostra interessante e suficientemente representativo quando se define o modelo multidimensional de conhecimento, visto que possui características que facilitam o particionamento em subdomínios, levando-se em conta as dimensões para o conhecimento.

Nas próximas seções apresentam-se os passos que foram dados no sentido de se implementar o sistema tutor. Parte-se da modelagem do domínio, a seguir define-se a sociedade de agentes, e por fim apresenta-se a estrutura geral do programa para um agente específico.

6.2. Modelagem do Domínio em Lógica

A lógica possui uma longa história, de mais de 23 séculos, que remonta aos antigos filósofos gregos, principalmente Aristóteles, que estabeleceu os fundamentos da lógica de maneira sistemática [BIT96].

A lógica pode ser dividida em duas partes: a lógica clássica e a lógica não clássica. A lógica clássica apresenta um ponto de vista filosófico um tanto restrito, quando utilizada para representação de conhecimentos do mundo real. No entanto, existem formalismos que estendem a lógica clássica de maneira que se possa representar o conhecimento sobre o

mundo de forma mais adequada, no sentido de se ter um ponto de vista filosófico menos restrito, o que dá origem as lógicas não clássicas.

A lógica clássica pode ser analisada a partir da lógica de primeira ordem ou a partir da lógica proposicional. A lógica clássica de primeira ordem é mais abrangente, visto que a lógica clássica proposicional é um caso particular da lógica clássica de predicados, sendo uma linguagem na qual todos os símbolos de predicado têm *Aridade* zero [BIT96].

Por questões práticas, optou-se, na implementação do sistema tutor, pela lógica clássica. Para se fazer a modelagem do conhecimento sobre este domínio, levando-se em conta o modelo teórico proposto no MATHEMA, deve-se particioná-lo em diferentes subdomínios, de forma que se leve em conta as seguintes dimensões para o conhecimento: contexto, profundidade e lateralidade.

Com relação ao particionamento deste domínio, definiram-se os contextos, as profundidades e as lateralidades como descritos a seguir. A notação utilizada segue o que foi descrito na seção 5.4.

D = *Lógica Clássica*.

⇨ *Contextos:*

C_1 = Teoria de Modelos;

C_2 = Métodos Clássicos de Prova;

C_3 = Prova Automática de Teoremas.

⇨ *Profundidades:*

P_{i1} = Lógica Clássica de Primeira Ordem;

P_{i2} = Lógica Clássica Proposicional.

com $i = 1, 2, \ldots, 7$

⇨ *Lateralidades:*

L_{111} = Álgebra de Boole;

L_{121} = Método de Herbrand;

L_{311} = Prolog;

L_{312} = Método de Skolem;

L_{313} = Unificação.

Com base neste particionamento do domínio proposto, pode-se então partir para uma definição dos agentes tutores artificiais que irão compor a sociedade de agentes tutores artificiais (SATA).

6.3. Definição dos Agentes

Os agentes são definidos a partir da modelagem do domínio, que por sua vez consiste de subdomínios d_{ij} associados ao par $< C_i, P_{ij} >$ e de subdomínios d_{ijk} associados às lateralidades L_{ijk}. Define-se então os agentes tutores AT_{ij} associados aos subdomínios d_{ij} e os agentes tutores AT_{ijk} associados aos subdomínios d_{ijk}.

Na tabela 6.1 tem-se uma relação dos subdomínios com as respectivas identificações para cada agente tutor (AT).

Pares	Lateralidades	Subdomínios	Identificação do AT
$< C_1, P_{11} >$		d_{11}	AT_{11}
	L_{111}	d_{111}	AT_{111}
$< C_1, P_{12} >$		d_{12}	AT_{12}
	L_{121}	d_{121}	AT_{121}
$< C_2, P_{21} >$	-	d_{21}	AT_{21}

$< C_2, P_{22} >$	-	d_{22}	AT_{22}
$< C_3, P_{31} >$		d_{31}	AT_{31}
	L_{311}	d_{311}	AT_{311}
	L_{312}	d_{312}	AT_{312}
	L_{313}	d_{313}	AT_{313}
$< C_3, P_{32} >$	-	d_{32}	AT_{32}

Tabela 6.1: Definição dos agentes tutores

Cada agente tutor possui um domínio de conhecimento *d*. Este domínio é constituído por um conjunto de unidades pedagógicas, definidas em função de objetivos de ensino-aprendizagem específicos que estão associados a um *curriculum*.

De forma mais detalhada, tem-se a seguinte estrutura pedagógica para cada um deste domínio *d*:

- ao domínio *d* associa-se um *curriculum*;
- ao *curriculum*, associa-se um conjunto de unidades pedagógicas definidas de acordo com uma determinada ordem, relacionadas ao pré-requisito e ao nível de dificuldade. Cada unidade pedagógica é constituída por um conjunto de problemas;
- cada problema está associado a um conjunto de unidades de conhecimento que servem de apoio à solução deste problema;
- as unidades de conhecimento estão ligadas à conhecimentos de suporte para a resolução de problemas. Como exemplos de conhecimentos de suporte tem-se a apresentação ao aprendiz de conceitos, resultados, exemplos (como particularidades dos resultados), contra-exemplos, dicas, problemas análogos, catálogo de erros, dentre outros.

Os agentes podem ser reescritos de forma mais detalhada, levando-se em conta a existência das unidades pedagógicas associadas ao *curriculum*. Tem-se então os pares $< C_i, P_{ij} >$ reescritos:

$< C_1, P_{11} >$: {*sintaxe, semântica, tabela verdade*}

$< C_1, P_{12} >$: {*sintaxe, semântica*}

$< C_2, P_{21} >$: {*método axiomático, dedução natural, seqüência de Gentzen*}

$< C_2, P_{22} >$: { *método axiomático, dedução natural, seqüência de Gentzen*}

$< C_3, P_{31} >$: {*forma clausal, resolução, estratégias de prova*}

$< C_3, P_{32} >$: {*forma clausal, resolução, estratégias de prova*}

A seguir trata-se da definição da arquitetura interna de um agente tutor (AT).

6.3.1. Arquitetura Geral de um Agente

Os agentes tutores artificiais que formam a SATA do sistema tutor inteligente em lógica, foram definidos de acordo com a arquitetura geral de um AT, descrita na seção 5.5. No entanto, não foram implementados todos os sistemas e/ou módulos propostos no ambiente MATHEMA, visto que o objetivo maior era o de trabalhar com as interações entre os agentes na SATA.

Com relação a visão macro da arquitetura de um AT (ver seção 5.5.1), o MATHEMA propõe três sistemas: tutor, social e de distribuição. Investiu-se principalmente no sistema social, que é o responsável pelas interações entre os agentes na SATA, bem como no sistema de distribuição que coordena o envio e recebimento de mensagens.

O sistema tutor, que é o responsável pela interação agente-aprendiz, ainda não está completamente implementado. Ele possui um módulo principal chamado *Resolvedor de Problemas* (RP), que tem como objetivo trabalhar com as tarefas, no sentido de identificar

as que são de competência do agente resolver, bem como as que devem ser encaminhadas para outro, por não fazerem parte das habilidades do agente em questão. O módulo RP desempenha os papéis relacionados à geração e resolução de problemas, explicações e sugestões.

Tem-se três tipos de habilidades, que se referem às habilidades pedagógicas: *resolução*, *instrução* e *diagnóstico*.

- Resolução (R): esta habilidade diz respeito à capacidade que o agente possui de resolver problemas em geral;

- Instrução (I): diz respeito aos recursos pedagógicos, tais como: explanação sobre algum assunto, provimento de dicas e sugestões. Surge de alguma necessidade provocada no processo de resolução de problemas;

- Diagnóstico (D): tem como função avaliar e diagnosticar as soluções apresentadas pelo aprendiz, quando da resolução de problemas, por exemplo. Ao avaliar a solução proposta pelo aprendiz, recai-se em três situações: o agente sabe fornecer diagnóstico sozinho; o agente não sabe mas conhece quem sabe e por fim, na pior situação, o agente não sabe fornecer o diagnóstico e também não conhece quem possa lhe ajudar.

As habilidades estão armazenadas no arquivo de fatos referente ao agente em questão (implementado no FASE), e estão definidas da seguinte forma:

$$Hab =< \text{Tipo} , \text{Identificação} >$$

onde Tipo: é definido como sendo *Resolução*, *Diagnóstico* ou *Instrução*;
Identificação é o nome que se atribui para a habilidade.

Cada tipo de habilidade é implementada em regras que se encontram na base de regras do FASE. Vale destacar que uma mesma habilidade pode possuir mais de uma regra associada.

No que diz respeito a interação com o aprendiz, que é uma das funções do sistema tutor, não houve um investimento significativo. Desta forma, a interface se reduz à troca de mensagens entre o usuário e o sistema, sem a preocupação de se ter algo com uma apresentação agradável ou que tenha recursos como som e imagens. A própria linguagem de programação utilizada (Common-Lisp) reduz as possibilidades de investimentos neste recursos. No entanto, vislumbra-se algumas alternativas para solucionar este problema, como por exemplo, o desenvolvimento de uma interface utilizando a INTERNET.

O sistema social do agente, de forma geral, possui informações a respeito do conhecimento das habilidades do agente e a respeito do conhecimento social associado às habilidades dos outros agentes integrantes da SATA, além de realizar a seleção dos agentes identificados como aptos a cooperarem, quando o agente se encontra numa situação em que não consegue executar uma tarefa sozinho.

Como se trata de um protótipo no qual a distribuição é simulada, o módulo de manutenção, que faz parte do sistema social e tem por objetivo atualizar o conhecimento social quando da entrada ou saída de algum agente, não é utilizado e portanto não está implementado. Sua presença, no entanto, tornar-se-á fundamental quando houver uma distribuição real dos agentes.

Quanto ao envio e recebimento de mensagens, responsabilidade do sistema de distribuição, existe um *buffer* de mensagens, o chamado ambiente de comunicação do MATHEMA, que é varrido através de um *loop* feito no programa principal. Desta forma analisa-se todas as mensagens que estão neste *buffer*, e encaminha-se para seus respectivos destinatários, que serão os agentes ou o usuário.

O agente possui um módulo chamado MAIL que armazena as mensagens provenientes do ambiente de comunicação. Assim, quando a mensagem for identificada como sendo para o agente AT_{11}, por exemplo, esta é imediatamente armazenada no módulo MAIL do AT_{11}, que é em seguida executado.

A figura 6.1 representa, de forma geral e esquemática, a implementação de um agente tutor. Como foi discutido anteriormente, R, I e D são os tipos de habilidades: resolução, instrução e diagnóstico, respectivamente.

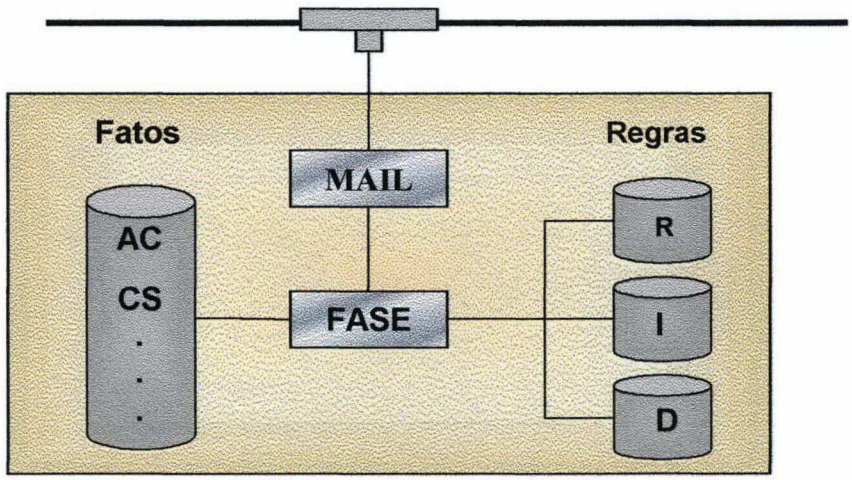


Figura 6.1: Implementação de um Agente Tutor

6.3.2. Interações entre Agentes Tutores

Os AT's na SATA interagem entre si no sentido de solucionar eventuais problemas quando da interação com o aprendiz. Conforme discutido na seção 5.6, existem duas categorias de protocolos utilizadas no MATHEMA: os protocolos de cooperação e os protocolos de manutenção.

As interações entre os agentes, como já foi comentado, se dá através da troca de mensagens, que são armazenadas no ambiente de comunicação (*buffer*).

A estrutura da mensagem é fixa e é implementada no Common-Lisp como uma estrutura que possui seis atributos: FROM, TO, PROTOCOL, SITUATION, PRIMITIVE e APPLICATION. Estes atributos formam três componentes mais gerais: distribuição, social e tutor (ver 5.6.3).

Baseando-se no formato de uma mensagem do MATHEMA, fez-se as adequações necessárias e a figura 6.2 ilustra o formato utilizado na implementação.

Distribuição		**Social**			**Tutor**
From	To	Protocol	Situation	Primitive	Application

Figura 6.2: Formato de uma mensagem

Com relação ao componente social, que é composto pelos atributos úteis para a interpretação do conteúdo da mensagem, tem-se três campos associados: protocol, situation e primitive. A relação entre estes três campos, bem como o conteúdo que podem assumir, podem ser visualizados na tabela 5.6 da seção 5.6.3.

Por se tratar de um protótipo com a distribuição simulada, não houve investimento nas interações que se utilizam dos protocolos de manutenção.

Os protocolos de cooperação, por sua vez, regem as situações de cooperação que podem gerar dois tipos de diálogo: mestre-escravo e licitação.

A licitação diz respeito às situações em que o agente não possui em seu conhecimento social dados suficientes para identificar quem pode lhe ajudar na resolução de alguma tarefa (ver 5.6.1). Estes protocolos não foram implementados até o momento.

Assim, tem-se os diálogos de mestre-escravo nos quais o agente identifica quais os agentes que podem lhe ajudar e envia a mensagem diretamente aos que possuem a habilidade de resolver determinada tarefa. A implementação destes protocolos é feita numa base de regras do FASE, que é executada quando o agente é ativado.

6.4. Aspectos Gerais da Implementação

Nesta seção descreve-se alguns aspectos referentes a implementação do sistema tutor inteligente em lógica. Utilizou-se para tal, a linguagem de programação Common-Lisp e o sistema FASE, que suporta as bases de regras de cada agente.

O programa principal pode ser dividido em três partes: (a) definições e inicializações; (b) função principal e (c) execução do agente e da interface com o usuário.

Num primeiro momento, quando se tem as definições e inicializações, trata-se das estruturas do agente e das mensagens. Com relação às mensagens, discutiu-se na seção anterior considerações a respeito de seus atributos.

A estrutura do agente, por sua vez, possui oito atributos: *name*, *mailbox*, *program*, *kb*, *rules-cd*, *rules-ng*, *rules-in*, *rules-out*. Desta forma, tem-se a seguir um exemplo da inicialização do agente AT_{11}.

```
(defvar *agents*
        (list (make-agent
              :name 'sintaxe
              :mailbox nil
              :program "sintaxe.lsp"
              :kb (kb-facts "sintaxe.f" :used t)
              :rules-cd (kb-rules "sintaxe-cd.r")
              :rules-ng (kb-rules "sintaxe-ng.r")
              :rules-in (kb-rules "sintaxe-in.r")
              :rules-out (kb-rules "sintaxe-out.r"))))
```

O programa *sintaxe.lsp* é implementado em Common-Lisp e define as funções que realizam as tarefas referentes as habilidades que o agente possui, e que são eventualmente chamadas a partir da execução de uma base de regras.

Define-se no arquivo de fatos (em :kb) as habilidades do agente, bem como alguns parâmetros que dizem respeito a execução de algumas regras. Para o agente exemplificado, tem-se os seguintes parâmetros referentes a algumas definições que o agente precisa ter em sua base de conhecimento:

```
((frame (alfabeto sintaxe
                (predicados (p q r s))
                (funcoes (f g h))
                (constantes (a b c d)))))
```

As habilidades, de forma genérica, são implementadas no arquivo *.f de cada agente, e possuem a seguinte forma geral:

```
((frame (habilidades <nome_do_agente>
      (resolucao (<lista de tarefas referentes à resolução>))
      (instrucao (<lista de tarefas referentes à instrução>))
      (diagnostico (<lista de tarefas referentes ao diagnóstico>)))))
```

Com relação à função principal, tem-se um *loop* que varre o *buffer* de comunicação com o intuito de distribuir as mensagens para seus respectivos destinatários. Quando a mensagem for para um agente, esta é armazenada no *mailbox* do mesmo, que é imediatamente executado. Se a mensagem for para o usuário, trata-se da execução de uma função que se encarregará do tratamento da mensagem, avaliando como se dará a interface. Isto porque dependendo da situação, esta mensagem pode ser uma pergunta ao usuário, uma resposta a um problema, uma instrução, etc.

O atributo *rules-cd* define o arquivo de regras para cada agente, na situação de mestre-escravo. Este arquivo, que é implementado no FASE, possui as regras que são testadas todas as vezes que o agente carrega o FASE. Da mesma forma, tem-se *rules-ng*, *rules-in* e *rules-out*, que definem os arquivos referentes a licitação, entrada e saída (que são os tipos de diálogos dos protocolos existentes), respectivamente. No entanto estes últimos, apesar de estarem definidos na estrutura do agente, ainda não estão implementados.

Ao receber uma mensagem, o agente avalia o seu conteúdo e monta a sua base de conhecimento, avaliando se possui condições de executar a tarefa sozinho ou se precisa da ajuda de outro agente. Se não houver necessidade de cooperação, o agente roda o FASE e monta a mensagem de saída, com a resposta da tarefa que precisava resolver.

Caso precise cooperar, ele monta a mensagem e a envia para o ambiente de comunicação, que por sua vez irá encaminhá-la ao destinatário. Desta forma, ele fica esperando a resposta do agente para quem pediu cooperação. Ao receber a mensagem de resposta, o agente consegue então montar a mensagem de saída.

A figura 6.3 apresenta um esquema resumido do funcionamento do sistema. O Resolvedor de Problemas pode receber entradas via usuário ou quando realiza a varredura no *buffer*. A cada interação o RP encaminha as mensagens para o agente requerido e coloca o agente ativo de forma que se possa gerar uma resposta ao usuário.

Um exemplo simples de interação entre o agente AT_{11} com o aprendiz pode surgir a partir de um questionamento do aprendiz a respeito do que significa um *termo* quando se trata da lógica clássica. Após encontrar em sua base de fatos a existência de conhecimentos sobre o que é um *termo*, o agente AT_{11} utiliza de sua habilidade de instrução para, num primeiro momento, definir a sintaxe da lógica clássica de 1ª ordem e em seguida exemplificar um *termo*.

Para verificar se o aprendiz entendeu o conhecimento exposto, o agente AT_{11} solicita ao mesmo um exemplo de um *termo*. Caso o aprendiz responda corretamente, o agente AT_{11} entende como concluída este interação, caso contrário, se o aprendiz responde ao que

lhe foi solicitado de forma incorreta, a interação prosseguirá com a análise do erro pelo agente e posterior apresentação de outro exemplo e/ou outra questão ao aprendiz. A tabela 6.2 apresenta um esquema resumido deste exemplo.

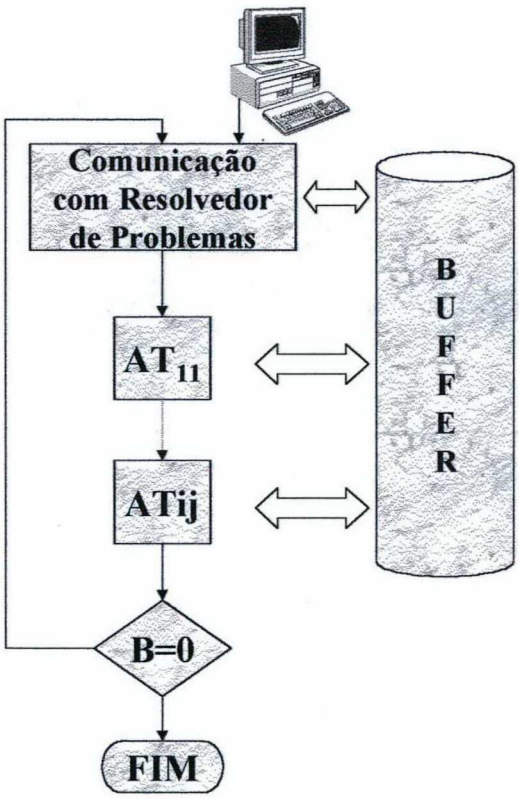


Figura 6.3: Esquema resumido do funcionamento do sistema

Aprendiz	Quero saber o que é um *termo*!
Agente busca na sua base de conhecimento: AT_{11}(*termo* , *fórmula* , *constante* , *alfabeto*)	
Agente AT_{11}	1. Define a sintaxe da Lógica Clássica de 1ª ordem (*instrução*) 2. Coloca um exemplo de um *termo* (*instrução*) 3. Solicita ao aprendiz um exemplo de um termo (*resolução*)
Aprendiz	P(a,b)
Agente AT_{11}	Este não é um *termo* pois P é *predicado*.

Tabela 6.2: Exemplo de uma interação com o aprendiz

6.5. Considerações Finais

O Sistema Tutor Multi-Agentes foi definido baseando-se no ambiente MATHEMA e a partir de considerações sobre o modelo do domínio que é a lógica clássica. No entanto, houveram módulos definidos no MATHEMA que não foram implementados por completo no sistema tutor em lógica.

Apesar disso, o sistema implementado se mostra bastante interessante, visto que foi concebido de forma que sua ampliação seja acessível e fácil de se realizar. Por outro lado, conseguiu-se definir os agentes de forma bastante genérica, o que significa que a inserção de outros agentes se torna facilitada.

Uma característica interessante deste sistema, é que cada agente tutor possui um sistema tutor inteligente associado, e este sistema é responsável por interagir com o aprendiz. Entretanto, cada agente pode precisar envolver outros agentes para atender aos seus objetivos locais e, assim o faz, devido à existência do sistema tutor multi-agentes.

CAPÍTULO 7

Conclusões e Perspectivas

O trabalho desenvolvido teve como objetivo geral a implementação de um sistema tutor multi-agentes, a partir de um ambiente interativo de apoio ao processo de ensino-aprendizagem.

Para tal, partiu-se do estudo do ambiente MATHEMA, de forma que se pudesse adequá-lo ao domínio escolhido – a lógica clássica. A escolha do MATHEMA pode ser justificada quando se analisa sua contribuição em termos de abrangência na proposta de distribuição de papéis em múltiplos agentes. Contribui ainda, de forma relevante, quando define o domínio a partir de um modelo multidimensional, conseguindo tratar um conteúdo específico de forma organizada e buscando de uma forma geral aspectos de qualidade para o sistema.

Como metodologia utilizada, partiu-se da pesquisa bibliográfica sobre os Sistemas Tutoriais, Inteligência Artificial Distribuída e questões relacionadas à Informática na Educação. Em paralelo, desenvolveu-se o estudo e a familiarização com o ambiente Common-Lisp em geral, e com o sistema FASE em particular. Neste sentido, houve o desenvolvimento de um pequeno sistema especialista que analisava funções polinomiais de grau 0, 1 ou 2.

Em seguida houve um estudo mais detalhado do ambiente MATHEMA com o intuito de especificar um sistema tutor no domínio da geometria [CBF97] [FBC97]. Este protótipo serviu de base para o desenvolvimento do sistema no domínio da lógica.

Houveram algumas limitações durante a implementação, como por exemplo, no que diz respeito à interface com o usuário. Sabe-se que é necessário, durante o desenvolvimento de qualquer *software* educacional, que existam reflexões no sentido de ser ter um sistema acessível e que capte a atenção do aluno e do professor, sem que existam prejuízos no que diz respeito à apresentação do conteúdo. A utilização do *software* pode se tornar catastrófica caso não se tenha isto em mente.

Como foi comentado no decorrer do trabalho, não houve este investimento na interface com o usuário, o que não significa que estes aspectos foram ignorados. Ao contrário, visualiza-se algumas soluções e perspectivas futuras neste contexto, como por exemplo, a utilização dos recursos disponíveis na INTERNET.

Outro aspecto que deve ser comentado é com relação à modelagem do aluno. Sabe-se que o sistema tutor inteligente deve possuir um modelo do aprendiz, de forma que possa trabalhar individualmente de acordo com as necessidades de cada um. O sistema proposto possui condições suficientes de armazenar as respostas dadas pelo aluno, bem como os passos que já foram seguidos pelo mesmo quando de sua utilização. Desta forma, pode-se iniciar um processo de modelamento de cada aluno, partindo dos referenciais citados.

Investiu-se basicamente na concepção de uma arquitetura de um agente tutor (seguindo as definições dadas no MATHEMA), de forma que não se tenha dificuldades, no que diz respeito à programação, para a inserção de outros agentes na sociedade.

Além disso, buscou-se a implementação das interações entre os agentes na SATA. Apesar destes agentes não estarem geograficamente distribuídos, o funcionamento do sistema em uma situação real de distribuição não deve apresentar maiores dificuldades.

Como contribuição deste trabalho tem-se a possibilidade da utilização do sistema tutor multi-agentes em disciplinas que discutam aspectos da lógica, ou até mesmo por alunos interessados no tema. Além disso, pode-se dizer que a utilização do ambiente MATHEMA como base conceitual ao desenvolvimento de qualquer sistema tutor multi-agentes é promissora, principalmente no que diz respeito à modelagem do domínio e à definição da estrutura dos agentes.

Por fim o sistema tutor desenvolvido ainda não está consolidado, ao contrário é um ponto de partida para o desenvolvimento de trabalhos de pesquisa na área de tutores no domínio da lógica. Além disso, existem alguns aspectos que precisam ser melhorados ou até mesmo inseridos no sistema.

De forma resumida, tem-se algumas perspectivas futuras que podem ser apontadas:

- Tratar a interface com o aprendiz de forma que seja mais fácil e agradável a utilização do sistema;
- Testar o funcionamento do programa com agentes distribuídos em locais geograficamente diferentes utilizando a INTERNET, de forma que o sistema se torne acessível e assim possa ser testado quando utilizado por vários aprendizes;
- Utilizar o sistema em sala de aula, depois de se investir mais na interface, para que se avalie o comportamento dos alunos quando da sua utilização;
- Implementar os módulos que não foram considerados, como por exemplo, a atualização do conhecimento social de cada agente quando da entrada ou saída de agentes da sociedade;
- Implementar as interações entre a sociedade de especialistas humanos (SEH) e a sociedade de agentes tutores artificiais (SATA), ambas propostas no MATHEMA. Desta forma, pode-se ampliar os meios para melhorar a qualidade do conhecimento da base de dados, buscando-os através de meios externos.

Referências Bibliográficas

[BAL94] BALACHEFF, N. e VIVET, M. Didactique et Intelligence Artificial. *Recherches en Didactique des Mathématique*. La Pensée Sauvage – Editions, Vol. 14/1.2 1994.

[BAR82] BARR, A. e FEIGENBAUM, E. A. *The Handbook of Artificial Intelligence*. Morgan Kaufman, Los Altos, CA, 1982.

[BAU90] BAULAC, Y. *Un micro-monde de géométroe, Cabri-géomètre*. Tese (Doutorado) – IMAG/LSDD, Université Joseph Fourier, Grenoble, França, Fevereiro, 1990.

[BB75] BROWN, J. S. e BURTON, R. R. Multiple representation of knowledge for tutorial reasoning. *Representation and Understanding: Studies in Cognitive Science*, D.G. Bobrow and A Collins editors, p. 311-349. Academic Press, New York, 1975.

[BIT95] BITTENCOURT, G. Um ambiente para ensino e desenvolvimento de sistemas especialistas. *III Workshop sobre Educação em Informática /IV Congresso Íbero-Americano de Educação Superior em Computação*, Canela, Agosto, 1995.

[BIT96] BITTENCOURT, G. *Inteligência Artificial – Ferramentas e Teorias*. Instituto de Computação, UNICAMP, São Paulo, 1996.

[BM93] BITTENCOURT, G. e MARENGONI, M. A customizable tool for the generation of production-based systems. *Eighth International Conference on Applications of Artificial Intelligence in Engineering (AIENG'93)*. Vol.1: Design, Methods and Techniques, p. 337-352, Tolouse, Julho, 1993.

[BRE96] BRENNER, M. F. M. *Uma Arquitetura para Agentes Inteligentes baseada na Sociedade da Mente*. Dissertação (Mestrado em Ciência da Computação) – Instituto de Computação, UNICAMP, Campinas, Dezembro, 1996.

[BRO86] BROOKS, R. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, p. 435-453, Março, 1986.

[CAM95] CAMPOS, G.H.B. Tecnologia, educação formal e qualidade de software. *CEMA–Caderno de Educação Matemática*. Vol II, p. 122-149, São Paulo, 1995.

[CAM96] CAMPOS, G.H.B. Avaliação da Qualidade de Software Educacional – notas de acompanhamento de curso. *Programa de Engenharia de Sistemas e Computação – Informática na Educação*. COPPE/UFRJ, Rio de Janeiro, Setembro, 1996.

[CB97] COSTA, A. C. P. L. e BITTENCOURT, G. Expert-Coop: An Environment for Cognitive Multi-Agent Systems. *Proceedings of the IFAC/IFIP Conference on Management and Control of Production and Logistics (IFAC/IFIP – MCPL'97)*. p. 492-497, Campinas, 1997.

[CB97a] COSTA, A. C. P. L. e BITTENCOURT, G. Parla: A Cooperation Language for Cognitive Multi-Agent Systems. *Proceedings of the 8th Portuguese Conference on Artificial Intelligence (EPIA'97), Springer-Verlag*, Coimbra, Portugal, 1997.

[CBF97] COSTA, E. B., BITTENCOURT, G. e FLEMMING, E. Um Sistema Tutor Multi-Agentes em Geometria. *Anais do VIII Simpósio Brasileiro de Informática na Educação (SBIE'97) / I Workshop de Sistemas de Tutoria Inteligente Aplicados à Educação e Treinamento*, Vol. II, p. 257-270, São José dos Campos, Novembro, 1997.

[CHA95] CHAIB-DRAA, B. *Distributed Artificial Intelligence: An Overview*. Sainte-Foy, Québec, Canada, 1995.

[CM85] CHARNIAK, E. e McDERMOTT, D. *Introduction to Artificial Intelligence*. Addison-Wesley Publishing Company, Reading, MA, 1985.

[COS97] COSTA, A.C.P.L. *Expert-Coop: Um ambiente para desenvolvimento de Sistemas Multi-Agentes Cognitivos*. Dissertação (Mestrado em Engenharia Elétrica) – Departamento de Engenharia Elétrica, Universidade Federal de Santa Catarina, Florianópolis, Junho, 1997.

[COT97] COSTA, E. B. *Um Modelo de Ambiente Interativo de Aprendizagem Baseado numa Arquitetura Multi-Agentes*. Tese (Doutorado em Ciências) – Departamento de Engenharia Elétrica, Universidade Federal da Paraíba, Campina Grande, Dezembro, 1997.

[CP96] COSTA, E.B. e PERKUSICH, A. Modelling the Cooperative Interactions in a Teaching/Learning Situation. *In Proceedings of Third International Conference on Intelligent Tutoring Systems - ITS'96*.Montreal, Canada, Junho, 1996.

[CP97] COSTA, E.B. e PERKUSICH, A. A Multi-Agent Interactive Learning Environment Model. *Proceedings of Artificial Intelligence in Education - Workshop on Pedagogical Agents*. Kobe, Japão , Agosto, 1997.

[CPJ97] COSTA, E.B., PERKUSICH, A. e JATOBÁ, A. A. Arquitetura e Protocolos para Cooperação entre Agentes em um Ambiente Interativo de Ensino e Aprendizagem. *A ser publicado nos anais do I Encontro Nacional de Inteligência Artificial (ENIA97)*. Brasília, Agosto, 1997.

[CUR96] CURY, D. *FLAMA: Ferramentas e Linguagem de Autoria para a Modelagem da Aprendizagem*. Tese (Doutorado em Ciências) – Curso de Engenharia Eletrônica e Computação, Instituto Tecnológico de Aeronáutica, São José dos Campos, Agosto, 1996.

[DEM93] DEMAZEAU, Y. La plate-forme paco et ses applications. *Actes de la 2ème Journée Nationale du PRC-GDR Intelligence Artificielle*. França, 1993.

[DRU93] DRUCKER, P. F. *Post-Capitalist Society*. HarperCollins, New York, 1993.

[DLC89] DURFEE, E. H., LESSER, V. R., e CORKILL, D. D. Trends in cooperative distributed problem solving. *IEEE Transactions on Knowledge and Data Engineering*, p. 63-83, Março, 1989.

[DP88] DUBOIS, D e PRADE, H. *Possibility Theory – An Approach to the Computerized Processing of Uncertainty*. Plenum Press, 1988.

[DR94] DURFEE, E. H. e ROSENSCHEIN, J. S. Distributed problem solving and multi-agent systems: Comparisons and examples. *Proceedings of the International Workshop on Distributed Artificial Intelligence*, Julho, 1994.

[DU94] DUARTE, J.A.O. *O computador na educação matemática: percursos de formação*. Dissertação (Mestrado em Educação) – Departamento de Educação da Faculdade de Ciências, Universidade de Lisboa, Lisboa, Portugal, Janeiro, 1994.

[EAD95] EKDAHL, B., ASTOR, E. e DAVIDSSON, P. Towards Anticipatory Agents. *Lecture Notes in Artificial Intelligence*, Intelligent Agents Theories, Architectures and Languages, p. 191-202, Springer Verlag, 1995.

[FBL71] FEIGENBAUM, E. A., BUCHANAN, B. G. e LEDERBERG, J. On Generality and problem solving: a case study using the dendral program. *Machine Intelligence*, vol. 6, p. 165-190, B. Meltzer e D. Michie editors, Edimbourgh, Gb, 1971

[FBC97] FLEMMING, E., BITTENCOURT, G. e COSTA, E. B. Desenvolvimento de um Sistema Tutor Distribuído no Domínio da Geometria Plana. *Anais do XX Congresso Nacional de Matemática Aplicada e Computacional – CNMAC*, p. 201-202, Gramado, Setembro, 1997.

[FER94] FERRETTI, C. J. et al (org). *Novas tecnologias, trabalho e educação: um debate multidisciplinar*. Vozes, Petrópolis, 1994.

[FG91] FERBER, J. e GASSER, L. Intelligence artificielle distribuée. *Tutorial Notes of the 11th Conference on Expert Systems and their Applications*. França, 1991.

[FG96] FRANKLIN, S. e GRAESSER, A. Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents. *Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages*. Springer-Verlag, 1996.

[GAS92] GASSER, L. Boundaries, identity and aggregation: Plurality issues in multiagent systems. *Descentralized Artificial Intelligence*. Eric Werner and Yves Demazeau editors, p. 199-212, Amsterdam, 1992.

[GIU85] GIUSTA, A.S. Concepção de aprendizagem e práticas pedagógicas. *Educação em Revista*. N.1, Belo Horizonte, 1985.

[HUN87] HUNS, M. *Distributed Artificial Intelligence*. Morgan-Kauffman, 1987.

[JEN93] JENNINGS, N. R. Commitments and conventions: the foudations of coordination in multi-agent systems. *The Knowledge Engineering Review*. P. 223-250, 1993.

[JS92] SICHMAN, J.S, DEMAZEAU, Y. e BOISSUER, O. When can knowledge-based system be called agents? *Anais do IX Simpósio Brasileiro de Inteligencia Artificial*, p. 172- 185, Rio de Janeiro, 1992.

[KEA87] KEARSLEY, G. (eds). *Artificial Intelligence and Instruction: applications and methods*. Addison-Wesley Publishing Company, 1987.

[MAR] MARIETTO, M. G. B. *Tendências nas Áreas de Sistemas Tutores Ingeligentes e Modelagem do Estudante*. Divisão de Ciência da Computação, Instituto Tecnológico de Aeronáutica, São José dos Campos.

[MCC79] McCORDUCK, P. *Machines Who Think*. Freeman, San Francisco, 1979.

[MH69] McCARTHY, J. e HAYES, P.J. Some philosophical problems from the standpoint of artificial intelligence. *Machine Intelligence 4*. D. Michie and B. Meltzer editors, p. 463-502, Edimbourgh University Press, Edimbourgh, 1969.

[MIN85] MINSKY, M. *The Society of Mind*. Simon and Schuster, New York, 1985.

[MLR93] MENDONÇA, L.F. e ROCHA, A.R.C. Avaliação de hipertextos. *Relatório Técnico do Programa de Engenharia de Sistemas e Computação*. COPPE/UFRJ, Rio de Janeiro, Abril, 1993.

[MOR86] MOREIRA, M. O uso do computador na Educação: pressupostos psicopedagógicos. *Educação em Revista*. N. 4, p. 13-17, Belo Horizonte, 1986.

[NEW80] NEWELL, A. Physical symbol systems. *Cognitive Science*, p. 135-183, 1980.

[NIL80] NILSSON, N. J. Two heads are better than one. *Sigart Newsletter*, 1980.

[NIL82] NILSSON, N. *Principles of Artificial Intelligence*. Springer-Verlag, 1982.

[PAP85] PAPERT, S. LOGO: *Computadores e Educação*. Ed. Brasiliense, São Paulo, 1985.

[PEA84] PEARL, J. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley, 1984

[PIA78] PIAGET, J. *Fazer e Compreender*. Melhoramentos: editora da Universidade de São Paulo, São Paulo, 1978.

[RAB95] RABUSKE, R.A. *Inteligência Artificial*. Editora da UFSC, Florianópolis, 1995.

[RIC83] RICH, E. *Artificial Intelligence*. McGraw-Hill Book Company, 1983.

[RK86] ROSENSCHEIN, J. S. e KAELBLING, L. P. *The systhesis of digital machines with provable epistemic properties*. Theoretical Aspects of Reasoning about Knowledge, p. 83-98, Los Altos, CA, 1986.

[SAN93] SANTOS, N. Computadores na Educação: discutindo alguns pontos críticos. *Em Aberto*. Ano 12, n. 57, p. 27-31, Brasília, 1993.

[SEL88] SELF, J. *Artificial Intelligence and Human Learning*. Chapman Hall, Londres, 1988.

[SHO76] SHORTLIFFE, E. H. *Computer-Based Medical Consultations: Mycin*. American Elsevier, New York, 1976.

[SJ84] STEELE Jr., G. L. *Common Lisp: The Language*. Digital Press, Burlington, 1984.

[SRD97] SANDHOLTZ, J. H., RINGSTAFF, C. e DWYER, D. *Ensinando com tecnologia: criando salas de aula centradas nos alunos.* Artes Médicas, Porto Alegre, 1997.

[STA90] STAHL, M.M. Software Educacional: características dos tipos básicos. *Anais do I SBIE – Simpósio Brasileiro de Informática na Educação.* P. 34-46, Rio de Janeiro, 1990.

[TB96] TONIN, I. e BITTENCOURT, G. LOGIK: Um Ambiente para o Ensino de Lógica. *V Congresso Iberoamericano de Educacion Superior en Computacion (EDUC'96).* Cidade do México, Setembro, 1996.

[VAL93] VALENTE, J. A. Diferentes Usos do Computador na Educação. *Núcleo de Informática Aplicada à Educação (NIED).* Gráfica Central da UNICAMP, p. 1-23, São Paulo, 1993.

[VAL95] VALENTE, J. A. Informática na educação: conformar ou transformar a escola. *Perspectiva.* Núcleo de Publicações – CED/UFSC, ano 13, n.24, p. 41-49, Florianópolis, Julho-Dezembro, 1995.

[VIC90] VICCARI, R. *Um Tutor Inteligente para a Programação em Lógica – Idealização, Projeto e Desenvolvimento.* Tese (Doutorado), Universidade de Coimbra, Coimbra, 1990.

[VIG96] VICCARI, R.M. e GIRAFFA, L.M.M. Sistemas Tutores Inteligentes: abordagem tradicional x abordagem de agentes. *Anais do XIII SBIA – Simpósio Brasileiro de Inteligência Artificial*, Curitiba, 1996.

[WAZ97] WAZLAWICK, R. Agentes Autônomos e Teoria da Equilibração Cognitiva. [online]. Disponível pela Internet via WWW <URL: http://www.inf.ufsc.br/~raul>,

[WEG87] WENGER, E. *Artificial Intelligence and Tutoring Systems*. Morgan Kaufman, Los Altos, CA, 1987.

[WIN84] WINSTON, P.H. *Artificial Intelligence (2nd Edition)*. Addison-Wesley Publishing Company, MA, 1984.

APÊNDICE
Conceitos sobre Lógica

1. Introdução

A lógica possui uma longa história (mais de 23 séculos), que remonta aos antigos filósofos gregos, dentre os quais destaca-se Aristóteles por estabelecer os fundamentos da lógica de maneira sistemática. Suas aplicações vão desde a fundamentação teórica de diversas áreas da matemática até a representação de conhecimento em IA.

De maneira geral, um sistema lógico consiste em um conjunto de fórmulas e um conjunto de regras de inferência. As fórmulas são sentenças pertencentes a uma linguagem formal cuja sintaxe é dada. Cada fórmula pode ser associada a um *Valor Verdade*, isto é, ao valor *verdadeiro* ou ao valor *falso*. A parte da lógica que estuda os valores verdade é chamada *Teoria dos Modelos*.

Uma regra de inferência é uma regra sintática que, quando aplicada repetidamente a uma ou mais fórmulas verdadeiras, gera apenas novas fórmulas verdadeiras. As regras de inferência fornecem uma estrutura dedutiva à linguagem lógica. A seqüência de fórmulas geradas através da aplicação de regras de inferência sobre um conjunto inicial de fórmulas é chamada de *prova*. A parte da lógica que estuda as provas é chamada de *Teoria das Provas*.

A partir da introdução, por Robinson e Smullyan em 1960, de procedimentos eficientes para demonstração automática de teoremas por computador, a lógica passou a ser estudada também como método computacional para a solução de problemas. Ambos os procedimentos exploram o fato de expressões lógicas poderem ser colocadas em *formas canônicas*, isto é, restringe-se o número de operadores e especifica-se a sintaxe a ser respeitada. O resultado, embora não seja adequado para ser manipulado por seres humanos devido a perda da estrutura original das expressões lógicas, permite uma manipulação computacional bastante eficiente.

O fato de ser possível associar uma semântica operacional a um procedimento de prova automática de teoremas, permitiu a definição de uma linguagem de programação baseada em lógica, a linguagem Prolog. Esta linguagem, inicialmente restrita a laboratórios de pesquisa, hoje é amplamente utilizada, e seus aperfeiçoamentos e possíveis extensões são objetos de intensas pesquisas.

Nas próximas seções, tem-se algumas características e propriedades da lógica clássica de 1ª ordem e da lógica clássica proposicional, que foram utilizadas para o domínio modelado do sistema tutor multi-agentes. Convém destacar que o embasamento teórico foi obtido a partir do livro *Inteligência Artificial – Ferramentas e Teorias* [BIT96], escrito pelo Prof. Guilherme Bittencourt.

2. Sintaxe

Formalmente, uma *Linguagem Lógica de Primeira Ordem* – $L(\mathbf{P},\mathbf{F},\mathbf{C},\mathbf{V})$ – é determinada pela especificação dos seguintes conjuntos:

- Um conjunto **P** de Símbolos de Predicado;
- Um conjunto **F** de Símbolos de Função;
- Um conjunto **C** de Símbolos de Constante;
- Um conjunto **V** de Símbolos de Variável.

A cada símbolo de predicado e de função é associada uma *Aridade*, isto é, o número de argumentos do predicado e da função. Os símbolos de predicado com aridade zero são chamados *Símbolos Proposicionais*, além disto os símbolos de constante podem ser também considerados como funções de aridade zero. Estes conjuntos formam o *Alfabeto* da linguagem lógica, a *Sintaxe* da linguagem pode ser definida através da seguinte linguagem formal:

$$\langle termo \rangle ::= \langle \text{var}\, iável \rangle \mid \langle cons\,\text{tan}\, te \rangle \mid \langle função \rangle \ (\langle termo \rangle_1, \ldots, \langle termo \rangle_n)$$

$$\langle fórmula\ atômica \rangle ::= V \mid F \mid \langle predicado \rangle \ (\langle termo \rangle_1, \ldots, \langle termo \rangle_n)$$

$$\langle fórmula \rangle ::= \langle fórmula\ atômica \rangle \mid \neg\ (\langle fórmula \rangle) \mid (\langle fórmula \rangle_1 \wedge \langle fórmula \rangle_2) \mid (\langle fórmula \rangle_1 \vee \langle fórmula \rangle_2) \mid (\langle fórmula \rangle_1 \rightarrow \langle fórmula \rangle_2) \mid (\forall x.(\langle fórmula \rangle) \mid (\exists x.\langle fórmula \rangle))$$

⟨constante⟩:: = *a,b,c,d* e outras palavras quaisquer iniciadas por minúsculas.
⟨variável⟩:: = *x,y,z* com ou sem índices .
⟨função⟩:: = *f,g,h* e outras palavras quaisquer iniciadas por minúsculas.
⟨predicado⟩:: = *P,Q,R* e outras palavras quaisquer iniciadas por maiúsculas.

onde, os símbolos V e F, são chamados, respectivamente, de *Verdadeiro* e *Falso*. Os símbolos de operadores lógicos têm os seguintes nomes: $\neg$ "não", $\wedge$ "e", $\vee$ "ou", $\rightarrow$ "implica", $\forall$ "qualquer", $\exists$ "existe". Os símbolos $\forall$ e $\exists$ são chamados ainda *Quantificadores.*

3. Semântica

A maneira de definir o significado da linguagem lógica foi proposta por Tarski e consiste em associar os elementos sintáticos da linguagem a estruturas matemáticas da teoria de conjuntos.

Para tal, define-se uma *interpretação* como um par formado pelos seguintes elementos:

- Um domínio D, um conjunto não vazio.
- Uma função de associação ξ que leva cada elemento da sintaxe da linguagem em uma estrutura matemática definida sobre o conjunto D.

A função de associação é definida da seguinte maneira:

- Cada símbolo de constante é associado a um elemento de D;
- Cada símbolo de função de aridade n é associado a uma função de D^n em D, onde D^n representa o produto cartesiano do conjunto D com ele mesmo n vezes;
- Cada símbolo de predicado de aridade n é associado a uma relação de aridade n contida em D^n.

4. Métodos de Prova

Um problema interessante no estudo da lógica, consiste em determinar se, dado um conjunto G de fórmulas, uma fórmula W é ou não uma *Conseqüência Lógica* de G, isto é, se toda interpretação que satisfaz as fórmulas em G, simultaneamente, satisfaz também a fórmula W. Esta relação entre fórmulas é notada por:

$$G \models W$$

Este problema é mais geral do que a determinação de valores verdade pois não depende de nenhuma interpretação em particular, sendo antes uma relação entre as fórmulas pertencentes ao conjunto $G \cup \{W\}$. Sua interpretação intuitiva é análoga ao "raciocínio hipotético" humano, isto é, a capacidade de assumir certas hipóteses como verdadeiras e considerar as conseqüências.

A dificuldade em relação a este problema vem do fato de que a definição de conseqüência lógica envolve *todas* as interpretações, que são em número infinito. Além

disto a definição não oferece um método operacional capaz de determinar se uma fórmula é ou não conseqüência lógica de um conjunto de fórmulas.

A solução encontrada foi o desenvolvimento de *Métodos de Prova* tais que, a partir do conjunto G e utilizando certas *Regras de Inferência*, seja possível gerar novas fórmulas, eventualmente levando à geração da fórmula W. Uma regra de inferência é simplesmente uma função sintática que, dado um conjunto de fórmulas lógicas, gera uma nova fórmula:

$$\rho_i(\{H_1, H_2, \ldots, H_n\}) = H$$

Se a partir de fórmulas verdadeiras uma regra de inferência produzir apenas fórmulas verdadeiras, a regra é dita *Correta*. Por serem sempre verdadeiras, tautologias, quando utilizadas como base para regras de inferência, levam necessariamente a regras corretas. O que permite que tautologias sejam utilizadas como regras de inferência é uma outra regra de inferência, mais geral, conhecida como *Regra da Substituição*, que afirma que uma tautologia formada por símbolos proposicionais permanece uma tautologia quando estes símbolos são substituídos por fórmulas lógicas arbitrárias. Sejam A e B fórmulas quaisquer, as seguintes tautologias são freqüentemente utilizadas como regras de inferência:

$(A \wedge (A \rightarrow B)) \rightarrow B$	*Modus Ponens*
$(\neg B \wedge (A \rightarrow B)) \rightarrow \neg A$	*Modus Tollens*
$((A \rightarrow B) \wedge (B \rightarrow C)) \rightarrow (A \rightarrow C)$	*Silogismo Hipotético*
$\forall x.A \rightarrow A\{x/a\}$	*Especialização*
$A\{x/a\} \rightarrow \exists x.A$	*Generalização*

Caso exista realmente uma seqüência de aplicações de regras de inferência que leve das fórmulas de G em W, então diz-se que W pode ser *Provado* a partir de G, e nota-se:

$$G \vdash W$$

Uma prova de W a partir de G sempre pode ser colocada na forma de seqüência de fórmulas $H_1, H_2, \ldots, H_n$ tal que, $H_n = W$ e:

$$H_i \in G \text{ ou } H_i = \rho_i(\{H_1, H_2, \ldots, H_{i-1}\})$$

onde ρ_i é uma regra de inferência aceita pelo método de prova.

Caso o método de prova seja correto, isto é, contenha apenas regras de inferência corretas, então:

$$G \vdash W \Rightarrow G \models W$$

Caso o reverso desta expressão, $G \vdash W \Rightarrow G \models W$, seja também verdadeiro, isto é, se, para toda a fórmula W, conseqüência lógica de G, seja possível encontrar uma prova de W a partir de G, então o método de prova é dito *Completo*.

Atualmente, existem diversos métodos de prova corretos e completos. A maioria foi criada visando sua utilização por seres humanos, por exemplo, *Sistemas de Axiomas*, *Dedução Natural* e *Seqüentes de Gentzen*, no entanto, a partir de 1960, diversos métodos voltados para a prova automática por computador foram propostos, como por exemplo, o *Método da Resolução* e o *Método de Tableaux*.

5. Método de Resolução

O método de resolução é um método de *Refutação*: para provar que $G \models W$, prova-se que $H = G \cup \{\neg W\}$ é insatisfazível, isto é, que não existe nenhuma interpretação que satisfaça simultaneamente todas as fórmulas de H.

Para aplicar o método de resolução é necessário inicialmente transformar as fórmulas do conjunto H para a *Forma Normal Conjuntiva*:

$$H = C_1 \wedge \ldots \wedge C_n$$
$$C_i = L_{i,1} \vee \ldots \vee L_{i,mi}$$

onde, C_i são *Cláusulas* e $L_{i,j}$ são *Literais*, isto é, fórmulas atômicas negadas ou não.

6. Linguagem PROLOG

A linguagem de programação Prolog (*Programming in Logic*) é fruto de uma das mais antigas linhas de pesquisa em Inteligência Artificial – a Prova Automática de Teoremas. O primeiro programa de IA já tinha por objetivo a prova automática de teoremas (o *Logic Theorist* desenvolvido por Newell, Shaw e Simon).

A linguagem Prolog também é um provador automático de teoremas, no qual a estratégia de escolha de cláusulas adotada é a estratégia SLD (*Selective Linear Resolution for Definite Clauses*). Além desta característica, esta linguagem também possui outra peculiaridade importante que é a necessidade do uso das *Cláusulas de Horn*, de forma que não se pode utilizar qualquer tipo de cláusula.