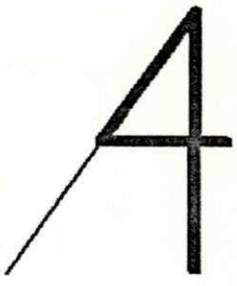
Universidade Federal da Paraíba
Centro de Ciências Tecnológicas
Departamento de Sistemas e Computação



Um Ambiente de Apoio à Aquisição Automática de Conhecimento

por

João José Peixoto Furtado - Vasco

Orientador: Giuseppe Mongiovi

Campina Grande, Junho 1993

Universidade Federal da Paraíba
Centro de Ciências Tecnológicas
Coordenação de Pós-Graduação em Informática

João José Peixoto Furtado

A4 - Um ambiente de Apoio à Aquisição Automática de Conhecimento

Este trabalho foi apresentado à Pós-Graduação em Informática do Centro de Ciências e Tecnologia da Universidade Federal da Paraíba como requisito parcial para obtenção do grau de Mestre em Informática.

Orientador: Giuseppe Mongiovi

Campina Grande, Junho de 1993

F992a Furtado, Joao Jose Peixoto
A4 : um ambiente de apoio a aquisicao automatica de conhecimento / Joao Jose Peixoto Furtado. - Campina Grande, 1993.
104 f. : il.

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

1. Sistemas Operacionais Especificos 2. Metodos Indutivos de Aprendizado Automatico 3. A4 - 4. Dissertacao I. Mongiovi, Giuseppe, M.Sc. II. Universidade Federal da Paraiba - Campina Grande (PB)

CDU 004.451.9(043)

A4 - Um Ambiente de Apoio à Aquisição Automática de Conhecimento

João José Peixoto Furtado

Dissertação aprovada em 30.6.1993
(*Com Distinção*)

Giuseppe Mongiovì, *M.Sc*
Orientador

Fernando Antonio de Carvalho Gomes, *Dr.*
Componente da Banca

Bernardo Lula Júnior, *Dr.*
Componente da Banca

Edilson Ferneda, *Dr.*
Componente da Banca

Campina Grande, 30 de Junho de 1993

À **Beth**, companheira na busca de minha lenda pessoal e à **Lara** que é a alegria de todos os momentos dessa procura.

Agradecimentos

Aos meus pais, pelo exemplo de vida, carinho e apoio demonstrado nos momentos decisivos.

Ao professor e amigo Giuseppe Mongiovi, pela competente orientação, companheirismo e constante motivação dados ao projeto.

Aos meus amigos campinenses pela excepcional recepção que me deram, em especial Jorge, Nena, Lucas e Netinha.

Aos professores Fernando Gomes, Bernardo Lula e Edilson Ferneda, pela participação na banca examinadora, e pelos importantes comentários e sugestões feitos sobre o trabalho.

Aos amigos Tojal e Walfredo que, através de seus comentários lúcidos e suas idéias inteligentes, foram essenciais para a realização do trabalho.

Aos alunos da graduação Wladmir e Michele, pelo apoio na implementação dos programas.

Ao professor Francisco Nunes do Departamento de Matemática da UFPB pelo auxílio nas formulas estatísticas do processo de discretização.

Ao SEPROCE pela oportunidade que me foi dada.

Aos amigos do SEPROCE que torceram pelo meu sucesso, especialmente Luis Eduardo, Regina Estela, Ana Lúcia, Lúcia Pompeu, Humberto, Alberto que em nossos esporádicos almoços me encorajaram de continuar.

A Universidade de Fortaleza (UNIFOR) pelo apoio financeiro.

A IBM Brasil pelo apoio financeiro.

Sumário

Capítulo 1

Capítulo 2

Capítulo 3

Aquisição de Conhecimento a Partir de Exemplos

Capítulo 4

O Ambiente de Apoio a Aquisição Automática de Conhecimento – A4 33

Capítulo 5

Modelagem do Domínio para os Métodos Indutivos de AC 44

Apêndice B

Apêndice C

Lista de Figuras

RESUMO

Métodos indutivos de aprendizado automático a partir de exemplos são amplamente utilizados para aquisição de conhecimento (AC) em sistemas baseados em conhecimento. Estes métodos apresentam alguns problemas, em função de restrições estipuladas à modelagem do mundo real, da diversidade e peculiaridades dos algoritmos que os implementam e da forma como seus resultados são representados. Além disso os trabalhos realizados em aquisição de conhecimento indutiva têm enfocado quase que exclusivamente o processo de generalização em si. Entretanto, para que os métodos indutivos de aquisição de conhecimento apresentem resultados satisfatórios, é necessário que as entradas dos algoritmos generalizadores reflitam ao máximo o domínio do problema a ser resolvido.

Neste trabalho propomos o A4 - Ambiente de Apoio a Aquisição Automática de Conhecimento que tem por finalidade auxiliar todo o processo de aquisição de conhecimento indutiva, desde a modelagem do mundo real em exemplos e conhecimento preliminar *(background knowledge)*, até o tratamento das saídas geradas pelos algoritmos indutivos. Com o A4 buscamos prover um ambiente integrado fundamentado em técnicas de aprendizado automático, mas que envolve também algumas características dos métodos baseados em entrevistas (semi-automáticos). Objetivamos fazer uso das vantagens dos dois enfoques.

A arquitetura do A4 é orientada a objetos e fundamentada em três classes básicas: a classe *modela domínio*, a classe *algoritmo indutivo* e a classe *trata saídas*. A utilização do paradigma de orientação a objetos acrescentou funcionalidades ao ambiente como fácil reusabilidade de procedimentos.

Daremos um enfoque maior à classe *modela domínio*, que propõe meios de automatizar a modelagem do mundo real em formas aceitáveis pelos algoritmos indutivos.

O A4 está implementado em C++ em estações de trabalho SPARC/SUN.

ABSTRACT

Concept learning methods have been strongly used for knowledge acquisition in the knowledge based systems. However, these methods present some problems due to constraints in the real world modeling, the inductive algorithm diversity and the form as their results are presented. Besides, the domain modeling for knowledge acquisition inductive methods has not been properly approached. These methods have approached only the generalization process. However their success depend on the quality of the domain modeling.

In this work we propose the A4 (Knowledge Acquisition Environment Support) that helps the inductive knowledge acquisition process, modeling the domain into examples and background knowledge and it treats the algorithms results. A4 is based on the integration of interviews and machine learning methods in order to get the advantages of both.

A4 architecture is object oriented. The three basic class are: *domain modeling*, *inductive algorithm* and *output treatment*. The use of object oriented paradigm added new functionalities to the environment such as procedure reutilization.

Mainly, the *domain modeling* class is studied. This class helps the real word modeling into examples and background knowledge. This modeling increases the automatization degree of the learning process which produces faster and more powerful results.

The A4 has been implemented in C++ in the SPARC/SUN workstation.

CAPÍTULO 1

Introdução

1.1 Inteligência Artificial

A Inteligência Artificial (IA) é uma disciplina da ciência da computação que procura fazer programas de computador que simulem o comportamento humano.

Uma forma de compreender melhor o que vem a ser IA é definindo os problemas particulares que lhes são objeto de estudo (áreas e sub-áreas). A figura 1.1 mostra um modelo adaptado de Nilsson [Nilsson 80] que descreve os elementos básicos e principais áreas que formam a IA. Os elementos básicos do modelo são:

- Busca heurística ou Exploração de Alternativas
- Representação e manipulação de conhecimento
- Linguagens e ferramentas

Buscas heurísticas são necessárias porque as tarefas que são consideradas em IA quase sempre se apresentam complexas e mal definidas. Em função dessa complexidade, o espaço de busca das soluções, bem como o número de soluções viáveis, pode crescer exponencialmente. Prover boas formas de explorar as alternativas à resolução de um problema é fundamental em IA. O uso de heurísticas nessas resoluções é importante porque aumentam a

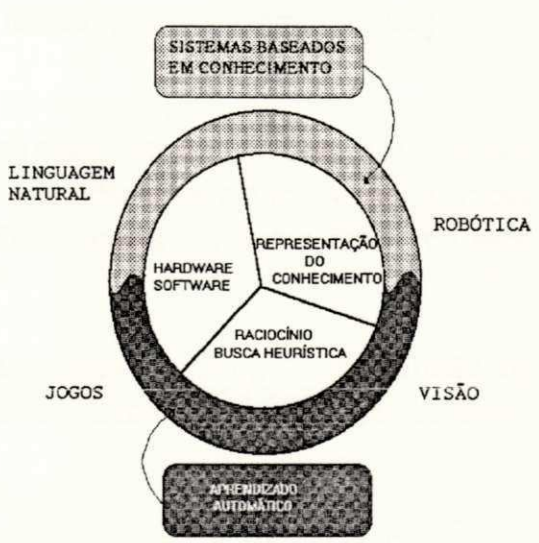


Figura 1.1 Pricipais áreas de IA

possibilidade de se encontrar uma solução aceitável.

Outro elemento básico da IA é o conceito de representação e manipulação de conhecimento. Uma das premissas básicas em IA é que inteligência requer conhecimento. Portanto, para o sucesso de suas aplicações, métodos eficientes de manipulação de conhecimento são fundamentais.

O último elemento básico do modelo refere-se aos aspectos de software básico, que suportam as aplicações de IA. Algumas linguagens e ferramentas, como LISP (*LISt Processing*), PROLOG (*PROgramming in LOGic*) e diversos *shells*, têm sido especialmente construídas para auxiliar o desenvolvimento de programas que visam resolver problemas típicos de IA.

Os elementos básicos do modelo citados anteriormente, suportam os seguintes sub-campos:

- Processamento de Linguagem Natural
- Robótica
- Visão por computador
- Sistemas Baseados em Conhecimento
- Aprendizado Automático (*Machine Learning*)

Processamento de linguagem natural é uma área relacionada com o desenvolvimento de interfaces para compreender a linguagem humana. Estas interfaces objetivam compreender tanto textos escritos como linguagem falada (compreensão dos sons).

A robótica relaciona-se com a construção de máquinas que simulem a atividade física e intelectual das pessoas. Os robôs podem ser entidades estacionárias ou móveis e devem exibir comportamento inteligente em situações em que o ambiente está em mutação.

A área de visão por computador é frequentemente usada em conjunção com a robótica e procura fornecer à máquina potencialidades para identificar objetos e/ou situações.

No modelo aqui descrito, consideramos as áreas de sistemas baseados em conhecimento e de aprendizado automático como uma camada especial. Essa consideração é feita, porque as tarefas de IA quase sempre envolvem atividades baseadas em conhecimento e consequentemente de aprendizado desses

conhecimentos. Ou seja, há uma alta aplicabilidade dessas duas áreas às outras da IA. Em função disso e por serem foco principal deste trabalho, descreveremos essas duas áreas com mais detalhe a seguir.

1.2 Sistemas Baseados em Conhecimento

O desempenho humano em suas principais atividades como compreender uma linguagem natural, planejar e tomar decisões ou mesmo realizar atividades físicas envolve o uso hábil de uma certa quantidade de conhecimento. A partir dessa premissa, surgiram os sistemas baseados em conhecimento (SBC).

Os SBCs são programas de computador que fundamentam-se na aquisição e manuseio de conhecimento. Esses sistemas têm como característica principal a existência de uma separação explícita entre o conhecimento que possuem (base de conhecimento) e as suas estratégias de controle (máquina de inferência).

Os Sistemas Especialistas (SE) são um tipo especial de sistemas baseados em conhecimento que visam resolver problemas complexos, em domínios específicos de conhecimento. Esses sistemas procuram simular o comportamento do especialista humano na resolução de problemas.

Em função do crescente sucesso dos sistemas especialistas e do consequente surgimento de aplicações que manuseiam grandes bases de conhecimento, procura-se cada vez mais alternativas para auxiliar a formação dessas bases. A tarefa de obter conhecimento de alguma fonte de conhecimento e transformá-lo em uma representação explícita, formando uma base de conhecimento, é chamada de **aquisição de conhecimento** (AC). Esta tarefa tradicionalmente é feita por um profissional de informática conhecido como engenheiro do conhecimento.

A fase de aquisição de conhecimento é considerada o *gargalo* do processo de desenvolvimento de um SE [Feigenbaum 81], pois em função de sua complexidade, consome uma grande quantidade de tempo. A experiência mostra que, no desenvolvimento de um SE se acrescenta por dia apenas de uma a quatro regras na BC [Quinlan 86].

1.3 Aprendizado Automático

Atualmente, a grande maioria dos programas de computador, inclusive os que utilizam técnicas de inteligência artificial, têm enormes limitações nas habilidades de aprendizado. Todo o conhecimento que eles possuem é proveniente de programações feitas por pessoas. Esses programas quando contêm erros, por si só não conseguem corrigi-los, repetindo-os todas as vezes que são executados. Além disso, eles não podem automaticamente gerar seus algoritmos, formular novas abstrações ou desenvolver novas soluções a partir de analogias ou descobertas.

Por outro lado, se considerarmos a inteligência humana, percebemos que as pessoas têm a capacidade de adquirir novos conhecimentos, aprender novos conceitos e melhorar com a prática e a partir dos próprios erros. São essas características que levam um jovem estudante inexperiente a tornar-se um médico, engenheiro, advogado, etc.

Devido a habilidade de aprender estar tão intimamente ligada a inteligência humana, alguns pesquisadores consideram como objetivo fundamental da inteligência artificial, prover a habilidade de aprendizado automático ou aprendizado pela máquina.

Máquinas que podem aprender são desejáveis porque a habilidade de aprendizado pode ser utilizada em muitas áreas e nas mais diversas aplicações, dentre as quais : visão por computador e reconhecimento de voz, entendimento de linguagem natural, sistemas tutoriais inteligentes e principalmente nos sistemas baseados em conhecimento.

Técnicas de aprendizado automático são usadas em sistemas baseados em conhecimento porque possibilitam a obtenção de conhecimento a partir de exemplos de decisões tomadas por um especialista.

1.4 Automatizando a Aquisição de Conhecimento

Como já foi dito, a tarefa de AC mostrou-se extremamente complexa. Além disso, em função da grande importância para a geração de bases de conhecimento eficazes, observa-se, hoje, um grande investimento na sua automatização.

Existem duas formas básicas de realizar aquisição automática de conhecimento: automatizando o processo de entrevista tradicional entre o engenheiro de conhecimento e o especialista ou aplicando técnicas de aprendizado automático.

Os sistemas de aprendizado baseados em entrevista (ou semi-automáticos) simulam o comportamento de entrevistadores humanos a fim de capturar informações sobre o domínio. Esse tipo de sistema está sendo vastamente utilizado e desenvolvido nas mais diversas formas. No entanto, sistemas baseados em entrevistas além de não diminuirem consideravelmente a presença do especialista por vezes exigem sessões de entrevista longas e cansativas.

Já os sistemas que se baseiam em aprendizado automático utilizam principalmente aprendizado a partir de exemplos. Esses métodos procuram adquirir conhecimento partindo de instâncias de casos reais (exemplos). A figura 1.2 ilustra a tarefa de AC a partir de exemplos. A idéia básica desse tipo de sistema é eliminar ao máximo a presença do especialista, dirigindo-se fundamentalmente pelos exemplos do domínio, daí serem chamados de métodos dirigidos a dados. Esta idéia ambiciosa também apresenta alguns problemas, principalmente pela falta de conhecimento semântico e heurístico, frequentemente utilizado pelos especialistas. Por esse motivo, tem se buscado alternativas de uso dessas informações semânticas, aqui denominado de conhecimento preliminar (*background knowledge*), para tornar mais eficaz o conhecimento gerado.

Figura 1.2 O processo de AC indutiva

Os métodos de aprendizado a partir de exemplos variam em função da utilização de conhecimento preliminar. Os métodos que se baseiam em deduções, também conhecidos como baseados em explanações (*explanation-based*), utilizam vasta quantidade desse conhecimento. Esses métodos normalmente são aplicados em situações nas quais o domínio em questão pode ser formalizado em uma teoria. Neles, deduções são utilizadas para operacionalizar partes do conhecimento sobre o domínio, através de explanações (provas) dos exemplos e de generalizações a outros exemplos.

Já os métodos indutivos baseiam-se, na sua maioria, em pouco ou nenhum conhecimento preliminar sobre o domínio. O enfoque indutivo baseia-se em induções

da descrição geral de um conceito, a partir de seus exemplos e contra-exemplos. Esses métodos estão sendo largamente utilizados, principalmente porque trabalham sem a necessidade de nenhum conhecimento preliminar e podem ser aplicados mais facilmente em situações reais. Exemplos da aplicação de sistemas de aquisição de conhecimento baseados em métodos indutivos são: o sistema de diagnóstico de falhas para circuitos impressos da ITT Europa, o sistema de separação gás-óleo da British Petroleum e o sistema de avaliação do desempenho do motor do ônibus espacial da NASA [Carter 87].

Mesmo com o relativo sucesso de alguns sistemas de aquisição de conhecimento indutivo, há problemas que dificultam a aplicação desses sistemas a uma maior variedade de domínios. Esses problemas podem ser analisados sob três enfoques:

1) Utilização de conhecimento preliminar : Os métodos indutivos podem apresentar resultados excessivamente complexos e incompreensíveis ao especialista. Isso ocorre em função de problemas como o problema semântico, decorrente de generalizações de conhecimento em função de conceitos irrelevantes [Mongiovi 90b]. A utilização de conhecimento preliminar pelos algoritmos indutivos, como feito pelos RPRISM [Cirne 91] e EG2 [Núñez 91], atenua esses problemas.

2) Restrições dos algoritmos : Outro problema dos métodos automáticos indutivos é que há uma série de restrições e limitações, embora distintas de algoritmo para algoritmo, que têm que ser cumpridas na modelagem do mundo real para que esses algoritmos possam ser aplicados. Essas restrições vão desde problemas de eficiência computacional a exigências conceituais.

3) Modelagem do mundo real : Os métodos automáticos indutivos consideram o conjunto de exemplos e o conhecimento preliminar sobre o domínio (quando usado), como já disponíveis [Quinlan 87a] [Cendrowska 88] [Cirne 92] [Núñez 91]. Na verdade, há uma lacuna entre como essas entradas se apresentam no mundo real e como elas se tornam operacionais aos algoritmos indutivos. Por exemplo, a identificação dos atributos mais representativos que formam os exemplos e a consequente construção do conjunto de treinamento (tabela de exemplos) tem se mostrado uma tarefa não trivial [Mongiovi 90a] [Hart 87]. Essa tarefa, se não for devidamente informatizada, diminui acentuadamente o caráter automático da aquisição de conhecimento.

A figura 1.3 ilustra a nossa visão do processo de AC indutivo a partir de exemplos, visto de uma forma mais completa. Na figura é ressaltada a existência de um processo de modelagem do domínio em exemplos e também em conhecimento preliminar.

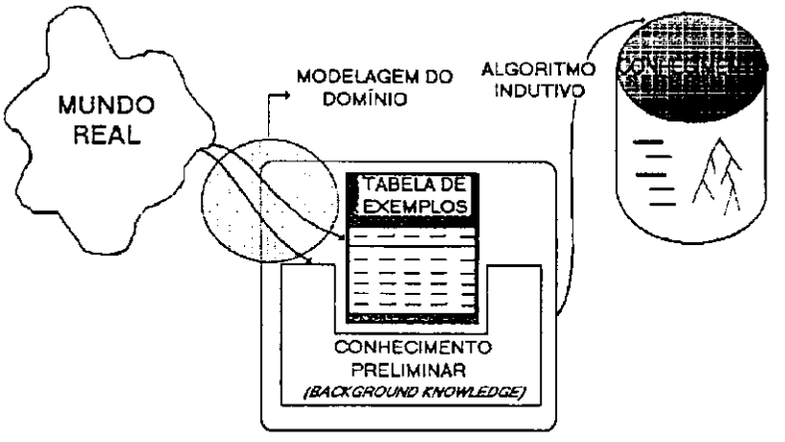


Figura 1.3 Uma visão completa de AC indutiva

Nos últimos anos surgiram vários algoritmos generalizadores, bem como algumas técnicas de simplificação dos resultados desses algoritmos. Cada um deles apresenta suas peculiaridades, com tratamentos próprios a cada um dos três principais problemas citados anteriormente. Em função dessa proliferação de algoritmos, quando se pensa em solucionar um problema utilizando um dado algoritmo indutivo, além de prover tratamento para os problemas básicos citados deve-se responder a questões como: O processo de indução embute alguma forma de simplificação das saídas ? O algoritmo generalizador utiliza algum tipo de conhecimento preliminar ? Se sim, qual ? Qual o tipo de saída produzida pelo algoritmo? Como comparar ou fundir resultados de dois ou mais diferentes algoritmos ? Como analisar os resultados de algoritmos que usam informações semânticas ?

Neste trabalho analisamos mais detidamente essas dificuldades, restrições e peculiaridades dos métodos de aprendizado automático indutivos e propomos o A4 (Ambiente de Apoio à Aquisição Automática de Conhecimento) [Vasco 92a]. O A4 objetiva facilitar o tratamento desses problemas e prover uma ferramenta única que possua condições de decidir que ferramenta utilizar em função da situação e responder às questões levantadas anteriormente. O A4 auxilia todo o processo de aquisição de conhecimento indutivo, desde a modelagem do mundo real em exemplos e conhecimento preliminar, até o tratamento das saídas geradas pelos algoritmos indutivos. A figura 1.4 mostra as camadas que compõem a arquitetura desse ambiente: a camada de *modelagem do domínio*, a de *indução* e a de *tratamento das saídas.*

Uma das características fundamentais da arquitetura do ambiente A4 é a integração de técnicas de aquisição automática de conhecimento com técnicas semi-automáticas

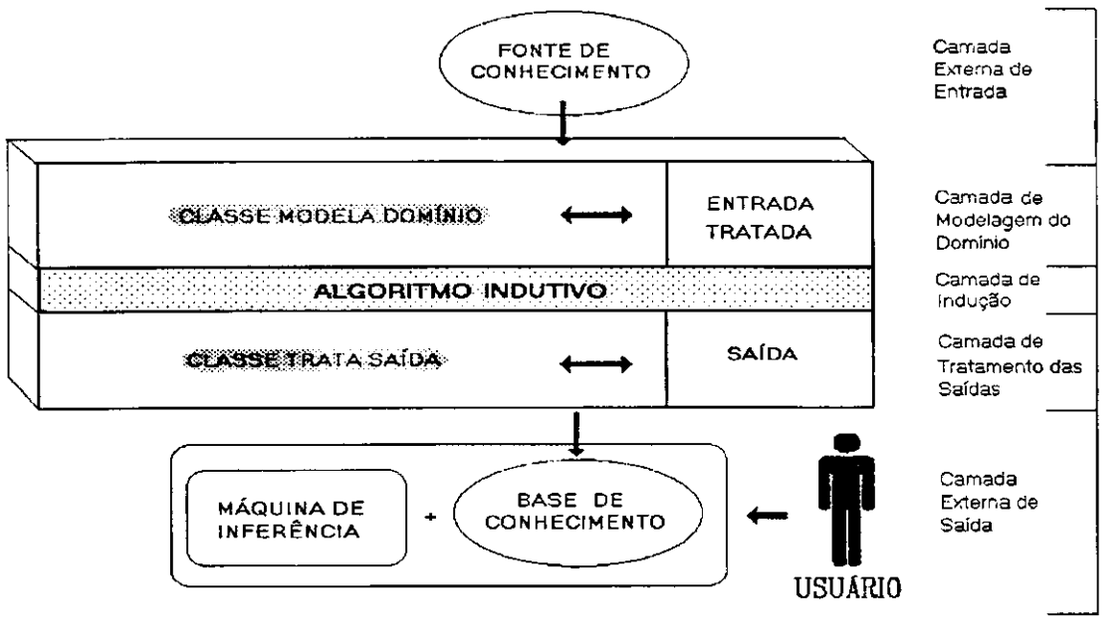


Figura 1.4 As camadas do A4

ou baseadas em entrevistas. A arquitetura é projetada no sentido de preservar as vantagens dos dois enfoques. Além dessas características, o A4 também fornece ferramentas para facilitar a escolha e a aplicação do algoritmo mais adequado a cada caso. Essas ferramentas auxiliam no cumprimento das restrições e limitações que determinados algoritmos podem exigir ou apresentar.

Neste trabalho, daremos um maior enfoque na modelagem do mundo real em representações tratáveis pelos algoritmos indutivos [Vasco 93a]. Essa modelagem resultará em exemplos devidamente estruturados e em conhecimento preliminar sobre o domínio, como relevância semântica [Cirne 91], custos e hierarquias de atributos [Núñez 91]. O processo dessa modelagem é realizado pela classe de objetos *modela domínio*, que está inserida no A4.

1.5 Estrutura da Dissertação

Este trabalho está dividido em 4 partes. A primeira parte é composta dos capítulos 2 e3 e se caracteriza por um enfoque conceitual da AC. No capítulo 2, fazemos uma análise da aquisição de conhecimento com um todo para, no capítulo 3, analisarmos mais especialmente aprendizado automático a partir de exemplos. Na segunda parte,

no capítulo 4, propomos o A4 e descrevemos seus componentes e no capítulo 5 enfocamos a tarefa de modelagem do domínio. Na terceira parte, nos capítulos 6 e 7, descreveremos os aspectos de desenvolvimento e implementação do A4. No capítulo 6 descrevemos a metodologia de desenvolvimento do Ambiente e no capítulo 7 descrevemos o funcionamento de sua interface. A última parte do trabalho são as conclusões e propostas para futuros trabalhos, as quais abordaremos no capítulo 8. Além das quatro partes citadas, esta dissertação possui três apêndices que complementam o trabalho. No apêndice A, descrevemos os métodos que serão utilizados por desenvolvedores de algoritmos para o A4. No apêndice B, descrevemos o procedimento necessário para inclusão de um algoritmo no ambiente. Por fim, no apêndice C, descrevemos o processo de discretização de atributos contínuos utilizado no A4.

CAPÍTULO 2

Aquisição de Conhecimento

O processo de aquisição de conhecimento consiste de se obter conhecimento de uma fonte de conhecimento e transformá-lo em uma representação explícita, formando uma base de conhecimento. Esta fonte de conhecimento pode ser um especialista, documentos sobre o domínio ou um banco de dados.

Faz-se necessária uma distinção entre os termos aquisição (*acquisition*) e eliciação (*elicitation*), pois algumas vezes estes termos são usados, ao nosso ver, erroneamente como sinônimos. Aquisição de conhecimento é um termo mais geral e inclui técnicas de eliciação de conhecimento, além de poder incluir técnicas de aprendizado automático. A eliciação de conhecimento refere-se somente ao processo de extração do conhecimento de especialistas. Alguns sistemas de AC implementam uma etapa de eliciação de conhecimento complementada por deduções e induções baseadas em técnicas de aprendizado automático [Gaines 88]. A figura 2.1 mostra a nossa visão de como esses conceitos se relacionam (adaptado de [Cirne 92]).

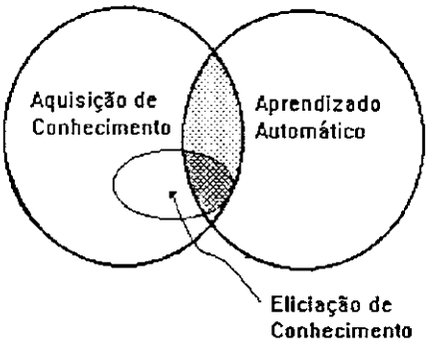


Figura 2.1 Relacionamento dos conceitos básicos de AC

2.1 Histórico dos Sistemas de AC

Os sistemas de aquisição de conhecimento tiveram seu início na década de 70 juntamente com o surgimento dos sistemas especialistas. O processo era totalmente manual e exclusivamente baseado em entrevistas com especialistas, sendo denominado de geração zero dos sistemas de aquisição de conhecimento.

Na verdade, a utilização de métodos de aprendizado automático em aquisição de conhecimento tem atraído a atenção de pesquisadores desde os anos 60 [McCarthy 60]. Mas só ao final dos anos 70 e início da década de 80 passou-se a fazer uso mais intenso do computador para aperfeiçoar o processo de aquisição de conhecimento. Nessa época surgiram outras técnicas de aquisição de conhecimento além das entrevistas tradicionais, caracterizando o que se chama a primeira geração de sistemas de aquisição de conhecimento.

Um dos primeiros sistemas de aprendizado voltados exclusivamente para aquisição de conhecimento de especialistas foi o META-DENDRAL [Buchanan 78], um programa que aprendia regras a partir de entrevistas, para identificar classes de compostos químicos a partir de espectogramas de massa. Após o sucesso do ID3 [Quinlan 83] em domínios determinísticos, como o jogo de xadrez, a tarefa de aquisição de conhecimento ganhou impulso.

Mesmo com estes avanços, o *gargalo* ainda permaneceu e essas técnicas se tornaram fracas, principalmente pela falta de integração com as outras fases de desenvolvimento de um SE.

Em meados dos anos 80 e o início da década de 90 surgiram os sistemas de suporte ao conhecimento. Esses sistemas representam a segunda geração de sistemas de aquisição de conhecimento. Eles se caracterizam por prover integração de várias técnicas e trabalhar com diversas fontes de conhecimento. Exemplos desses sistemas são AQUINAS [Boose 90] e KITTEN [Gaines 88].

Hoje em dia, uma das propostas que é mais estudada e considerada promissora é a de ambientes de AC. Nesses ambientes pode-se acoplar diferentes técnicas, gerenciadas por um sistema baseado em conhecimento [Vasco 93b]. Essa gerência, realizada em função das características do usuário e do problema, é responsável por indicar a técnica a ser adotada para atacar um problema.

2.2 Dificuldades da Aquisição de Conhecimento

A tarefa de aquisição de conhecimento para sistemas especialistas apresenta dificuldades que podem ser vistas em diferentes perspectivas.

Primeiro, devemos compreender que a atividade de adquirir conhecimento é uma tarefa de *modelagem* e não uma simples translação de conhecimentos dentro de

programas. As abstrações e distinções necessárias para construir os modelos podem não ser totalmente previstas ou mesmo serem imprecisas. Alie-se a isso, o fato de que sempre existem dificuldades em se implementar mecanicamente modelos, dificuldades essas que podem levar a problemas de representação gerados pela lacuna semântica entre o mundo real e o conhecimento em uma forma computável.

A segunda perspectiva nos mostra a dificuldade de aquisição de conhecimento em função das diferentes formas do conhecimento humano, principalmente no que se refere ao *conhecimento tácito ou compilado* que é um conhecimento baseado em experiências e difícil de ser enunciado.

Uma terceira perspectiva aponta para o problema da *indeterminação.* A indeterminação surge quando o especialista se expressa vagamente ou indeterminadamente sobre a natureza das associações entre os eventos. O problema da indeterminação reflete o fato de que normalmente os especialistas não conversam sobre associações de eventos em um caminho que precisamente seja o adequado na resolução de um dado problema. Embora o especialista possa ser encorajado ou guiado para ser o mais específico possível, um sistema de aquisição de conhecimento tem que ser hábil para tolerar ambiguidades e indeterminações.

Uma última perspectiva nos mostra o problema da *incompletude,* que pode ser examinado por dois ângulos diferentes. Em um deles, o especialista esquece de especificar alguma parte do conhecimento. Em outro ângulo, a incompletude ocorre porque o conhecimento do especialista não cobre completamente o asösunto em questão. Um sistema guiado para vencer a incompletude procura identificar conhecimentos que estão faltando e incrementalmente adicioná-los à base de conhecimento.

Em função dessas dificuldades que envolvem a tarefa de aquisição de conhecimento, podemos afirmar que: o processo de aquisição de conhecimento é *incremental e gradual.* As pessoas, e principalmente os especialistas, levam anos para formar seu conhecimento sobre um assunto. Seria por demais pretensioso desejar que em uma única sessão de aquisição de conhecimento um sistema de computador pudesse ser considerado apto a responder satisfatoriamente sobre o domínio do problema. A fase de refinamento se estende indefinidamente, por isso o sistema deve estar sempre pronto para *aprender* mais.

2.3 Uma Taxonomia para Aquisição de Conhecimento

Boose [Boose 88a] propõe uma divisão dos métodos de aquisição de conhecimento em dois : métodos cognitivos e métodos automatizados. Os métodos automatizados por sua vez dividem-se em : baseados em entrevistas (semi-automáticos) e baseados em aprendizado automático (automáticos).

2.3.1 Os métodos cognitivos

A aquisição de conhecimento por métodos cognitivos ou aquisição manual fundamenta-se na interação entre o engenheiro do conhecimento e o especialista. O engenheiro de conhecimento tem a responsabilidade de eliciar e analisar o conhecimento do especialista para gerar uma base de conhecimento. As técnicas normalmente utilizadas no enfoque cognitivo são entrevistas, análise de protocolos, observação direta e *brainstorming* [McGraw 89].

O processo manual de aquisição de conhecimento apresenta uma série de dificuldades. Essas dificuldades vão desde problemas de comunicação entre o especialista e o engenheiro de conhecimento à dificuldade de mapear conhecimento em uma forma computável em função de erros de transcrição e interpretação na formulação da base de conhecimento. Além disso, o engenheiro do conhecimento requer um perfil difícil de ser atingido. Um perfil que contemple conhecimento sobre técnicas de representação de conhecimento, técnicas de comunicação, técnicas psicológicas e um certo domínio do problema a ser resolvido. Apesar disso, técnicas cognitivas ainda são bastante utilizadas para o desenvolvimento de SBCs, principalmente em problemas que não sejam de classificação e mesmo porque a complexidade de alguns domínios, domínio jurídico, por exemplo, inviabilizam a aplicação de ferramentas automatizadas.

2.3.2 Aquisição de Conhecimento Semi-automática ou Baseada em Entrevistas

A idéia de automatizar a aquisição de conhecimento é atenuar as dificuldades dos métodos cognitivos e então conseguir maior agilidade, eficiência e consequentemente redução de custos no processo.

No enfoque baseado em entrevistas, a interação entre o engenheiro do conhecimento e o especialista é diminuída em função de uma ferramenta automatizada que realiza algumas das tarefas anteriormente exclusivas do engenheiro do

conhecimento. A figura 2.2 ilustra a inclusão de uma ferramenta automatizada no processo de AC e a diminuição da interferência do engenheiro do conhecimento neste processo.

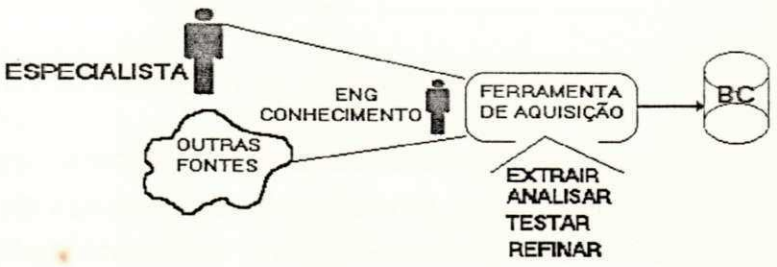


Figura 2.2 O processo semi-automático de AC

A seguir, analisamos as principais técnicas utilizadas nos sistemas de aquisição de conhecimento semi-automáticos. É importante salientar que embora essas técnicas possam ser utilizadas de forma manual, aqui, enfocaremos somente o aspecto automatizado das mesmas.

Entrevistas

Um dos mais importantes e utilizados métodos de aquisição de conhecimento é o das entrevistas [Gammack 88]. As entrevistas são feitas através de perguntas ao especialista sobre conceitos importantes do domínio do problema e consequentemente com expansões do seu raciocínio inicial. A forma de aplicação da entrevista é inerente à forma de como os conhecimentos que estão sendo adquiridos são representados e do método de resolução do problema que estabelece e controla a sequência de ações que realizam a tarefa.

As entrevistas são mais adequadas para aquisição de conhecimento substantivo(conhecimento básico que identifica relacionamentos simples), sendo normalmente utilizadas no ínicio do processo de aquisição. Existem trabalhos que procuram generalizar o processo de entrevista a partir da identificação de primitivas que direcionam o processo, fazendo com que a entrevista fique independente do domínio do problema. Um exemplo desses sistemas genéricos é SIS(*Shell Interview System*) [Kawaguchi 87], um *shell* para sistemas de entrevistas.

Um tipo especial de entrevista e que é voltada para identificar conhecimento estruturado hierarquicamente é o *laddering*. Questões sobre supertipos e instâncias de conceitos genéricos são realizadas com a finalidade de derivar uma estrutura taxonômica que relacione objetos quanto a uma superordenação ou subordinação. Perguntas do tipo *porque você fez isto ?* desenvolvem relações de superordenação e perguntas tipo *como você faz isto* ? levam a relações de subordinações. Movendo-se

em direção das superordenações atinge-se relações ou objetos mais genéricos, já as relações de subordinações levam a especificidade.

Teoria das Construções Pessoais (Redes de Repertório)

A Teoria das Construções Pessoais (*Personal Constructs Psicology - PCP*) não é propriamente uma técnica de aquisição de conhecimento, mas fundamenta a técnica de **Rede de Repertório** (*Repertory Grid*). A PCP foi desenvolvida por George Kelly [Kelly 55] e se baseia em distinções de dicotomias para conhecimento das coisas. O fundamento básico da teoria são as construções (modelos abstratos que o homem cria e os usa para compreender a realidade do mundo). Uma construção é definida como uma escala interna bipolar que discrimina as diferenças e similaridades do conjunto de elementos envolvidos. A teoria PCP foi desenvolvida dentro de um contexto clínico psicológico e, portanto, preparada para ultrapassar as defesas cognitivas das pessoas. Uma grande vantagem da utilização de técnicas baseadas em PCP é a fundamentação teórica que elas passam a possuir. Outra grande vantagem da PCP é que ela facilita a utilização de mecanismos para identificação de hierarquias. Isto facilita a tarefa de aquisição, porque possibilita ao especialista raciocinar em diferentes níveis de abstração.

O mapeamento dos elementos dentro de construções produz uma rede chamada rede de repertório. Essa rede tem duas polaridades, com elementos concretos ou abstratos em cada polo, que definem uma área ou característica de um evento. Cada elemento é definido na rede com seu equivalente contraste em outra polaridade, formando assim uma construção. A figura 2.3. mostra uma rede de repertório para

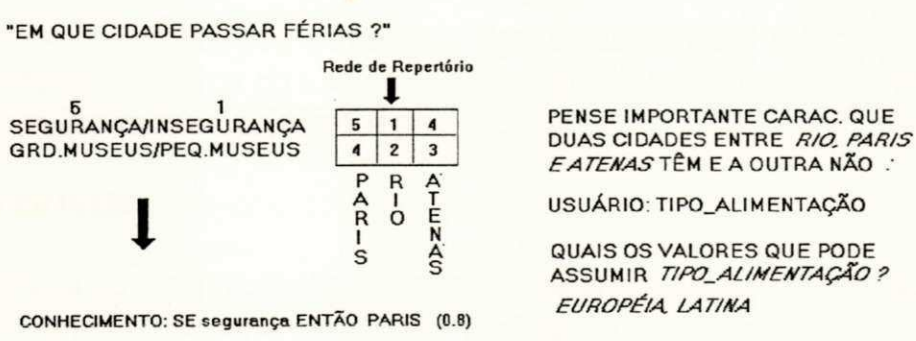


Figura 2.3. Exemplo de rede de repertório

escolher uma cidade onde passar férias. Redes de repertório foram inicialmente usadas: em psicologia [Shepard 82]; em educação [Pope 81]; e para estudar decisões

gerenciais feitas individualmente e em grupo [Shaw 80]. As redes de repertório são adequadas para obtenção de conhecimento substantivo.

Análise de protocolo

A análise de protocolo é a técnica que visa automatizar análises de *pensamentos em voz-alta* do protocolo de resolução do problema pelo especialista. Ou seja, o especialista resolve o problema ditando o seu processo mental. O resultado da análise de protocolo pode ser considerado como um caminho através de sucessivos estados do conhecimento representando a sequência de resolução do problema.

Uma possível estratégia de automatização da análise de protocolo é a usada em [Diederich 88]. Nela, a análise é feita em segmentos determinados pela pausa da voz do especialista e em alguns conceitos já obtidos sobre o domínio do problema. Nessa análise são pesquisadas diferentes tipos de relações entre os objetos do problema como: relações de ordem (X menor que Y), relações de especializações (X é um Y), relações de agregação (X é uma parte de Y).

Existem pelo menos três grandes dificuldades a serem vencidas na implantação de análise automática de protocolo, quais sejam:

- A dependência da qualidade da gravação e posteriormente da transcrição do protocolo, que impossibilita a eliminação de intermediários entre o especialista e o sistema.
- A distinção entre o conhecimento do especialista dos ruídos (informações não importantes) no protocolo, pois normalmente o conhecimento relevante vem acompanhado de informações irrelevantes.
- O especialista pode ultrapassar alguns passos de inferências, que se não afetam a eficácia do futuro sistema especialista pelo menos o fazem perder em capacidade de explanação.

Análise de textos

A engenharia do conhecimento recomenda que, ao iniciar o processo de aquisição de conhecimento, o engenheiro deve se municiar de documentos sobre o problema em questão, para ambientar-se com o mesmo. Isso normalmente leva muito tempo. Para contornar esse problema, as ferramentas de aquisição de conhecimento baseadas em entrevistas buscam um conhecimento básico do domínio através de análise de textos. As informações obtidas da análise de textos devem ser usadas como primeiro alimentador de um modelo do domínio do problema, servindo como base para o refinamento do modelo. É importante frisar que os objetos selecionados para a

análise de texto (livros, documentos, etc) devem ser atestados pelo especialista como fidedignos sobre o assunto em questão. As dificuldades encontradas nesse processo são as mesmas da análise de protocolo. Exemplos de sistemas que implementam análise de texto podem ser vistos em [Shaw 88a] [Shaw 88b] [Diederich 88].

2.3.3 Aquisição Automática de Conhecimento

As técnicas de aprendizado automático podem ser utilizadas para realizar aquisição de conhecimento para SBCs. Essas técnicas são úteis no refinamento de uma base de conhecimento utilizando os conceitos de indução e dedução para expansão ou derivação de conhecimento. Além disso, pode-se adquirir fatos e regras a partir de casos. Esta última funcionalidade é importante sobretudo em situações em que o especialista tem dificuldades de explicitar o seu raciocínio.

Uma série de taxonomias, inclusive com diferentes nomenclaturas, definem aprendizado automático.Nesta seção são descritos os principais paradigmas de aprendizado automático, os quais parecem obter um certo consenso. Depois é feita uma pequena análise de qual paradigma adotar em função do problema.

2.3.3.1 Aprendizado Automático: Paradigmas

Aprendizado automático pode ser dividido em quatro grandes paradigmas [Carbonell 89]: métodos conexionistas (modelos baseados em redes neurais artificiais), algoritmos genéticos (sistemas classificadores), aprendizado analítico (aprendizado baseado em explanações e certas formas de analogias) e aprendizado indutivo (adquirir conceitos a partir de induções sobre exemplos e contra-exemplos).

O Paradigma Conexionista

Sistemas de aprendizado automático baseados no paradigma conexionista, também chamados de redes neurais (*neural networks*), ultimamente têm recebido muita atenção. Eles conseguiram vencer as limitações dos Perceptrons[Rosenblatt 58] e das antigas redes lineares, incluindo camadas escondidas (*hidden layers*) para representar processamento intermediário e computar funções de reconhecimento não-lineares. Um sistema do tipo rede neural consiste de uma rede de elementos interconectados, tipo neurônios, que realizam algumas (normalmente simples) funções lógicas.

Sistemas conexionistas aprendem a discriminar entre classes de padrões de entrada de um domínio. A eles é apresentado um conjunto de instâncias de cada classe corretamente rotuladas (com alguma tolerância a ruído). O aprendizado consiste de modificações incrementais na força das conexões entre os elementos, normalmente

por mudanças nos pesos associados às conexões. Estas modificações seguem a filosofia de determinados algoritmos como *Boltzmann* [Hinton 84] ou *backpropagation*.

Exemplos de sistemas conexionistas de aprendizado, além do Perceptron, são o Pandemonium [Selfridge 59] e posteriormente [Tsypkin 72], [Caianiello 84] e [Hinton 84]. O trabalho de [Machado 89] é uma interessante experiência de modelos conexionistas em sistemas baseados em conhecimento. Nesse trabalho os neurônios representam conceitos e obtém-se uma base de conhecimento explícita. Essa filosofia é extremamente interessante, pois ataca o grande problema dos sistemas conexionistas para SBCs: a dificuldade de explanações.

O Paradigma Genético

Algoritmos genéticos (também chamados de sistemas classificadores) representam uma posição empírica extrema entre os paradigmas de aprendizado. Eles têm sido inspirados por uma analogia direta às mutações reprodutivas biológicas e à lei de Darwin da seleção natural (sobrevivência das espécies em um nicho ecológico). Diferenças de descrições de conceitos correspondem a indivíduos de uma espécie, e as modificações e combinações induzidas destes conceitos são testadas em comparação com uma função objetiva (o critério natural de seleção) para ver quais preservam a herança genética. Os algoritmos genéticos são eminentemente paralelos.

Desde o trabalho de [Holland 75], tem crescido muito o número de pesquisadores e de soluções dentro deste enfoque. Alguns problemas desse enfoque são compartilhados com outros paradigmas como os de redes neurais e de métodos indutivos. Por exemplo, a dificuldade de atribuir valores a funções que regulam a realização do algoritmo é estudada no enfoque indutivo desde a década de 60 [Samuel 63]. Uma alternativa de solucionar esse problema no enfoque genético é visto no algoritmo *bucket brigade* [Holland 86]. A questão de assinalamento de créditos e culpas (punição e recompensa em [Machado 89]) está intimamente ligada aos modelos conexionistas como na técnica de *backpropagation*.

O Paradigma Analítico

O paradigma analítico, chamado por alguns de paradigma dedutivo, é baseado em aprendizado a partir de poucos exemplos (as vezes somente um) mas com uma rica e fundamentada teoria sobre o domínio em questão. Utilizam experiência de problemas passados para guiar quais cadeias dedutivas percorrer quando da resolução de novos problemas, ou para formular regras de controle que tornem mais eficientes as pesquisas no domínio. Nesse paradigma, um exemplo corresponde a uma parte da cadeia dedutiva correspondente à resolução do problema. Uma característica desse

enfoque é que eles podem melhorar a eficiência sem sacrificar a acurácia ou generalidade. Exemplos de métodos baseados no paradigma analíticos estão em [Mitchell 86] [Carbonnell 86] [Laird 86].

O Paradigma Indutivo

Dos quatro paradigmas citados, o mais amplamente estudado e aplicado é o indutivo. Nele, procura-se induzir uma descrição geral de um conceito a partir de uma sequência de instâncias deste conceito e (normalmente) conhecidos contra-exemplos do conceito. A tarefa é construir uma descrição geral do conceito no qual todas as instâncias prévias podem ser rederivadas por instanciação mas nenhuma das instâncias prévias negativas (os contra-exemplos) podem ser rederivadas pelo mesmo processo. Nesse nível de abstração o problema pode soar simples, mas existe uma série de dificuldades a serem vencidas. A maior dificuldade a considerar é que o espaço de busca de sistemas indutivos pode crescer exponencialmente, principalmente em domínios não determinísticos, devido a existência de ruídos. Os ruídos existem em função de exemplos que possuem desvios do padrão, seja por estarem classificados erroneamente ou seja pela raridade de aparecimento.

2.3.3.2 Que Paradigma Utilizar ?

Podemos fazer algumas observações de caráter comparativo entre os paradigmas descritos anteriormente. Essas observações nos direcionam quanto a aplicabilidade de cada método.

- O enfoque conexionista é superior aos outros quando se destina a uma tarefa de reconhecimento de padrões contínuos em domínios não estruturados. Reconhecimento de fonemas da voz é um exemplo.
- O enfoque indutivo e genético são adequados a reconhecimento de padrões discretos. O enfoque indutivo é particularmente indicado para se adquirir novas descrições de conceitos e regras para sistemas especialistas. A característica eminentemente simbolista desses enfoques é traduzida em conhecimentos inteligíveis que facilitam explanações, validações e manutenções. Essas características são de extrema importância para sistemas baseados em conhecimento.
- Se há a possibilidade de se elaborar uma teoria sobre um domínio, o enfoque analítico é também adequado para aquisição de conhecimento em sistemas baseados em conhecimento. Nessa situação, esse enfoque pode prover alta eficiência a qualquer sistema baseado em regras, pois uma grande quantidade de conhecimento pode ser utilizada em reformulações de parâmetros de controle que dirigem a busca pela solução.

- Existem tarefas que podem ser enfocadas por mais de um método. Em alguns domínios, tarefas complexas de aprendizado podem coexistir e por isso múltiplos métodos podem ser aplicados. Por exemplo, o enfoque conexionista sensoriando a interface, métodos indutivos formulando regras empíricas de comportamento, e o enfoque analítico sendo utilizado para melhorar o desempenho quando o domínio já está bem compreendido.

2.4 Conclusão

Neste capítulo traçamos um breve histórico de aquisição automática de conhecimento, além de definirmos alguns conceitos básicos da área.

Embora haja uma enorme proliferação de taxonomias definindo aquisição de conhecimento automatizada e de como ela pode ser dividida, a classificação das técnicas automatizadas em baseadas em entrevistas e baseadas em aprendizado automático é a mais adotada.

Os métodos de aprendizado automático foram analisados a partir de quatro grandes paradigmas: conexionista, indutivo, analítico e genético. Sendo os paradigmas indutivo e analítico os mais utilizados no processo de aquisição de conhecimento para sistemas especialistas.

A utilização de diferentes técnicas de uma forma integrada tem se mostrado uma promissora alternativa para solucionar os problemas de AC. Em função disso, estão surgindo ambientes de aquisição de conhecimento que utilizam tanto técnicas de aprendizado automático, nos seus diferentes paradigmas, como técnicas de AC baseadas em entrevistas.

Capítulo 3

Aquisição de Conhecimento a Partir de Exemplos

Neste capítulo, analisaremos mais detidamente a aquisição automática de conceitos a partir de exemplos para formação de bases de conhecimento, que de agora em diante denominamos de aprendizado de conceitos. Os métodos a serem analisados se enquadram basicamente nos paradigmas indutivo e analítico que, como já comentamos anteriormente, são mais apropriados para aquisição de conhecimento para sistemas especialistas.

Nossa idéia central neste capítulo é definir um quadro conceitual a partir de três características básicas, consideradas por [MacDonald 89] como pilares mestres para os sistemas de AC: a forma de representar o exemplos e os conceitos, a maneira de tendenciar *(biasing)* a pesquisa no espaço de solução do problema e o modo como um sistema de AC interage com o *instrutor*. A figura 3.1 mostra esquematicamente

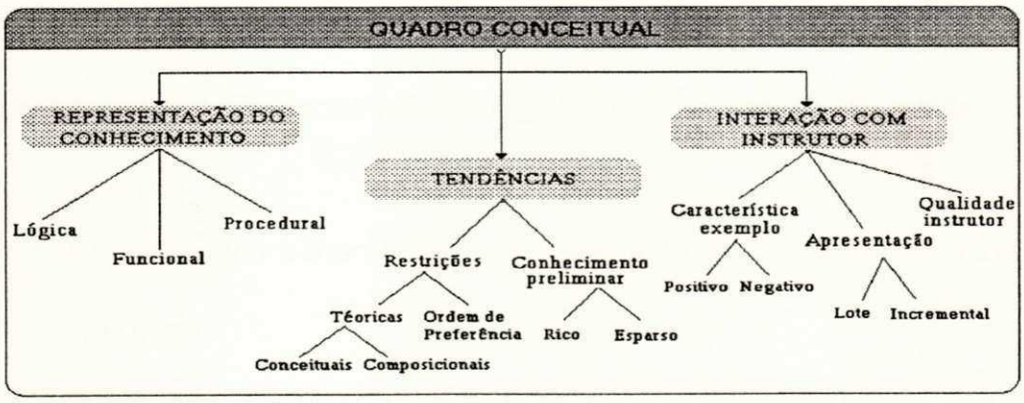


Figura 3.1. Características de aprendizado de conceitos

essas três características. A definição deste quadro conceitual tem os seguintes objetivos:

- classificar alguns sistemas existentes a partir dessas características.

- auxiliar a identificação de que características um sistema de aprendizado deve possuir para um certo problema.
- auxiliar na escolha de que plataforma (em função dessas três características) deverá ter o ambiente de aquisição de conhecimento que será proposto como alternativa de solução para o problema de AC.

Mas, antes da descrição dessas características, decidimos definir dois conceitos fundamentais que norteiam os principais tipos de sistemas de aquisição automática de conhecimento: indução e dedução.

3.1 Indução x Dedução

Os métodos **indutivos** fazem inferências de conclusões gerais a partir de exemplos. Como o raciocínio não é dedutivo a validade das conclusões obtidas por indução não são totalmente garantidas. Contudo, o raciocínio indutivo é de extrema importância em qualquer processo de descoberta e aprendizado.

Uma definição formal do que vem a ser indução foi apresentada por Genesereth e Nilsson [Genesereth 87]. Nela, as premissas são divididas em uma teoria base ou conhecimento preliminar CP e em um conjunto de treinamento (os exemplos) E. Desta forma podemos dizer que um conceito C é dito induzido se:

i) Os exemplos não são consequência lógica do conhecimento preliminar

$$CP \not\Rightarrow E$$

ii) O conceito não é inconsistente com o conhecimento preliminar e com os exemplos

$$CP \cup E \not\Rightarrow \neg C$$

iii) Os exemplos são consequência lógica do conhecimento preliminar e do conceito juntos.

$$CP \cup C \Rightarrow E$$

Podemos ver um exemplo de um processo, utilizando o domínio do jogo de cartas. Dado que :

J,Q e K são símbolos para definir as cartas com face; S,C,P,O representam os naipes; Num, Face, Vermelho e Negro são relações unárias.

Temos o seguinte conhecimento preliminar:

$\forall n \forall z \ (n = < 10 <===> Num([n,z]))$

$\forall n \forall z \ (n > 10 <===> Face([n,z]))$

$\forall n \forall z \ ((z = S \ V \ z = P) <===> Negro([n,z]))$

$\forall n \forall z \ ((z = C \ V \ z = O) <===> Vermelho([n,z]))$

Os exemplos representam retiradas aleatórias feitas do conjunto de cartas do baralho. A cada retirada, em função de uma certa lei de formação (o conceito a ser descoberto), diz-se que a carta é ou não premiada. A partir então do seguintes exemplos

Premiada([4,P])

Premiada([7,P])

Premiada([2,S])

$\neg$ Premiada([5,O])

$\neg$ Premiada([J,S])

é razoável propormos por indução o seguinte conceito

$\forall x \ (Num(x) \wedge Negro(x) <===> Premiada(x)) \qquad (\alpha)$

Esta conclusão é consistente com o conhecimento preliminar além de permitir inferências sobre outras possíveis cartas a serem retiradas.

Já nos métodos de aprendizado **dedutivo** os exemplos e conceitos são deduzíveis do conhecimento preliminar. Os exemplos servem para mostrar quais deduções são importantes. Na maioria dos problemas de aprendizado dedutivo há uma enorme quantidade de conceitos que podem ser deduzidos do conhecimento preliminar e o que se pretende é selecionar os apropriados aos exemplos. A formalização pode ser a seguinte :

i) Os exemplos são consequência lógica do conhecimento preliminar

$CP \Rightarrow E$

ii) O conceito não é uma parte explícita do conhecimento preliminar

$C \not\in CP$

iii) O conceito é consequência lógica do conhecimento preliminar

$CP \Rightarrow C$

iv) Os exemplos são consequência lógica dos conceitos

$$C \Rightarrow E$$

Permanecendo no domínio do jogo de cartas, podemos exemplificar o processo de formação de conceito a partir de dedução da seguinte forma:

Supondo a existência do conhecimento preliminar definido anteriormente, adicionamos a seguinte premissa:

$$\forall n \forall x ((n = S) <==> Premiada([x,n]))$$

então, a partir dos exemplos

Premiada([J,P])

Premiada([Q,P])

Premiada([K,P])

podemos concluir por dedução que

$$\forall x (Negro(x) \wedge Face(x) <==> Premiada(x)) \qquad (\beta)$$

A exemplificação feita acima serve também para observarmos a diferença fundamental entre as duas filosofias. O conceito β encontrado por dedução é garantido como verdadeiro, enquanto que o conceito α adquirido por indução não o é.

3.2 Definindo o Quadro Conceitual de Aprendizado de Conceitos

3.2.1 Representação de Conceitos e Exemplos

Conceitos e exemplos são, respectivamente, a saída e a entrada dos sistemas de aprendizado de conceitos. A forma de representá-los é uma importante característica que os engenheiros do conhecimento devem dominar, a fim de selecionar a mais adequada para a aplicação num problema, pois influenciam no poder representacional do sistema. Analisaremos conceitos e exemplos sob os três grandes enfoques de representação de conhecimento em inteligência artificial: enfoque baseado em lógica, enfoque funcional e enfoque procedimental. Estas representações são diferentemente apropriadas em função do tipo de problema. Por exemplo, árvores de decisão são naturalmente representadas por expressões lógicas, polinômios são mais

adequadamente representados por funções, já tarefas de robôs são bem representadas por procedimentos.

3.2.1.1 Lógica

A forma de representar conhecimento baseada em lógica é a mais comumente utilizada. Principalmente em aprendizado dedutivo, porque a lógica já embute mecanismos próprios de realizar deduções.

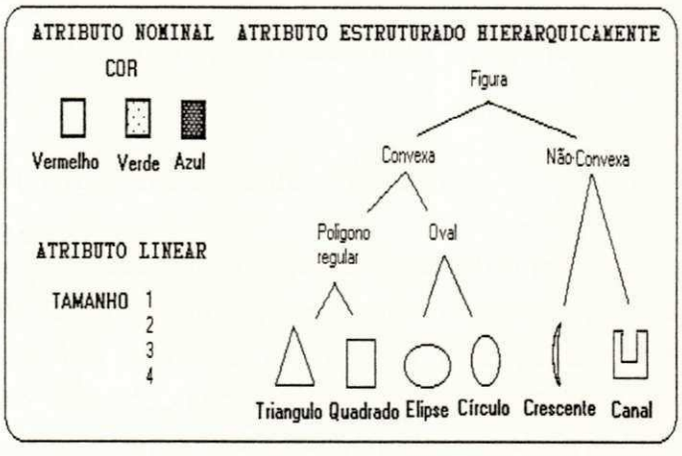


Figura 3.2 Tipos de atributos

Dois tipos de representações lógicas são as mais usadas: cálculo proposicional e cálculo dos predicados de primeira ordem. O cálculo proposicional utiliza predicados com constantes e operadores para disjunção, conjunção, implicação e negação [Nilsson 80]. A representação mais natural para os exemplos, e que se baseia na lógica proposicional, é a lista de pares atributo-valor. Os tipos destes atributos podem ser nominal, linear e estruturado em árvore. A figura 3.2 exemplifica esses atributos com seus respectivos valores.

3.2.1.2 Funcional

Representação funcional é apropriada para complexas expressões numéricas e não-numéricas. Relacionamentos entre uma certa quantidade sensoriada por um robô e os parâmetros de uma consequente movimentação são uma possível aplicação deste enfoque. A forma mais utilizada de representação funcional para conceitos é o cálculo lambda (em que se baseia a linguagem Lisp) [Glasser 84].

3.2.1.3 Procedimental

Os conceitos que são representados por procedimentos são tipicamente procedimentos realizados por robôs como: levantar pesos, montar, deslocar-se, etc. A representação procedimental é adequada para representar execução sequencial de

procedimentos, quando efeitos colaterais, como modificações no mundo real após um movimento do robô, são vitais para a perfeita execução do procedimento.

3.2.2 Tendências (*BIAS*) de Pesquisa

Pesquisa é fundamental em todo sistema de aquisição de conhecimento automático. Em grandes espaços de busca, as pesquisas quase sempre se tornam intratáveis, se não forem fortemente tendenciadas (*biasing*). Isso é verdade tanto para métodos indutivos como dedutivos. Dividiremos as formas de tendenciar as pesquisas em função de duas características: a quantidade de conhecimento preliminar existente e as restrições aplicadas à indução.

3.2.2.1 Tendências em função do conhecimento preliminar

A quantidade de conhecimento já possuída de antemão sobre um problema afeta fundamentalmente o aprendizado. Podemos afirmar que quanto mais conhecemos um dado assunto, fica mais fácil fazermos induções, inferências, deduções, analogias, enfim aprendermos mais sobre o assunto. Técnicas de aprendizado de conceitos podem ser divididas nas que necessitam de pouco conhecimento e nas que necessitam de muito conhecimento.

- *Técnicas baseadas em pouco conhecimento preliminar*: Estes métodos são usados quando pouco ou nenhum conhecimento está explicitamente disponível durante o aprendizado. Os sistemas que utilizam esse tipo de tendência apresentam a vantagem da simplicidade de implementação e de aplicação em diferentes domínios, pois se baseiam somente nos exemplos ou em parâmetros numéricos reajustáveis. Mas apresentam a desvantagem da falta de semântica que pode implicar em resultados não eficazes.
- *Técnicas baseadas em muito conhecimento preliminar* : Estas técnicas aplicam considerável quantidade de conhecimento sobre o domínio. O grande problema desse enfoque é que nem sempre se consegue ter um domínio que já possua um definição concisa, que possa ser utilizada no processo de aprendizado. Ou seja, existe o problema de que para se obter o conhecimento preliminar é necessário realizar um processo de aquisição de conhecimento.

3.2.2.2 Tendência por restrições na indução

As restrições na indução são formas de diminuir o espaço de busca que um sistema de AC deve percorrer para formar um conceito. Examinaremos aqui três formas pelas quais a indução pode ser restringida: por tendências conceituais (*conceptual bias*), tendências composicionais (*compositional bias*) e ordem de preferência.

- *Tendências conceituais* : Limitam o vocabulário para expressar os conceitos e os exemplos. Numa representação baseada em lógica do jogo de baralho, por exemplo, os valores ao invés de serem elementos do conjunto {2,3,4,5,6,7,8,9,10,J,Q,K,A}, seriam somente as descrições {carta_com_face, carta_sem_face}. Os métodos indutivos que geram árvores de decisão, por exemplo, possuem tendências conceituais, nas quais os valores dos atributos são limitados a valores nominais, não aceitando-se valores contínuos.
- *Tendências composicionais* : restringem como o vocabulário pode ser utilizado para compor conceitos. Nesse caso o que é restringido é a linguagem, sendo impossível a formação de alguns tipos de conceitos. Por exemplo, no domínio do baralho, disjunções poderiam ser proibidas e somente conjunções como espadas $\wedge$ carta_com_face seriam válidas.
- *Ordem de preferência* : Sistemas de aquisição de conhecimento devem prover formas de escolher entre conceitos que estão competindo. Uma maneira simples e altamente utilizada é escolher o conceito mais simples (menor em condições). O tamanho e a simplicidade do conceito pode ser medido em função da quantidade de: disjunções em expressões lógicas, chamadas a funções em expressões funcionais, *loops*, condições e declarações em procedimentos.

3.2.3 Forma de Interação com o instrutor

Nos sistemas de aprendizado de conceitos existe a figura do instrutor ou professor. Ele é o responsável por apresentar os exemplos ao sistema. Como o instrutor pode ser um especialista, engenheiro do conhecimento, um leigo ou até dados vindos de uma base de dados, busca-se trabalhar com o tipo de exemplo que o instrutor acha mais normal e natural, ordenado em uma forma também natural para ele. Três considerações são importantes de serem feitas sobre a forma de interação do instrutor com o sistema: como a apresentação dos exemplos é realizada, qual é a qualidade dos exemplos apresentados e qual é a qualidade do instrutor.

3.2.3.1 Forma de apresentação dos exemplos

Os exemplos podem ser apresentados em lote ou incrementalmente. Com a apresentação em lote todos os exemplos são apresentados de uma só vez [Dietterich 83]. A partir desses exemplos é obtida uma descrição da classe, que não fica aberta a modificações. Já um outro enfoque mais recente baseia-se em aprendizado incremental [Van de Velde 89]. Aprendizado incremental possibilita a apresentação de exemplos de uma forma parcelada. A representação de conhecimento é transformada a cada exemplo obtido. Essa técnica está sendo fortemente estudada porque reflete mais perfeitamente as situações do mundo real, pois aprendizado é um processo

gradual. Além disso, sistemas incrementais supõe-se serem mais rápidos, pois a contribuição de cada exemplo é obtida mais cedo [MacDonald 89].

3.2.3.2 Tipos dos exemplos fornecidos

Exemplos são a principal (muitas vezes a única) informação que um sistema tem para ajudar a identificar um conceito particular. Por isso a escolha dos exemplos é essencial.

Duas questões são fundamentais: se os exemplos contêm ou não ruídos e se os exemplos são positivos ou negativos. O comportamento de métodos de AC diferem bastante em função da forma como se postam ante essas características. Os primeiros sistemas de aprendizado indutivo consideravam que toda instância estava corretamente classificada com respeito a um dado conceito. Em problemas reais tal suposição é muito difícil de ser satisfeita, principalmente em domínios não-determinísticos. Novos sistemas passaram então a aceitar informações inconsistentes, instâncias não rotuladas e até informações incompletas.

Quanto ao tipo de exemplo aceito pelo algoritmo de aprendizado, é importante observar que se são aceitos tanto exemplos positivos como negativos o aprendizado tende a ser mais rápido.

3.2.3.3 Qualidade do instrutor

Um instrutor de boa qualidade pode apresentar exemplos *iluminados* que podem simplificar a tarefa do sistema. O instrutor poderia : apresentar exemplos com a classificação sempre correta, indicar as ausências explicitamente, determinar uma ordem de apresentação interessante, introduzir só o essencial a cada *lição* e sobretudo apresentar exemplos representativos que tornariam o processo de aprendizado mais eficaz. Em função da dificuldade de se poder contar com *bons instrutores*, os sistemas de AC procuram uma certa independência da qualidade do instrutor.

3.3 Utilizando o quadro conceitual

Nesta seção, analisamos alguns sistemas de aprendizado mais tradicionais e os classificamos dentro das características que compõem o quadro conceitual definido anteriormente. Esses sistemas foram divididos em quatro enfoques diferentes: enfoque baseado em similaridades, enfoque baseado em hierarquias, enfoque baseado em

explanações e enfoque baseado em descoberta. Essa divisão serve para enquadrar mais uma nomenclatura, também extremamente utilizada em aprendizado automático, ao quadro conceitual que definimos.

3.3.1 Enfoque Baseado em Similaridade

No enfoque baseado em detecção de similaridades (*similarity-based*) encontramos as maiores aplicações de AC automática. A representação de conhecimento baseada em lógica é a mais apropriada e utilizada neste enfoque. Os conceitos são representados na forma de regras de produção e os exemplos através de vetores de atributos. As regras de produção podem vir de árvores de decisão(AD), grafos ou serem geradas diretamente pelo algoritmo generalizador.

Procedimentos que geram árvores de decisão foram popularizados com o algoritmo ID3[Quilan 1986]. Árvores de decisão são compostas por folhas (elementos a serem classificados), nós não terminais (atributos dos fatos) e por ramos que ligam os nós (valores dos atributos associados). O objetivo dos algoritmos que constroem árvores de decisão é gerar árvores mínimas, sem sacrificar sua qualidade. Ou seja, árvores que possam classificar todos os exemplos dados.

O ID3 gera a árvore de uma maneira *top-down*, sendo por isso classificado como TDIDT (*Top Down Induction of Decision Trees*). Ele utiliza uma função de avaliação baseada na quantidade de informação que cada atributo possui. Os atributos de melhor função de avaliação são colocados nos primeiros níveis da árvore. Por causa de sua simplicidade e potencialidade, o ID3 tem sido estudado exaustivamente e diversas melhorias foram feitas no algoritmo a fim de vencer algumas limitações encontradas quando da resolução de problemas reais.

Outra forma de se conseguir regras de decisão por métodos indutivos é obtê-las diretamente dos exemplos, sem ter que passar por árvores de decisão [Cendrowska 88][Cirne 92]. A vantagem deste enfoque é que as regras produzidas são modulares, mais simples e mais genéricas. A figura 3.3 mostra, para o domínio de oftalmologia, regras de decisão advindas de uma AD e regras advindas diretamente do algoritmo generalizador, no caso o PRISM [Cendrowska 88]. Podemos percerber que, embora sejam em mesmo número, as regras geradas pelo PRISM são menores e modulares pois um mesmo atributo não aparece nas premissas de todas as regras.

A apresentação dos exemplos nos casos citados acima é feita em lote, mas outros enfoques aceitam apresentação incremental dos exemplos (*version space* [Mitchell 82]) e até mesmo em métodos que geram árvores de decisão [Van de Velde 89].

REGRAS ID3	REGRAS PRISM
1. D1 => E3 2. D2 & C1 & B1 & A1 => E2 3. D2 & C1 & B1 & A2 => E2 4. D2 & C1 & B1 & A3 => E3 5. D2 & C1 & B1 => E2 6. D2 & C2 & B1 => E1 7. D2 & C2 & B2 & A2 => 8. D2 & C2 & B2 & A2 => E3 9. D2 & C2 & B2 & A3 => E3	1. D1 => E3 2. C1 & D2 & A2 => E2 3. C1 & D2 & A1 => E2 4. C1 & B1 & A3 => E3 5. C1 & D2 & B2 => E2 6. D2 & C2 & B1 => E1 7. A1 & C2 & D2 => E1 8. C2 & B2 & A2 => E3 9. C2 & B2 & A3 => E3

Figura 3.3 Regras geradas pelo PRISM e pelo ID3

Normalmente, os métodos do enfoque baseado em similaridade não utilizam conhecimento sobre o domínio para restringir a busca, mas algumas alternativas de se utilizar conhecimento preliminar podem ser vistas nos algoritmos RPRISM [Cirne 91], APR-X[Donato 93] e EG2 [Núñez 91]. Esses métodos utilizam, sobretudo, tendências conceituais.

Alguns sistemas indutivos utilizam representação funcional. Um exemplo é Bacon [Langley 83], que tenta descobrir leis científicas de funções induzidas a partir de dados observados. Outro exemplo é Noddy [Andreae 84], um sistema de aprendizado para robôs. Noddy utiliza forte tendência composicional para restringir o espaço de busca. Noddy utiliza também representação procedimental. Os exemplos são passos de um procedimento desejado. Noddy adquire procedimentos de robôs e completa-os com informações de controle que não estão explicitamente presentes nos exemplos. Ele aplica conhecimento sobre leis e operadores matemáticos que podem ser combinados para obter generalizações.

3.3.2 Enfoque de Aprendizado Hierárquico

No enfoque hierárquico, baseada em árvores conceituais, tanto representações lógicas como funcionais são utilizadas. Há a necessidade de se representar muitos relacionamentos entre os atributos e até entre as classes. Representação tipo cálculo de predicado é utilizada em Marvin [Sammut 86]. Esse sistema aprende conceitos de estruturas hierárquicas. Dado um exemplo, ele procura uma forma de expressá-lo em termos de conceitos conhecidos. O sistema tenta deduzir conceitos e interativamente pergunta ao instrutor se o conceito é uma generalização do exemplo. Marvin também utiliza representação funcional para representar relacionamentos de atributos como funções. Neste tipo de aprendizado o conhecimento preliminar cresce em função de

cada conceito aprendido, com novas construções sendo construídas em cima de antigas.

Representação de conhecimento baseada em redes semânticas também são utilizadas no enfoque hierárquico. Principalmente quando os conceitos não são estruturados em uma forma de árvore.

3.3.3 Enfoque Baseado em Explanações

No enfoque baseado em explanações (*explanation-based*) se parte de muito conhecimento sobre o domínio. Esse conhecimento é o que constitui a teoria sobre o domínio e que será usada para deduzir conceitos. Aqui encontramos os métodos baseados no paradigma analítico. Técnicas dedutivas são usadas para operacionalizar partes do conhecimento sobre o domínio. O método funciona a partir de explanações (provas) de um exemplo dado, que são generalizadas a outros exemplos.

Um dos primeiros exemplos de sistemas baseados em explanações é Lex-II, o qual aprende heurísticas de resolução de problemas de integrais[DeJong 86]. Um outro exemplo do enfoque baseado em explanações é visto em [Mitchell 86] para o domínio do mundo dos blocos.

O enfoque baseado em explanações utiliza principalmente lógica. Desta forma deduções são feitas mais naturalmente. Um Exemplo de um sistema deste enfoque que utiliza representação funcional é Coper [Kokar 86].

3.3.4 Enfoque baseado em Descobertas

Os sistemas que se enquadram neste enfoque operam quase que autonomamente, realizando experiências para melhorar sua base de conhecimento. Em vez dos exemplos serem apresentado por um instrutor, sistemas como AM[Lenat 78] e EURISKO [Lenat 83] buscam gerar conceitos a partir de exemplos descobertos pelo próprio sistema. É uma tentativa extremamente ambiciosa e não há muitos resultados práticos nesse enfoque. A representação utilizada é a lógica.

Esses sistemas podem proceder tanto indutivamente como dedutivamente. Eles formam conjecturas após examinar poucos exemplos de conceitos (as vezes só um). A partir de então tentam provar essas conjecturas (fazem indução e depois dedução).

3.4 Conclusão

Neste capítulo, definimos um quadro conceitual para aprendizado de conceitos, a partir de três características básicas: a forma de representação dos exemplos e conceitos, as tendências realizadas no processo de aprendizado e a forma de interação do instrutor com o sistema.

Essas três características básicas serão úteis quando da definição do ambiente de aquisição de conhecimento a ser descrito no capítulo seguinte. Além disso, com esse quadro conceitual, nos foi também possível analisar alguns importantes sistemas de aprendizado de conceitos existentes, em função das três características básicas do quadro. Dividimos os sistemas de aprendizado de conceitos em quatro enfoques frequentemente citados na bibliografia sobre o assunto. Os sistemas baseados em similaridades, hierarquias, explanações e descobertas.

CAPÍTULO 4

O Ambiente de Apoio a Aquisição Automática de Conhecimento - A4

Neste capítulo analisamos as dificuldades e restrições dos métodos de aprendizado automático baseados em indução e propomos o A4 (Ambiente de Apoio à Aquisição Automática de Conhecimento), com o objetivo de facilitar o tratamento desses problemas.

A idéia que permeia a definição do A4 é prover um ambiente integrado para auxiliar a tarefa de aquisição automática de conhecimento. O ambiente fundamenta-se em funcionalidades e técnicas de aprendizado automático, mas envolve também algumas características dos métodos baseados em entrevistas (semi-automáticos). Com o A4, objetivamos fazer uso de vantagens dos dois enfoques.

Primeiramente, faremos uma análise crítica dos métodos de AC. Essa análise é feita com base na descrição desses métodos realizada no capítulo 2 e serve como justificativa para o desenvolvimento do A4. Após essa análise, enquadramos o A4 no quadro conceitual sobre aprendizado de conceitos definido no capítulo 3. E finalmente, especificamos os seus componenetes, bem como a funcionalidade que cada um deles deve prover.

4.1 Uma Análise dos Métodos Automáticos de AC

Após a dissertação realizada no capítulo 2 deste trabalho, uma análise crítica nos levou a observar os seguintes pontos. Primeiramente, com relação aos métodos semi-automáticos, podemos relacionar como pontos positivos:

- Tentam captar informações semânticas a respeito do conhecimento do especialista, já que simulam o comportamento de entrevistadores humanos.
- Dependendo da ferramenta, o especialista recebe ajuda através de heurísticas e estratégias de descobrimento e diferenciação de conhecimento, o que facilita a identificação de porções de conhecimento

negligenciadas. Esse auxílio procura evitar esquecimentos de algumas partes de conhecimento.

- Existência de uma enorme variedade de aplicações desenvolvidas. Principalmente as tarefas de análise, incluindo tarefas classificatórias, têm sido abordadas por esses métodos. Embora em menor variedade, existem também aplicações para problemas de síntese.
- A maioria das ferramentas têm facilidades para adquirir conhecimento incerto e incompleto.

Como pontos negativos dos métodos semi-automatizados, podemos relacionar:

- A presença do especialista é indispensável e sua atuação quase sempre longa, exaustiva e cansativa. Em função disso, o processo de aquisição de conhecimento torna-se dispendioso.
- A dificuldade do especialista explicitar seu conhecimento através de entrevistas. Isto se deve ao fato do conhecimento heurístico se apresentar no estado compilado. Ou seja, é mais fácil para o especialista solucionar um problema do que explicitar os processos mentais responsáveis por sua solução.
- A maioria dos sistemas baseados em entrevistas geram somente regras atômicas (somente uma condição na premissa). Essas regras não são muito representativas, pois não conseguem expressar situações onde mais de uma condição é necessária para uma determinada conclusão.

Com relação aos métodos de aprendizado automático a nossa análise se restringirá ao escopo dos métodos automáticos indutivos a partir de exemplos, devido à sua maior aplicabilidade à aquisição de conhecimento para sistemas baseados em conhecimento.

Podemos relacionar como vantagens dos métodos automáticos indutivos:

- Minimizar ou até ignorar a presença do especialista, que é difícil de ser conseguida e quase sempre muito cara.
- Em alguns domínios é mais fácil adquirir conhecimento a partir de casos reais (exemplos). Além do mais, o especialista pode achar mais fácil relacionar casos de um domínio do que seguir o seu processo mental em uma resolução.
- Ter o poder de captar conhecimento através de generalizações e induções. Isso faz os algoritmos generalizadores parecerem inteligentes na tarefa de aprender.

Quanto aos pontos negativos dos métodos automáticos indutivos, podemos dizer que:

- Há uma suposição básica de que as instâncias ou exemplos (normalmente pares atributo/valor), que servem como entrada para os algoritmos, são conhecidos de antemão. A prática tem mostrado que essa suposição não é fácil de ser satisfeita. A identificação dos atributos que formam os exemplos e a consequente construção do conjunto de treinamento também tem se mostrado um ponto de estrangulamento no processo de aquisição.
- Há uma série de restrições e limitações, embora distintas de algoritmo para algoritmo, que têm que ser cumpridas na modelagem do mundo real para que possam ser aplicados algoritmos generalizadores. Estas restrições vão desde problemas de eficiência computacional a exigências conceituais pertinentes ao algoritmo.
- A maioria desses métodos são puramente sintáticos: não usam informações semânticas sobre o domínio. Os algoritmos de indução usuais generalizam a partir de dados, sem referência ao conhecimento preliminar [Núñez 91] [Holland 86]. O conhecimento preliminar pode ser eliciado do especialista de várias formas diferentes, como relevância semântica [Mongiovi 90a], grafos de hierarquia IS-A e informação relativa ao custo dos atributos [Núñez 91]. Ele contém informações relativas ao domínio, mas não encontradas entre os dados. Sua utilidade está em facilitar o trabalho do algoritmo de indução e melhorar a qualidade do conhecimento induzido.
- Um ponto negativo específico dos métodos indutivos que geram árvores de decisão é que a saída gerada pode ser pouco inteligível, pois árvores de decisão normalmente provocam a introdução de condições artificiais às conclusões [Cendrowska 88].

4.2 Objetivos e Características do A4

O A4 tem a finalidade de integrar os enfoques analisados anteriormente (técnicas semi-automáticas e automáticas), tentando preservar suas vantagens e minimizar suas desvantagens. O enfoque principal é centrado nos métodos de aprendizado automático indutivo. Os métodos baseados em entrevistas auxiliam a modelagem do domínio [Vascol 92b].

O A4 objetiva:

- Preencher o vazio existente na maioria dos métodos indutivos, com respeito à obtenção e utilização de conhecimento preliminar.
- Prover um ambiente flexível, que possa abrigar diversos tipos de algoritmos generalizadores para avaliações comparativas entre eles e, consequentemente, permitir o uso do mais adequado a cada problema.
- Facilitar a tarefa de construção da tabela de exemplos para, assim, tornar mais simples e eficiente o processo de aquisição de conhecimento.
- Prover um ambiente que permita fácil extensão pela inclusão de novas ferramentas.
- Minimizar a presença do especialista e do engenheiro do conhecimento no processo de aquisição de conhecimento.

Para atingir os objetivos anteriormente relacionados, o ambiente provê as seguintes funcionalidades:

- Dispor de diferentes tipos de entrevistas focadas para a eliciação dos diversos tipos de conhecimento preliminar. O conhecimento preliminar não constitui uma forma "completa" de representar o conhecimento sobre um dado domínio. Sua intenção é captar apenas alguns aspectos objetivos do conhecimento. Essa objetividade é responsável por facilitar sobremaneira a sua eliciação, o que permite uma melhoria significativa na qualidade da indução automática, sem recair na exaustão inerente às entrevistas semi-automáticas.
- Auxiliar o especialista/engenheiro do conhecimento na identificação de atributos em diferentes níveis de abstração, a partir de conhecimento preliminar, e da análise da correlação entre os atributos.
- Identificar problemas na composição da tabela de exemplos que exigem tratamento anterior à aplicação de um determinado algoritmo generalizador, como: preencher "buracos" (*missing values*), identificar e confirmar se os contra-exemplos são realmente desejados (se o algoritmo aceitar contra-exemplos), identificar e tratar ambiguidades da tabela de exemplos, evitar a interdependência dos atributos, garantir a exclusão mútua de valores dos atributos, etc.
- Prover mecanismos de validação dos resultados obtidos pelos algoritmos generalizadores.
- Tornar as saídas geradas pelos algoritmos mais claras e compreensíveis, através da utilização de informações semânticas do especialista. Pretende-se, assim, aumentar a inteligibilidade do conhecimento gerado.

- Prover mecanismos de simplificação, para tornar mais simples e eficientes as saídas geradas pelos algoritmos de indução.
- Auxiliar na combinação de resultados gerados por diferentes algoritmos.

4.3 O A4 dentro do Quadro de Aprendizado de Conceitos

Uma análise do A4 em função das três característica básicas definidas no quadro conceitual do capítulo 3, nos mostrou os seguintes pontos:

Em relação à *representação de conhecimento*, se utiliza uma representação baseada em lógica para os exemplos, com pares de atributo/valor que concluem o elemento classificador. Os conceitos podem ser gerados em árvores de decisão, regras de produção, grafos de conhecimento ou listas de decisão.

Em relação às *tendências,* como o A4 se destina a métodos indutivos baseados em exemplos, ele utiliza tendência conceitual, com os atributos tendo que possuir valores nominais. Mas para auxiliar o usuário no cumprimento dessa restrição, é possível entrar com atributos com valores contínuos, que são posteriormente tratados pelo ambiente para que tomem a forma exigida.

Com relação às tendências em função de conhecimento preliminar, a idéia do A4 é incentivar o uso deste tipo de conhecimento pelos algoritmos indutivos. Para isso o A4 auxilia na modelagem de relevância semântica, custo e hierarquias, além de possibilitar a inclusão de novas ferramentas que modelem outras formas de conhecimento.

Em relação à *forma de interação com o instrutor*, buscamos simular no A4 um *instrutor inteligente*. Através do objeto *entrevistador* o ambiente auxilia a apresentação dos exemplos. O ambiente sugere a ordem da apresentação dos exemplos, pesquisa a frequência de aparecimento de exemplos em função de elementos classificadores e/ou de pares atributo/valor, busca adquirir exemplos representativos e identifica se os exemplos apresentados são ambíguos ou não.

4.4 A Arquitetura do A4

A figura 4.1 mostra a arquitetura do ambiente A4 com seus componentes.

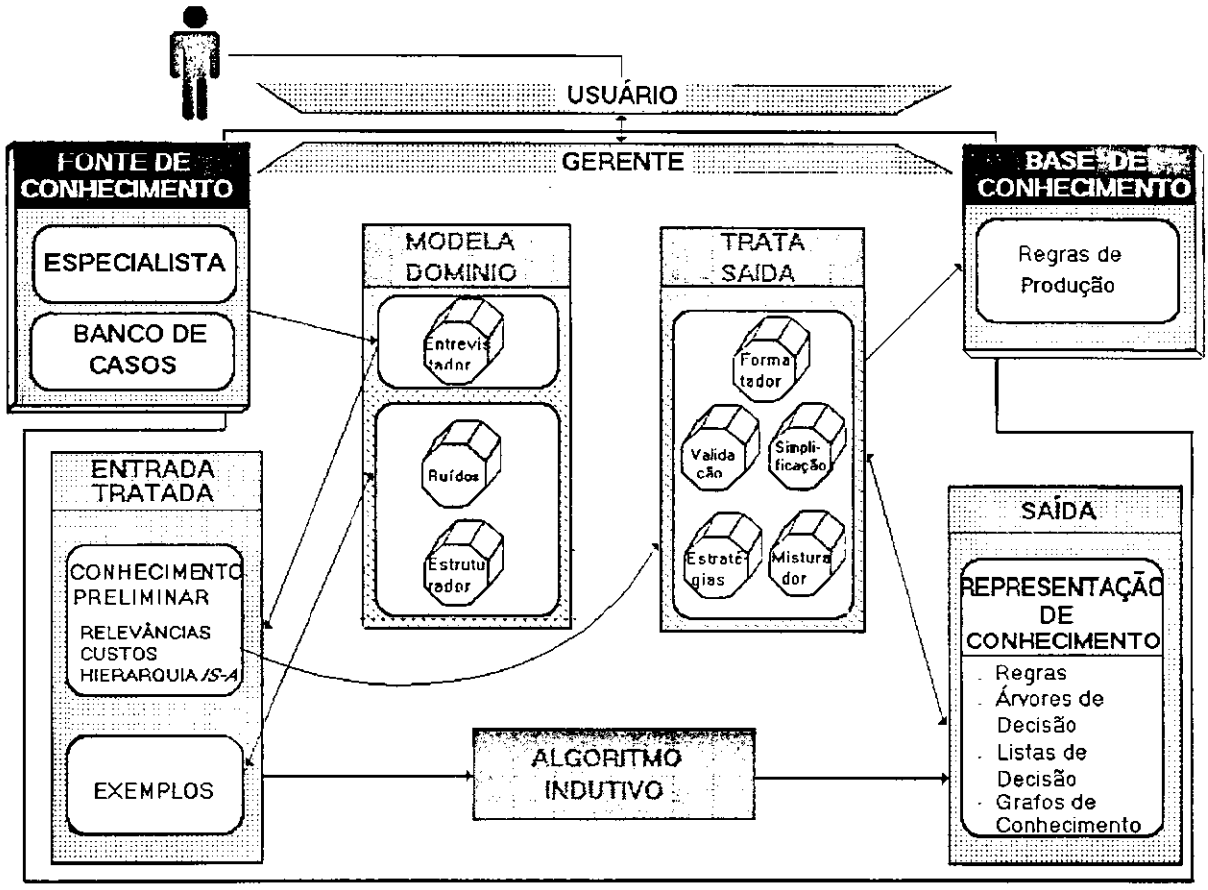


Figura 4.1 Arquitetura do A4

O ambiente não tem uma estrutura estritamente sequencial de atuação. O usuário, através do *gerente*, pode determinar uma ordem própria de ativação dos objetos. Por exemplo, após a geração de uma saída por um determinado algoritmo generalizador, o usuário pode ativar as opções de simplificação e validação dessa saída. Mas, em qualquer momento, ele pode decidir remodelar a entrada. A tarefa de aquisição de conhecimento será realizada em várias e possivelmente repetidas etapas. Esta característica iterativa reflete uma propriedade importante do processo de aquisição

de conhecimento que é a incrementabilidade. Ou seja, o conhecimento é adquirido aos poucos, com sucessivos refinamentos e incrementos de novas informações.

Dois componentes da figura 4.1 são externos ao A4: a *fonte de conhecimento* e a *base de conhecimento*. A arquitetura contempla as informações relativas a esses componentes através das classes *entrada tratada* e *saída*, respectivamente. A *entrada tratada* serve como área de trabalho à modelagem do domínio. Já a idéia de separar a *base de conhecimento* da *saída* objetiva compatibilizar as mais diversas formas de conhecimento que podem ser geradas pelos algoritmos. A *base de conhecimento* tem um formato padrão, baseado em regras de produção, o que facilita o uso aos diversos *shells* do mercado.

A seguir definiremos os componentes do A4. Uma pequena descrição de suas funcionalidades acompanha as definições.

4.4.1. Gerente

O *gerente* tem como função básica integrar todos os componentes do ambiente e controlar sua utilização. A infra-estrutura básica para o funcionamento do ambiente, bem como as funcionalidades para criação e manutenção dos objetos também cabem ao *gerente*. A figura 4.2 mostra suas principais funções.

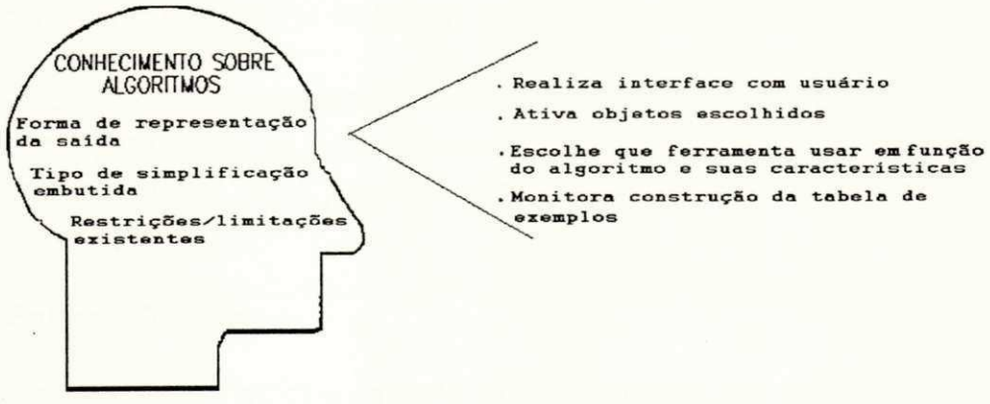


Figura 4.2. O conhecimento do gerente

É o *gerente* que se responsabiliza pela interface com o usuário. Há a necessidade dele estar sempre apto a receber algum evento, já que o usuário pode escolher opções de processamento em diferentes sequências.

Além da interface com o usuário, o *gerente* deve conter informações sobre os algoritmos generalizadores (por exemplo, que tipos de problemas resolvem, quais as restrições que impõem à modelagem do mundo real e que tratamentos podem ser aplicados às suas saídas). É o gerente que controla a execução dos algoritmos e antes de realizar a chamada do algoritmo ele checa se as exigências que o algoritmo faz estão satisfeitas.

4.4.2. Algoritmo Indutivo

A classe *algoritmo indutivo* é fundamental para o ambiente, mas a arquitetura não provê nenhum algoritmo novo. Na verdade, os algoritmos são tratados como caixas pretas. Há somente mecanismos para incluir e retirar algoritmos. É necessário que, ao se incluir um determinado algoritmo, ele se adeque à estrutura básica da arquitetura. Ou seja, os algoritmos têm que estar preparados para acessar as entradas e gerar as saídas dentro do formato definido pelo A4. Para isso, basta utilisar os métodos reutilizáveis definidos no A4 e que são descritos no apêndice A dessa dissertação. Além disso, é necessário que se acrescente conhecimento ao *gerente* sobre as características básicas do novo algoritmo, como: tipo de representação de conhecimento usada, se realiza poda, quais limitações exige da entrada, se já trata "buracos", etc.

O A4 está instanciado com algoritmos regulares PRISM, RPRISM e RNPRISM e com o ID3. Os três primeiros fazem parte da classe *Algoritmos regulares* (algoritmos que geram diretamente regras de produção) que não é uma classe básica do ambiente. Ela foi incluída com o propósito de fatorar características comuns a estes algoritmos.

O desenvolvimento desses algoritmos e sua implantação no A4 nos serve também como exemplo para possíveis inclusões de outros algoritmos no ambiente. Os passos que devem ser seguidos pelos projetistas de algoritmos estão definidos no apêndice B desta dissertação.

4.4.3. Entrada Tratada

A *entrada tratada* é um modelo da fonte de conhecimento. Ela pode ser manuseada pelos objetos da classe *modela domínio* para se adequar ao algoritmo desejado. As informações podem estar na *entrada tratada* sob duas formas: exemplos e conhecimento preliminar. Inicialmente, o A4 está preparado para trabalhar com três tipos diferentes de conhecimento preliminar: relevância semântica, custos e informações hierárquicas. Contudo, o tratamento de outros tipos pode ser adicionado ao ambiente.

4.4.4. Modela domínio

A camada pré-indução constitui-se da classe *modela domínio*. Ela possui subclasses com objetos que atuam na *entrada tratada*, antes da mesma ser utilizada pelos algoritmos generalizadores. Esses objetos têm a finalidade de auxiliar a modelagem do mundo real, facilitando a criação da tabela de exemplos, eliciando conhecimento preliminar sobre o domínio e adequando as entradas à aplicação de um determinado algoritmo. A classe *modela domínio* é composta por três sub-classes: a classe de objetos que trata da estruturação do exemplo, a classe de objetos que trata ruídos e a classe entrevistadora (uma descrição mais detalhada da classe modela domínio será vista no capítulo 5).

A classe *estruturador* auxilia a modelagem do domínio com objetos que identificam e refinam os atributos que compõem a estrutura do exemplo. Esses objetos propõem agrupamentos de atributos, realizam discretizações de valores contínuos, identificam interdependência de valores entre atributos e adequam a estrutura do exemplo às restrições impostas pelo algoritmo generalizador.

A classe *entrevistador* responsabiliza-se por eliciar do especialista os exemplos, bem como o conhecimento preliminar sobre o domínio. O processo de entrevista pode ser guiado pelo entrevistador para que se selecione exemplos representativos do domínio em questão.

A classe *trata-ruído* visa identificar e, se possível, tratar os ruídos que podem aparecer na formação da tabela de exemplos. Ruídos, como *buracos (missing values)* e contra-exemplos, são examinados.

4.4.5. Trata Saída

A classe *trata saída* possui os objetos que atuam nas saídas dos algoritmos. Além do que pode ser acrescentado, estão previstos os seguintes objetos:

4.4.5.1. Validador

O *validador* terá a função de testar a saída gerada pelo algoritmo generalizador quanto a sua eficiência e eficácia. Esta validação pode necessitar de mais semântica do especialista, o que pode forçar a comunicação do *validador* com o especialista. Outra possibilidade de adicionar semântica na validação é utilizar as informações preliminares, como relevância semântica, adquirida pela classe *modela domínio* junto ao especialista.

Além dessa validação semântica, pode-se submeter casos diferentes aos do conjunto de treinamento, para identificar o grau de acurácia da saída gerada. Para essa validação ser realizada é preciso aplicar um mecanismo de inferência nas saídas geradas pelo algoritmo.

4.4.5.2. Simplificador

Após a execução do algoritmo generalizador, obtém-se uma saída em alguma forma de representação de conhecimento. Essa saída pode ser aperfeiçoada através da diminuição do número de condições que levam à classificação dos exemplos. Isso pode acontecer principalmente em domínios em que se encontram ruídos. Nesses domínios, como os algoritmos tendem a gerar saídas que reflitam o domínio, verifica-se uma grande quantidade de conceitos destinados a cobrir casos específicos e de pouca importância, obtidos por causa dos ruídos. Eliminar essas partes pouco importantes ou até irrelevantes é função do *simplificador*. Essa tarefa, também conhecida como poda, objetiva tornar a tarefa de classificação mais simples e eficiente. O que se observa é que a poda de uma saída, embora implique em perda de conteúdo sobre algumas classificações, no geral torna a classificação melhor realizada. Existem duas alternativas de realizar poda. De uma maneira geral, a poda depois do algoritmo ter gerado as saídas é mais aconselhável [Niblet 87] [Quinlan 87a], mas algumas implementações são feitas no próprio algoritmo generalizador [Quilan 87b].

Outro ponto importante quando se realiza simplificações nas saídas é que, em geral, após a simplificação, o conhecimento obtido está mais compreensível. Prover mais compreensão é fundamental para se dizer que um sistema de aprendizado realmente adquiriu conhecimento. A clareza dos resultados auxilia, por exemplo, a prover boas explanações sobre classificações realizadas.

A tarefa de simplificação pode ser realizada, por exemplo, pela eliminação de arcos (ligação entre dois nós) de uma árvore de decisão ou de condições de uma regra. O *simplificador* pode também receber informações semânticas vindas do especialista ou fazer uso do conhecimento já adquirido sobre o domínio, como a relevância semântica.

4.4.5.3. Misturador

O *misturador* é o objeto que integra as várias soluções geradas por algoritmos diferentes para um mesmo domínio. Suponha que um conjunto de regras de produção é gerado de uma árvore de decisão após a aplicação de um algoritmo da família TDIDT. O *misturador* poderá comparar (com auxílio de conhecimento preliminar ou do especialista, através de entrevistas) que regras equivalem ou são contrastantes em relação a um outro conjunto de regras advindo, por exemplo, de algoritmo tipo PRISM

[Cendrowska 88]. Essas comparações podem gerar um conjunto de regras mais robusto e de melhor qualidade.

4.4.5.4. Estrategista

As estratégias são conhecimento sobre que ação tomar a seguir em uma dada situação. Essas decisões são tomadas considerando-se fatores que influenciam a qualidade da escolha.

Um exemplo do uso de estratégias pode ser visto num domínio médico. A escolha de que teste um paciente deve realizar para identificar uma certa disfunção é povoada de estratégia. O médico não solicita logo ao paciente aquele teste que pode dar o resultado com total precisão, pois fatores outros podem estar envolvidos como, custo do teste, fator de risco do teste e o estado do paciente. Por exemplo, uma tomografia computadorizada, devido a seu custo, só é sugerida após a tentativa de descoberta do problema por outras formas. O exemplo em questão contempla a estratégia ligada a custo. Ou seja, se o teste é potencialmente conclusivo, mas o custo é muito alto, não o realize inicialmente. Outros tipos de estratégias podem ser vislumbrados, sendo que de uma forma geral relacionam-se a escolhas de que ações tomar a seguir.

Dentre as diversas formas de representar estratégias [Gruber 89], duas estão previstas no A4 : meta-regras que controlam a ativação de regras de produção formatadas na *base de conhecimento*, e modificações na topologia ou ordem dos atributos da própria representação de conhecimento gerada pelo algoritmo indutivo.

4.4.5.5. Formatador

A classe de objetos que fazem formatação se responsabiliza por transformar a *saída* (representação de conhecimento intermediária) no formato da base de conhecimento. Aqui é contemplada a geração de regras de produção a partir de árvores de decisão, grafos de conhecimento e outras representações.

4.4.6. Saída

A classe *saída* contém objetos que armazenam as saídas advindas dos algoritmos. Ela deve aceitar diferentes formas de representação de conhecimento, que serão utilizadas dependendo do algoritmo que gerará esse conhecimento. As formas de representação previstas são: regras de produção (PRISM [Cendrowska 88], RPRISM [Cirne 92]), árvores de decisão (ID3 [Quinlan 86], APREND [Gomes 89], C4 [Quinlan 87b]), listas de decisão (CN2 [Elomaa 89]) e grafos acíclicos de conhecimento [Machado 88], mas nada impede que outras sejam agregadas.

CAPÍTULO 5

Modelagem do Domínio para os Métodos Indutivos de AC

Os métodos automáticos indutivos consideram o conjunto de exemplos e até o conhecimento preliminar sobre o domínio (quando usado), como já disponíveis [Quinlan 87b] [Cendrowska 88] [Cirne 92] [Núñez 91]. Na verdade, há uma lacuna entre como essas entradas se apresentam no mundo real e como elas se tornam operacionais aos algoritmos indutivos. Por exemplo, a identificação dos atributos mais representativos que formam os exemplos e a consequente construção do conjunto de treinamento (tabela de exemplos) tem se mostrado uma tarefa não trivial[Mongiovi 90a] [Hart 87]. Essa tarefa, se não for devidamente informatizada, diminui acentuadamente o caráter automático da aquisição de conhecimento.

Para preencher a lacuna acima citada, neste capítulo propomos formas de auxiliar a modelagem do mundo real em formas tratáveis pelos métodos indutivos de aquisição de conhecimento. A modelagem resultará em exemplos devidamente estruturados e em conhecimento preliminar sobre o domínio, como relevância semântica [Cirne 91], custos e hierarquias de atributos [Núñez 91]. O processo dessa modelagem é realizado pela classe de objetos *modela domínio*, que está inserida no A4.

5.1 Em Que Modelar o Domínio

Os trabalhados realizados em aquisição de conhecimento indutiva têm enfocado quase que exclusivamente o processo de generalização em si. Entretanto, para que os métodos indutivos de aquisição de conhecimento apresentem resultados satisfatórios, é necessário que as entradas dos algoritmos generalizadores reflitam ao máximo o domínio do problema a ser resolvido. Podemos aqui reafirmar a tradicional máxima de processamento de dados de que "se entra lixo, sai lixo".

No processo de aquisição de conhecimento via técnicas indutivas, o domínio tem sido modelado na forma de exemplos e conhecimento preliminar. Sendo que a primeira

é a única forma de entrada para a maioria dos métodos indutivos. A seguir, apresentamos uma descrição conceitual dessas duas formas.

5.1.1 Exemplos

A definição mais comumente encontrada na literatura é que um exemplo ou caso é um conjunto de pares atributo/valor e um elemento de classificação. Tabela de exemplos é uma sequência de exemplos advindos diretamente do especialista ou de uma outra fonte de conhecimento (banco de dados, fichários, etc).

Os atributos que compõem um exemplo podem ter valores lineares (ordinais ou contínuos), nominais ou estruturados hierarquicamente [MacDonald 89].

A seleção de quais atributos e valores devem compor a estrutura do exemplo tem que ser feita de forma criteriosa, pois eles devem ser representativos do domínio em questão. Além disso, se persegue uma racionalização nos valores dos atributos, evitando, por exemplo, valores contínuos.

Na modelagem dos exemplos devemos levar em consideração as restrições impostas pelos métodos de aprendizado automático que os utilizam. Algumas dessas limitações são em decorrência de limitações intrísecas aos próprios algoritmos. Por isso, elas podem variar de algoritmo para algoritmo, como a aceitação ou não de "buracos" (*missing values*), onde divergem o ID3 (que não os aceita) [Quinlan 86] e o C4 (que os trata) [Quilan 87b].

Contudo, a mais generalizada limitação dos métodos de aprendizado automático reside na possibilidade de crescimento do espaço de soluções do problema. Esta situação ocorre, principalmente, em domínios não determinísticos e que apresentam um grande número de atributos. Nesses casos, além do grande esforço computacional exigido, as saídas podem se tornar por demais complexas e, consequentemente, incompreensíveis ao especialista.

5.1.2 Conhecimento Preliminar

O conhecimento preliminar ou informações semânticas sobre o domínio constituem conhecimento do especialista sobre o problema a ser resolvido. Estas informações devem complementar o conhecimento que é capturado somente através de informações sintáticas, como os exemplos. Na semântica do especialista é que está embutido o conhecimento heurístico.

5.1.2.1 Relevância Semântica

A relevância semântica relaciona, em um dado domínio, elementos de classificação com os atributos e seus respectivos valores. Relevância, de um modo geral, indica a importância que tem um determinado par atributo/valor para concluir um elemento de classificação. Os algoritmos generalizadores têm, na sua maioria, condições de descobrir relevância sintática através de estatísticas. Como essa relevância nem sempre reflete totalmente o conhecimento do especialista, dá-se a necessidade do uso de relevância semântica. A sua ausência fica especialmente evidente quando há casos raros no conjunto de treinamento ou quando se trabalha com conjuntos de treinamento pequenos, o que pode fazer com que coincidências sejam generalizadas como conhecimento [Cirne 92].

Uma forma de representar a relevância semântica é através de uma matriz composta pelos atributos e elementos de classificação, denominada matriz de relevância (MR) [Mongiovi 90a].

A matriz de relevância pode ser também uma Matriz de Relevância Nebulosa (MRN) [Mongiovi 93], na qual a relação entre o elemento classificador e o par atributo/valor é representada por um conjunto nebuloso. Com a MRN podemos modelar termos linguísticos como muito relevante, mais ou menos relevante e pouco relevante e assim representar mais fidedignamente o quão um atributo é relevante para a conclusão de

Classe / Atributo	Sem Predisposição	Pouca Predisposição	Média Predisposição	Muita Predisposição
Sexo				
Colesterol	Normal	Normal		Alto
Stress		Baixo	Alto	Alto
Glicose	Normal			
Pressão				Alto
Fumante		Não	Sim	Sim

Matriz de Relevância

Matriz de Relevância Nebulosa

Classe / Atributo	Sem Predisposição	Pouca Predisposição	Média Predisposição	Muita Predisposição
Sexo	0.1 /Masc.			
Colesterol	0.8 /Normal	1.0/Normal	0.7/Alto	1.0/Alto
Stress	0.7 /Baixo	0.9/Baixo+0.2/Normal	0.1/Normal+0.7/Alto	0.5/Alto
Glicose	0.7 /Normal			
Pressão	0.6 /Normal		0.7/ Alto	0.8/Alto
Fumante	0.6 /Não	0.7/Não	0.8/ Sim	0.7/ Sim

Figura 5.1 Exemplos de matriz de relevância

uma classe. A figura 5.1 mostra um exemplo de uma MRN e uma MR para um domínio cardiológico. Podemos, comparando as duas matrizes, observar que a MRN possibilita expressar fatos que a MR não permite. Por exemplo, na MRN podemos indicar que

stress é *mais ou menos relevante* (grau de pertinência 0.5) para concluir *muita_predisposição*, o que é impossível de ser feito numa MR.

5.1.2.2 Informação sobre Custo

Informações sobre o custo de cada atributo é outro tipo de conhecimento que não é contemplado pela maioria dos métodos indutivos. As informações sobre custos não são necessariamente monetárias, mas podem ser vistas também em função de risco de vida, dificuldade de aplicação, urgência da informação, etc.

Os métodos indutivos de aquisição de conhecimento que geram árvores de decisão podem fazer uso das informações sobre custo dos atributos, no momento da escolha dos atributos que compõe a árvore. Esses métodos devem levar em consideração não apenas a quantidade de informação de um dado atributo, mas uma relação entre custo e benefício desse atributo.

5.1.2.3 Informação sobre Hierarquia

Através de modelos que reflitam a organização hierárquica dos atributos do domínio, pode-se facilitar o aprendizado a partir de exemplos. Uma possibilidade disto pode ser vista em [Núñez 91], no qual uma rede de dependência *IS-A* entre os valores dos atributos auxilia a substituição de atributos apresentados nos exemplos por informações mais genéricas. Informações hierárquicas são úteis para se definir o grau de generalização, ou o nível de abstração que se deseja trabalhar. Este tipo de informação influencia fortemente a quantidade de pesquisa que o algoritmo tem de realizar. A figura 5.2. mostra um exemplo de como informações hierárquicas podem ser usadas por algoritmos indutivos para gerar árvores de decisão menores. Este exemplo contempla um tipo de generalização chamada *climbing generalization tree* [Michalski 83].

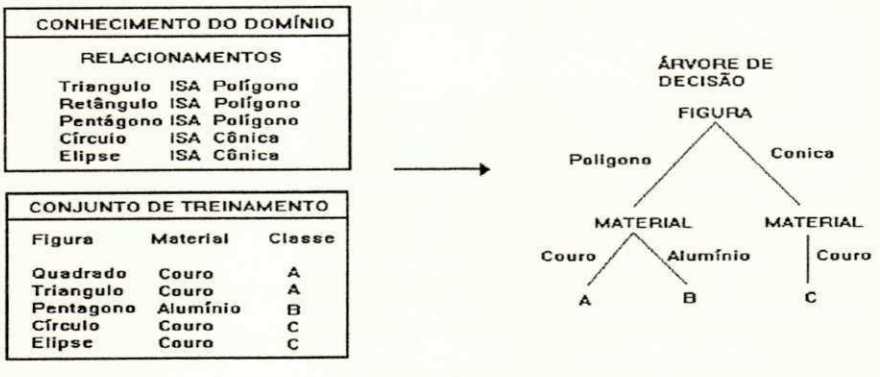


Figura 5.2 Hierarquias otimizando árvores de decisão

5.2 A Classe MODELA-DOMÍNIO

A classe *modela domínio* possui subclasses com objetos que representam o mundo real em entradas tratáveis pelos algoritmos generalizadores. Esses objetos facilitam a criação da tabela de exemplos e a eliciação de conhecimento preliminar, adequando-os à aplicação de um determinado algoritmo. A classe *modela domínio* é composta por três sub-classes: a classe de objetos que trata de "entrevistar" o especialista , a de estruturar o exemplo e a classe que trata ruídos.A seguir faremos uma descrição de cada uma delas.

5.2.1 Entrevistador

A classe *entrevistador* é responsável por entrevistar o especialista para identificar quais são os atributos, valores e o elemento de classificação que compõem a estrutura de um exemplo. Nessa entrevista são feitas sugestões de quais exemplos devem ser selecionados e de como estruturá-los. Além disso, cabe à classe *entrevistador* a eliciação de conhecimento preliminar sobre o domínio. Podemos dividir a tarefa do entrevistador em duas. Uma primeira função é a de tutorial do processo de modelagem do domínio. Mensagens orientando que exemplos devem ser apresentados ou que outros passos devem ser tomados dirigem a entrada de informações. A outra função está integrada com a interface do ambiente pois no processo de entrevista se utiliza intensamente os recursos da interface gráfica.

5.2.2 Estruturador

A classe estruturador tem a tarefa de auxiliar a criação da tabela de exemplos, através da identificação e refinamento da estrutura desses exemplos. Isso é realizado através das subclasses: *compõe atributo, trata interdependência de valor, trata exclusão mútua* e *discretizador*.

5.2.2.1 Compõe-atributo

Em princípio, a modelagem de certos domínios pode requerer um grande número de atributos. Conforme visto na seção 5.1, dentro do possível, isto deve ser evitado. Uma estratégia possível para trabalhar com um número menor de atributos é compor (fundir) dois ou mais atributos em um só. Isto pode ser feito com atributos que não ocorram simultaneamente na conceituação de um exemplo. Caso os atributos ocorram

simultaneamente, podemos então tentar fundi-los e combinar seus valores. Esta estratégia está ilustrada na figura 5.3.

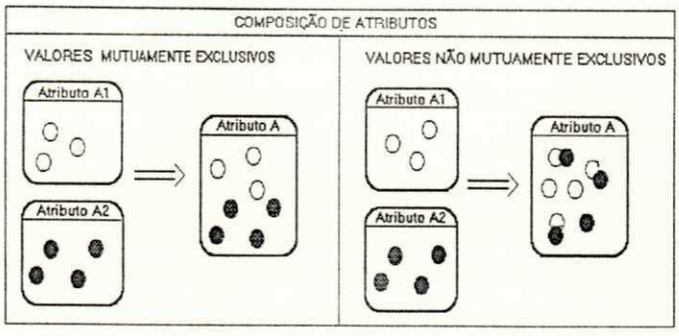


Figura 5.3 Estratégias de composição de atributos

Algumas heurísticas são usadas para identificar atributos que podem ser agrupados. Por exemplo, atributos que aparecem pouco na tabela de exemplos (provavelmente têm baixo grau de importância), são candidatos em potencial ao agrupamento. Outra heurística possível é agrupar dois atributos que contenham poucos valores (dois ou três). Com essa exigência evitamos uma explosão combinatorial no número de valores do atributo resultante. Contudo, mesmo que a quantidade de combinações seja grande, podemos ainda eliminar aquelas que são impossíveis de ocorrer na prática.

5.2.2.2 Trata Interdependência de valor

Também perseguindo diminuir o número de atributos que compõem a estrutura do exemplo, podemos buscar a identificação de atributos que mantenham uma certa dependência entre si. Isso acontece quando, numa tabela de exemplos de um certo domínio, percebemos que toda vez que um atributo **A** possui valor *a* um outro atributo **B** possui valor *b*. *Trata interdependência de valor* identifica esse relacionamento, e os atributos envolvidos são, então, candidatos naturais ao agrupamento. Esse tipo de auxílio serve também para identificar níveis hierárquicos diferentes entre os atributos, pois os atributos que estão se mostrando dependentes podem ser substituídos por um outro de nível de abstração maior.

5.2.2.3 Discretizador

Outra característica fundamental de vários métodos indutivos é a necessidade de discretizar valores contínuos (transformar valores lineares em nominais). Isso é fundamental no desempenho dos algoritmos. O discretizador examinará os valores dos atributos para classificá-los dentro de uma determinada escala. Na prática, nem sempre é trivial identificar *a priori* em quantas faixas os valores de um certo atributo devem ser divididos. Observe que a importância de um mesmo atributo é contextual. Por exemplo, num contexto legal geralmente é suficiente discretizar o atributo idade em *menor* (<18 anos) e *maior* (>18 anos). Já num contexto desportivo, poderia se ter os seguintes valores *infantil* (< 12), *juvenil* (de 12 a 18), *adulto* (de 18 a 35) e *veterano* (> 35).

A idéia do objeto *discretizador* no A4, é de auxiliar o usuário na tarefa de transformar valores contínuos em discretos. Quando da definição da estrutura dos exemplos, o usuário deve determinar quais os atributos ordinais(valores contínuos). Posteriormente, na alimentação dos exemplos, estes atributos são preenchidos com os seus valores contínuos tais como se apresentam no mundo real. As tabelas de exemplos definidas com atributos com valores ordinais, antes de serem usadas por algum algoritmo generalizador, são analisadas e então são sugeridos intervalos os quais podem ser adotados como valores nominais para esses atributos. O usuário pode aceitar as sugestões do discretizador ou definir os valores que ele achar mais conveniente. Com esse processo de discretização perseguimos manter o princípio básico que norteia a classe *modela domínio*: facilitar a modelagem do mundo real.

5.2.2.4 Trata exclusão mútua

Existem restrições e limitações impostas à modelagem do mundo real que são específicas de cada algoritmo. Uma restrição de vários algoritmos é que os valores dos atributos sejam mutuamente exclusivos. Uma solução possível para essa exigência é dividir o atributo em dois e redistribuir seus valores. Caso contrário, os valores não mutuamente exclusivos são combinados. Na criação da estrutura do exemplo, quando da definição dos valores dos atributos, o A4 informa a necessidade da exclusão mútua dos atributos e sugere alternativas de tratamento. Além de auxiliar a modelagem de valores de atributos, o *trata exclusão mútua* pode identificar se dois determinados atributos são mutuamente exclusivos, o que indica a viabilidade da composição destes atributos.

5.2.3 Trata ruído

Os ruídos são informações danosas à execução dos algoritmos. Eles acontecem porque na grande maioria dos domínios do mundo real não há uma garantia de que, num exemplo, a correlação entre os pares atributo/valor e os respectivos elementos de classificação é fidedigna. A tabela de exemplos pode conter contra-exemplos, o que pode prejudicar ou até impossibilitar a execução de muitos algoritmos generalizadores (por exemplo , os da família TDIDT). Alguns desses ruídos resultam de falhas em medições, valores ausentes, limites mal definidos (ex: Quanto uma pessoa é alta ?) e interpretações subjetivas (ex: Que critérios são usados para descrever uma pessoa como atlética ?). Há uma enorme subjetividade envolvida na identificação de ruídos, dificultando assim um tratamento genérico. Entretanto, alguns tipos de ruído podem ser identificados e tratados. Dois dos mais frequentes são: "buracos" e ambiguidades.

UFPb/BIBLIOTECA/PRAI

5.2.3.1 Trata buraco

Os buracos (*missing values, unknown values*) são encontrados quando há atributos não valorados na tabela de exemplos. Este problema é extremamente sério e deve ser contornado, pois informações faltando podem levar a resultados incorretos. Uma das dificuldades ao se tratar os "buracos" é que eles podem acontecer por razões diversas. Uma razão é devido à impossibilidade de se obter o valor do atributo. A outra alternativa é porque o atributo não é necessário para o exemplo. E finalmente porque a presença do atributo é indiferente na conclusão do exemplo.

Trata buraco é o responsável em contornar esses problemas, considerando essas diversas perspectivas e as características específicas de cada algoritmo generalizador.

5.2.3.2 Trata ambiguidade

Em algumas situações, a tabela de exemplos pode apresentar casos com um mesmo conjunto de pares atributo/valor concluindo classes diferentes. Essa situação pode caracterizar um ruído na tabela de exemplos, um caso intencional de contra-exemplo ou mesmo uma falha na estrutura do exemplo (situação em que um outro atributo pode distinguir os exemplos, desfazendo, assim, a ambiguidade). Nessas ocasiões, *trata ambiguidade* alerta o usuário, auxiliando-o, quando necessário, na solução do problema.

O trata ambiguidade guarda informações da existência de contra exemplos numa tabela de exemplos, pois essa informação será útil para o gerente decidir se um determinado algoritmo poderá ou não usar essa tabela de exemplos.

5.3 Como Realizar a Modelagem

Em situações reais, observamos que a obtenção de uma estrutura que identifique com exatidão os exemplos, com seus atributos e valores, tais quais se apresentam no mundo real, apresenta alguma complexidade. A seleção de quais atributos e valores devem compor um exemplo, tem que ser feita levando-se em consideração as limitações impostas pelos métodos de aprendizado automático que os utilizam.

No A4, a modelagem do mundo real é fundamentada em interações do especialista e/ou engenheiro de conhecimento com o ambiente. A filosofia desse processo interativo é exigir o mínimo do especialista. Ele é necessário somente para construir a

estrutura da tabela de exemplos, fornecer algumas características semânticas e, posteriormente, validar os resultados obtidos. Essa participação é considerada pequena em relação aos métodos semi-automáticos, e não empobrece os resultados que os algoritmos generalizadores provêem.

5.3.1 Alternativas de eliciação

No ambiente A4, a aquisição dos elementos necessários à formação da estrutura de um exemplo e do conhecimento preliminar tem duas alternativas distintas de ser realizada.

1) É feita a alimentação do conjunto de casos sem a formação de uma estrutura prévia dos exemplos. A medida que os casos vão sendo fornecidos, os atributos e seus respectivos valores vão sendo cadastrados.

2) Divide-se em duas fases explícitas.

2a) Criação de uma estrutura básica para os exemplos. Nesta fase, os objetos que tratam as entradas estão ativos e podem propor modificações na estrutura dos exemplos. Esses objetos também guiam o processo de alimentação dos casos, auxiliando para que sejam selecionados casos mais representativos e, provavelmente, com atributos mais significativos.

2b) Alimentação do grande contingente de casos. Onde os exemplos são informados sem que sejam feitas modificações na estrutura do exemplo.

Outras alternativas poderiam existir, mas na realidade seriam apenas uma variação das duas citadas.

Em ambas as alternativas anteriores, ao final da alimentação de todos os exemplos, é feita uma nova crítica nas entradas e gerada uma versão para validação pelo especialista.

A escolha da alternativa mais adequada de realizar o processo de criação da tabela de exemplos é circunstancial. O que verificamos como ponto positivo na alternativa 2 é que ela possibilita uma maior liberação do especialista, que é necessário somente na fase de estruturação e ao final na validação. Nesta opção, a fase de validação provoca poucas alterações na estrutura inicial, visto que na formação da estrutura do exemplo foram usados casos selecionados e mais representativos. Já a alternativa 1 deve ser escolhida quando se trabalha em domínios onde os exemplos se apresentam com uma certa organização. Nesse caso, o processo de alimentação poderá então

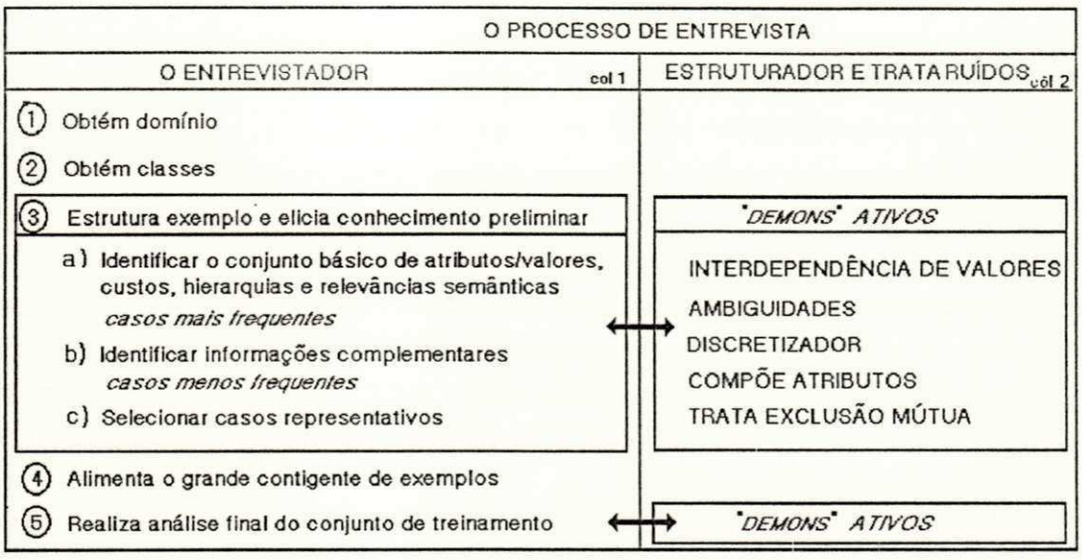


Figura 5.4 O processo de entrevista

ser realizado por um usuário com pouco conhecimento do domínio, não necessariamente o especialista.

O conhecimento preliminar sobre o domínio segue as mesmas alternativas dos exemplos, em virtude dele ser adquirido ao mesmo tempo que os exemplos.

5.3.2 O processo de entrevista

As fases do processo de eliciação de exemplos e conhecimento preliminar são descritas na figura 5.4. A seguir faremos algumas observações sobre essas fases (de 1 a 5).

1) A informação sobre o domínio (médico, fiscal, educacional, etc) e a tarefa a se realizar (classificação, interpretação, diagnóstico, etc), indicam a forma de interagir do sistema, visto que diferentes terminologias podem ser usadas em diferentes tipos de problemas.

2) Os elementos de classificação que compõem os exemplos devem ser logo apresentados pelo usuário, pois auxiliam o processo de entrevista e da obtenção de conhecimento preliminar.

3) O *entrevistador* passa a solicitar casos que concluam os elementos de classificação apresentados. Ele poderá sugerir características dos casos a serem apresentados, solicitando, por exemplo, casos que se destaquem por frequencia (alta

ou baixa) de ocorrência, para identificar tanto os atributos mais comuns, como os mais raros. O auxílio do *entrevistador* pode levar à solicitação de exemplos que representem pares atributo/valor ainda não presentes na tabela de exemplos. Devido à boa representatividade que possuem, os exemplos selecionados podem levar à formação de conceitos mais acurados e que podem classificar todo o contingente da tabela de exemplos mais rapidamente.

A aquisição de relevância semântica e de custo dos atributos é realizada nesta fase, juntamente com a identificação dos atributos e valores que compõem um exemplo. Isso é feito com entrevistas focadas, nas quais as perguntas não exigem um processo grande de raciocínio e as respostas não são muito amplas.

A coluna 2, referente a classe estruturador e classe trata ruído, cita os objetos destas classes, os quais se ativam quando necessário (*demons*), podendo sugerir modificações na estrutura do exemplo. O usuário pode definir um número mínimo de exemplos incluídos, que os objetos dessas classes devem observar antes de se ativarem. Por exemplo, somente a partir de 10 exemplos incluídos é que os objetos podem atuar.

4) Na fase exclusiva de alimentação dos exemplos, são informados todos os casos que devem compor a tabela de exemplos. Nessa fase, se aparecerem novos atributos, o tratamento dado é em função da alternativa de eliciação escolhida. Se houve uma fase de estruturação do exemplo, a inclusão de novos atributos é tratada como uma exceção, podendo-se reformatar a estrutura da tabela de exemplos.

5) Ao se analisar a tabela final de exemplos, independentemente da forma pela qual o processo de geração e digitação da mesma tenha sido realizado, todos os objetos da classe modela domínio são ativados para uma análise final. No A4, a tarefa de discretização é feita após uma análise da distribuição dos valores dos atributos na tabela de exemplos. Esta análise é feita, a partir do cálculo de intervalos de confiança para os valores do atributo. O apêndice C descreve este processo. A discretização e a determinação da escala de valores resultantes, é feita interativamente com o engenheiro do conhecimento e/ou especialista. Também o tratamento de "buracos" é feito nesta fase. Como *default*, o A4 utiliza a estratégia de preencher os "buracos" com o valor que mais ocorreu na tabela de exemplos para o atributo correspondente [Clark 89].

Ainda nesta fase, é eliciado o conhecimento preliminar que não foi totalmente informado. Para aquisição de níveis hierárquicos superiores de atributos, utiliza-se uma simulação do método de *laddering*. Com isso descobre-se relações de

superordenações e de subordinações em relação aos atributos que compõem a estrutura do exemplo.

Também na análise final, o A4 faz um cruzamento entre as relevâncias semânticas e a tabela de exemplo informada. Neste cruzamento pode-se identificar valores de atributos ou até mesmo atributos que são irrelevantes para o problema. Como resultado dessa atividade, uma lista de atributos que não foram considerados relevantes para nenhuma elemento classificador é mostrada ao usuário para que ele proceda com as correções necessárias.

CAPÍTULO 6

ASPECTOS DE DESENVOLVIMENTO DO A4

Neste capítulo consideraremos os aspectos de implementação que nortearam o desenvolvimento do A4. Primeiramente, teceremos algumas considerações sobre a filosofia de orientação a objetos e de que características o A4 possui em função de seu desenvolvimento ter se baseado neste paradigma. Posteriormente descreveremos, em linhas gerais, o processo de desenvolvimento do A4 e a metodologia de projeto utilizada.

6.1 Vantagens de Seguir a Filosofia de Orientação a Objetos

O fato de seguir o paradigma de orientação a objeto acrescentou ao A4 as seguintes funcionalidades:

- Fornecer flexibilidade aos usuários do ambiente para abrigar diversos tipos de algoritmos generalizadores e consequentemente realizar avaliações comparativas entre os mesmos.
- Prover um ambiente que permita fácil extensão pela inclusão de novas ferramentas.
- Prover reusabilidade de procedimentos fornecidos *ad hoc* pelas classes primitivas do ambiente.
- Possibilitar que os objetos do ambiente atuem ortogonalmente em entradas e saídas independentemente a quais algoritmos se refiram.
- Prover uma interface interativa e amigável, orientada a eventos.

Além dessas funcionalidades, a utilização do paradigma de orientação a objetos possibilitou que a solução encontrada para o problema refletisse mais fielmente o mundo real[Vasco 93c].

6.2 Metodologia de Desenvolvimento

O objetivo básico de uma metodologia é organizar o trabalho de desenvolvimento de um sistema computadorizado, a fim de se compreender melhor os elementos envolvidos neste sistema. Numa metodologia são definidas etapas ou passos a serem seguidos, bem como os resultados que em cada passo deverão ser obtidos. Estes resultados normalmente são em forma de diagramas que auxiliam a visualização e compreensão dos objetos envolvidos tanto no problema como na solução a ser adotada.

Nesta seção descrevemos os passos que foram utilizados para o desenvolvimento do A4 e os resultados obtidos em cada passo. Mas antes desses passos serem abordados, é importante citar três características típicas do desenvolvimento orientado a objetos e que influenciaram fortemente o desenvolvimento do A4 :

- Há um forte apelo à prototipação no processo de desenvolvimento.
- O problema é modelado em objetos e não mais em procedimentos e/ou estrutura de dados;
- Somente uma mesma abstração é utilizada na modelagem do mundo real durante todo o transcorrer do desenvolvimento: objetos.

Essas duas últimas características são bem exemplificadas pela diagramação da metodologia utilizada no A4. Pois nessa diagramação os objetos do domínio são inicialmente obtidos em uma forma mais genérica e a partir de sucessivos refinamentos são utilizados até a fase de implementação.

O uso de prototipação foi extremamente facilitado em função das características de reusabilidade e modularidade inerentes a filosofia de orientação a objetos. No A4, um exemplo em que a criação de protótipos foi auxiliada pela utilização de código já existente é o desenvolvimento das suas rotinas de interface, desenvolvidas a partir de classes pré-definidas do sistema XView da SUN microsystems[SUN 91].

A metodologia utilizada foi baseada na de Furtado [Furtado 93] e na de Booch [Booch 91] e possui as seguintes etapas:

- Identificação de classes e objetos
- Identificação da semântica de classes e objetos

- Identificação dos relacionamentos entre classes e entre objetos
- Implementação das classes

Duas observações se fazem necessárias sobres os passos acima citados: a sua execução não é realizada de uma forma estritamente sequencial e sempre que possível, em cada fase, é incentivada a realização de protótipos.

6.2.1 Identificação de classes e objetos

Esta fase foi centrada na compreensão do problema a ser abordado e numa primeira identificação dos objetos que compõem o domínio da aplicação. Para identificar esses objetos, buscamos uma familiarização com o vocabulário do domínio. Nesse processo de familiarização, descobrimos coisas tangíveis, como algoritmos indutivos, funções desempenhadas por pessoas, como modelagem do domínio, propriedades mensuráveis, como custo de um atributo, etc.

No desenvolvimento do A4, essa fase teve uma grande duração em função da inexistência de um sistema bem definido de aquisição de conhecimento, que já utilizasse integradamente todos as ferramentas necessárias para uma perfeita realização da tarefa de AC automática. Ou seja, não era somente uma tarefa de automatizar um sistema manual já existente, mas também de criação de uma rotina que agregasse as diversas funções identificadas no domínio.

Também nessa fase, definimos os principais objetos que compõem o domínio. Partimos da premissa que objetos representam entidades que existem no tempo e no espaço e pudemos caracterizá-los pela forma de sua descoberta, ou seja:

Coisas tangíveis: algoritmo indutivo, exemplos, relevâncias semânticas.

Funções desempenhadas: modelagem do domínio, tratamento de saída.

Abstrações: hierarquias, árvores de decisão, grafos de conhecimento, regras.

Propriedades mensuráveis: custo de um atributo, grau de pertinência de uma relevância.

Propriedades operacionais: Ajudas(*help*), interface.

A partir dos objetos já conhecidos, partimos para uma identificação das suas semelhanças de estrutura e comportamento. Com essa análise, pudemos classificá-los em classes. A classe de um objeto representa uma abstração onde se busca

caracterizar propriedades comuns entre um grupo de objetos. A criação da classe de objetos *conhecimento preliminar* é um exemplo de como essa análise de semelhanças foi realizada no A4. Esse é um caso típico de generalização, no qual podemos dizer que *custo, relevâncias e hierarquias* são **um tipo de** *conhecimento preliminar.*

Como resultado dessa fase, obtivemos um esboço do diagrama de classes, onde visualisamos quais as classes mais importantes do problema. Além disso, fizemos uma primeira definição do quadro de características de cada classe, conhecido como *template*. Nesse quadro definimos as principais informações de cada classe como: hierarquia, atributos, métodos, cardinalidade e persistência. A figura 6.1 mostra o *template* da classe *exemplo*. Esses templates são refinados a medida que surgem novos métodos, novas características e que são determinados novos relacionamentos entre as classes.

TABELA DE EXEMPLOS	
Comentário	Conjunto de pares atributo/valor que concluem um elento classificado
Superclasse	Entrada Tratada
Cardinalidade	n
Uso	Entrevistador
Atributos	Atributo, Classe, Valor
Operações	QuantidadeExemplo, IncluExemplo RemoveExemplo QtdeAtributos
Persistência	Sim
Concorrência	Não

Figura 6.1 Template da classe EXEMPLO

6.2.2 Identificação da semântica de classes e objetos

A identificação da semântica das classes é realizada com a organização de sua hierarquia. Fizemos essa organização através de generalizações diante das semelhanças encontradas ou especificações a partir de novas classes que surgem. Esta tarefa, na verdade, é uma continuação da tarefa de análise de semelhanças feita anteriormente. O objetivo é se obter diagramas para cada classe que representem sua estrutura hierárquica. A figura 6.2, mostra o diagrama da classe *modela domínio*. Os números dentro dos ícones significam a

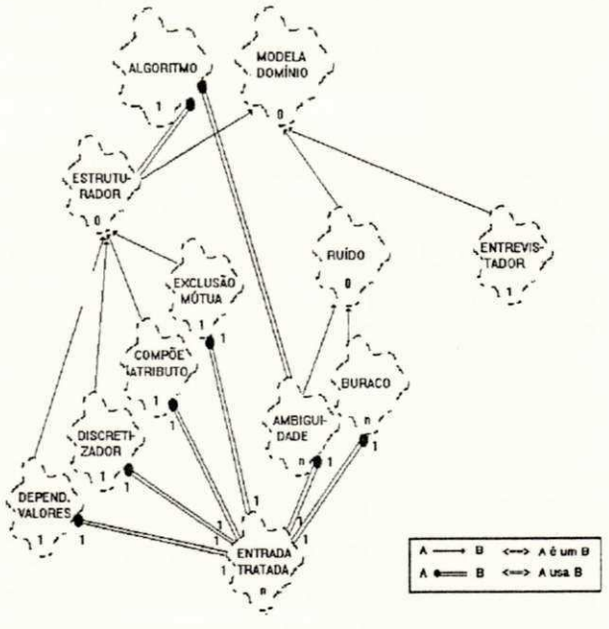


Figura 6.2 Diagrama da classe MODELA DOMÍNIO

quantidade de instâncias possíveis para a classe. Classes com o número zero(não possuem instâncias) são conhecidas como classes abstratas.

Com relação aos objetos procuramos identificar *quem faz o que*. Os objetos anteriormente identificados foram criteriosamente examinados, para a identificação dos seus estados e das ações que modificam esses estados. O estado de um objeto é um conjunto de todas as possíveis situações, presentes ou futuras, que podem ser vivenciadas por ele. A partir das ações que atuam sobre um objeto, identificamos os eventos que ocorrem no problema e que disparam tais ações. Eventos devem ser identificados e controlados, principalmente em se tratando de programação baseada em eventos externos, como sistemas baseados no X windows, porque são eles que refletem as ações tomadas pelos usuários do sistema.

Embora não haja uma fronteira bem definida entre a análise do problema e a proposição da solução, normalmente a partir desta fase o projetista começa a pensar na solução do problema. Isso acontece porque, no momento da distribuição de métodos nas classes apropriadas, surgem objetos que não existem no problema original, mas que são fundamentais para a solução do problema. Um exemplo dessa situação no A4, foi o aparecimento do objeto *gerente.* Identificamos esse objeto quando percebemos a necessidade de *alguém* que possuísse informações sobre características de um determinado algoritmo e que, a partir dessas informações, pudesse ativar os objetos responsáveis por um tratamento adequado.

Com a descoberta dos objetos que surgiram exclusivamente em função da solução, pudemos esboçar o funcionamento do sistema ou seja, a solução do problema. Neste momento, fizemos a definição da arquitetura básica do A4 (vista no capítulo 4 e ilustrada pela figura 4.2) e também projetamos os passos do processo de entrevista que auxilia a modelagem do domínio (vistos em detalhe no capítulo 5).

Como resultado desta fase, pudemos obter diagramas das classes mais refinados. A atualização dos *templates* das classes com o preenchimento de informações referentes a hierarquias, ações realizadas e atributos das classes também foi realizada. Por fim, obtivemos uma descrição da solução do problema com o esboço da arquitetura do A4.

6.2.3 Identificação dos relacionamentos entre classes e entre objetos

Nesta fase descobrimos como as *coisas* interagiam dentro do sistema e estabelecemos os seus relacionamentos. É a fase de descobrir *quem pede o que a quem.* Na verdade a identificação do que se pede, de uma certa forma, já foi realizada no momento em que ações dos objetos foram analisadas. Por isso nessa etapa

analisamos principalmente o destino das mensagens dos objetos. O que não significa que neste momento não pudessemos descobrir novos métodos para os objetos, coisa que aliás, ocorreu frequentemente(lembrar sempre que os passos não são obrigatoriamente sequenciais).

Fizemos a análise dos relacionamentos de uma forma *top down*, com a construção de um diagrama *top level* que mostrou os relacionamentos das principais classes do domínio, em um alto nível de abstração. No A4 especificamente, essa fase foi fundamental para que se consolidasse a arquitetura do ambiente. A figura 6.3 mostra o diagrama de *top level* que obtivemos para o domínio de aquisição automática de conhecimento.

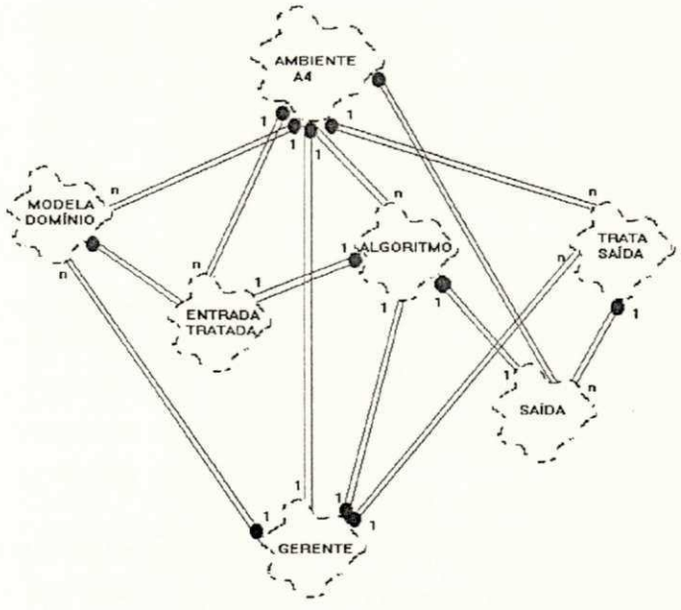


Figura 6.3 Diagrama Top Level do Ambiente A4

A partir dos diagramas individuais de cada classe e de seus *templates* partimos para uma análise do comportamento dos objetos. Nessa análise, identificamos as mensagens que cada objeto enviava e recebia e para quem ele as fazia. O resultado dessa análise foi o diagrama de objetos. O diagrama do objeto *algoritmo* é um exemplo desses diagramas e pode ser visto na figura 6.4.

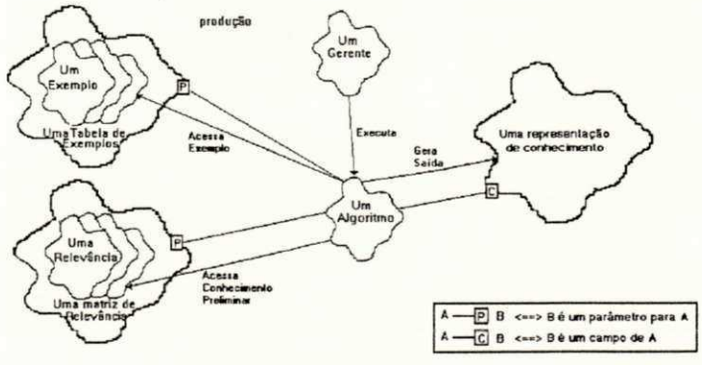


Figura 6.4 Diagrama do objeto ALGORITMO

6.2.4 Implementação das classes

Neste ponto tivemos a *visão de dentro* de cada classe, para decidir como ela seria implementada. Esta fase consistiu em transferir a visão conceitual dos objetos para dentro das estruturas de dados usadas na implementação e definir os algoritmos para os métodos.

As considerações de implementação levaram a decisões como a de transformar uma classe em métodos de uma outra de maior nível de abstração ou de agrupar duas classes em uma.

Neste momento, também foi de suma importância reconhecer todas as possibilidades de utilização de classes já existentes e de como elas seriam anexadas ao sistema. Esse reconhecimento se mostrou importante sobretudo pelo fato da utilização das classe primitivas do sistema Xview. A reutilização das classes do sistema Xview e a utilização do gerador de interface que ele dispõe (Open Development Windows Guide) tiveram um papel essencial no desenvolvimento do A4, principalmente pela forte exigência de protótipos embutida na metodologia.

O escolha do Xview como plataforma induziu a escolha da linguagem C++ como ferramenta para implementar o A4, pois os mesmos se mostram plenamente compativeis entre si. Além disso o C++ apresenta as vantagens de portabilidade e eficiência advindas da linguagem C.

CAPÍTULO 7

Aspectos de Implementação e Uso do A4

Atualmente é plenamente aceito que a utilização dos recursos gráficos oferecidos pelas estações de trabalho aumentam a produtividade na utilização do computador. Em aquisição de conhecimento e especificamente no A4, essa utilização faz-se ainda mais importante, pois as interfaces gráficas tornam a comunicação entre o usuário e computador mais rica e poderosa.

A interface gráfica para comunicação com o usuário utilizada no A4 é baseada no sistema *X window*. O sistema *X window* é o coração de um sistema de janelas e que está sendo atualmente adotado como padrão *de facto* de comunicação homem-máquina através de interface gráfica.

A seguir faremos uma breve definição dos conceitos que envolvem o contexto da comunicação gráfica com o usuário. Com isso visamos facilitar a compreensão dos componentes de interface do A4, os quais também descreveremos neste capítulo.

7.1 Interface Gráfica (GUI – Grafical User Interface)

As GUIs utilizam símbolos gráficos para representar objetos em vídeos mapeados a bits (*bit mapped*). Cada objeto possui aparência e comportamento distintos. Janelas, menus, ícones, quadros de diálogo são exemplos de alguns destes objetos.

Este tipo de interface permite que o usuário execute aplicações, selecione comandos e realize outras tarefas através da interação via teclado ou *mouse* com o computador.

Em relação as interfaces não gráficas, uma GUI possui as seguintes vantagens:

- Facilidade de memorização e reconhecimento dos símbolos gráficos.
- Interações com feedback visual instantâneo, o que reduz o tempo de aprendizagem e fornece uma sensação de controle sobre o computador.

- Padronização nos controles e operações de cada objeto.

Um conceito importante quando se fala em GUI é o de *look and feel* da interface. O *look* é sua aparência física e o *feel* são as características do seu mecanismo de interação com o usuário. Um *look and feel* atraente na interface de um produto pode ser tão importante para o seu sucesso quanto a sua funcionalidade.

7.2 O Sistema X Window

O sistema *X Window* é uma camada, construída sobre o sistema operacional, que oferece serviços básicos para a criação e o gerenciamento de GUIs.

As principais características do X Window são:

- Tansparência por esconder diferenças de arquitetura, sistemas operacionais e protocolos de rede.
- Portabilidade por diminuir as dependências de características de hardware.
- Extensibilidade por possuir um mecanismo específico para incorporar novas funções.

Bibliotecas do X WINDOW

Para desenvolver aplicações no nível do X Window, o programador dispõe de dois níveis de abstração. O primeiro, de mais baixo nível, é oferecido pela biblioteca *Xlib* e o segundo conhecido como biblioteca *Xtoolkit*. A *Xlib* implementa funções que executam operações básicas em vídeos tais como: criar, abrir, mover e destruir uma janela. A *Xtoolkit* é uma biblioteca de mais alto nível que define rotinas com as quais o programador pode criar e gerenciar os objetos das GUIs, simplificando o esforço de programação.

7.3 O XView

O XView é o *Xtoolkit* da SUN para o sistema *X Window*. O *look and feel* adotado pelo XView é o padrão Open Look desenvolvido pela Sun e AT&T . O Xview é baseado na *Xlib* do sistema *X Window* e é formado basicamente por janelas e *widgets*. As janelas

são áreas retangulares no vídeo, como se fossem telas em miniaturas. Estas janelas são basicamente de quatro tipos : janelas base, de texto, de desenho e *popup*. O manuseio dessas janelas pode ser realizado por três formas distintas :

- Através de menus como o da figura 7.1 (a).
- Através do acionamento do símbolo do canto esquerdo superior visto na figura 7.1 (b) e que corresponde a opção *close* do menu da figura 7.1.
- Através do acionamento do mouse nos cantos da janela para modificação do seu tamanho.

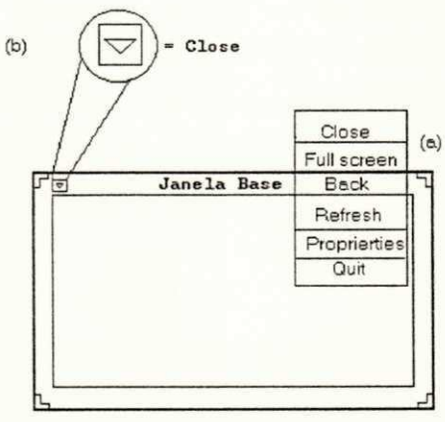


Figura 7.1 Janela base

Os *widgets* são objetos de interface com o usuário tais como menus, botões, barras de *scroll*, listas, *notices* e ícones. A figura 7.2 mostra os *widgets* que são utilizados no A4. Aos *widgets* estão associadas ações que eles devem executar ao serem acionados.

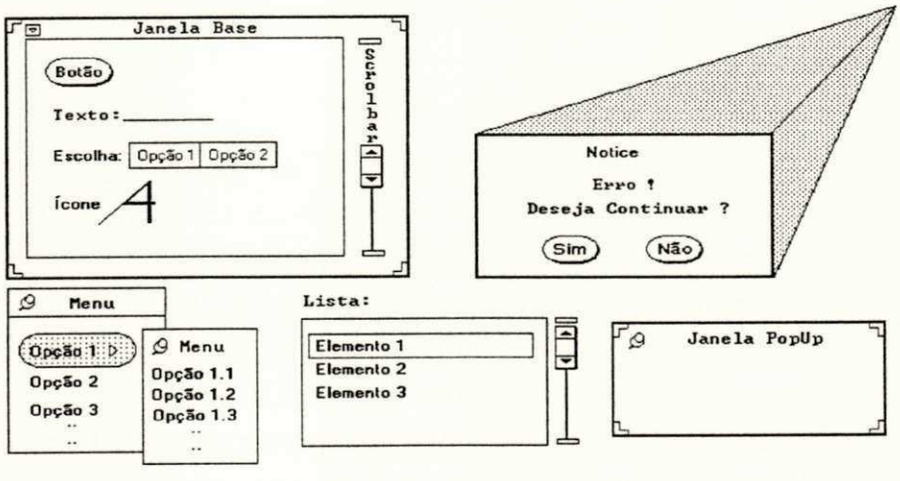


Figura 7.2 Widgets existentes no A4

7.4 A Interface do A4

A janela principal do A4 é vista na figura 7.3. Esta primeira janela apresenta as opções : modelar o domínio, algoritmo indutivo, saída e configurar. As três primeiras opções refletem o domínio do problema que, como foi abordado no capítulo 5, foi dividido nessas três grandes classes. Já a classe de configuração (ícone configurar) refere-se somente a aspectos de implementação do A4.

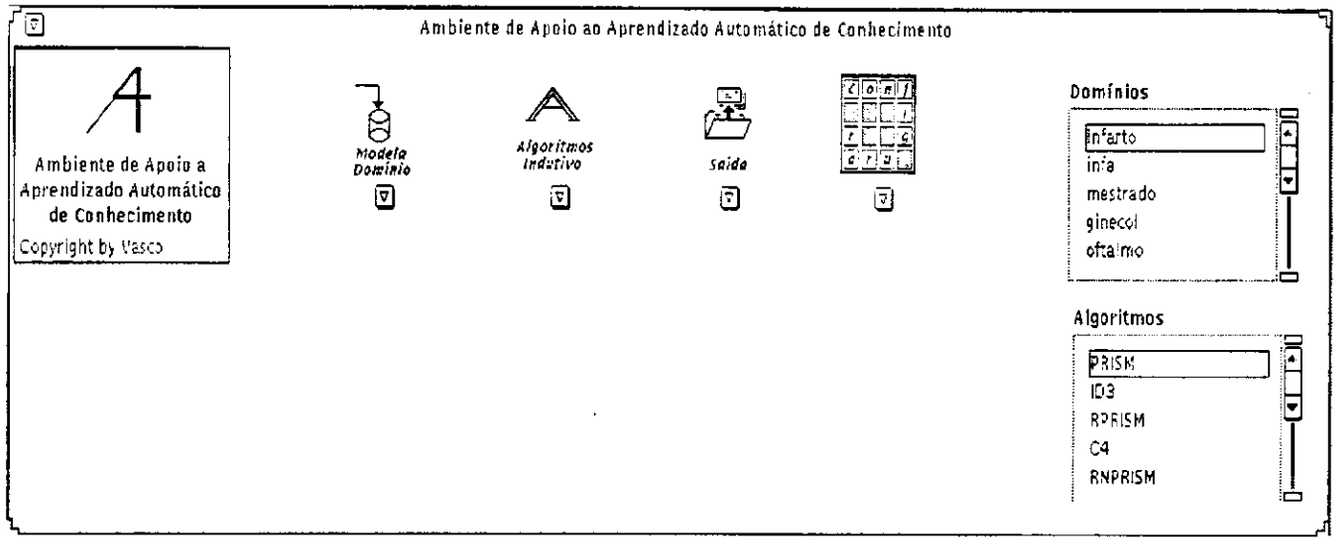


Figura 7.3 Janela principal do A4

A figura 7.3 mostra ainda, em seu lado direito, duas listas: a lista de domínios e a lista de algoritmos. Na lista de domínios encontram-se os nomes dos domínios que foram e podem ser tratados pelo A4. Esses domínios têm o mesmo nome de tabelas de exemplo criadas no A4, mas sem a extensão *.TAB* A lista de domínio auxilia o usuário a escolher os objetos que ele deseja carregar no ambiente, como tabelas de exemplo, hierarquias, relevâncias ou custos. Na lista de Algoritmos estão os algoritmos que fazem parte do A4 e que consequentemente podem ser manuseados pelo usuário (ver tela de tratamento de algoritmos em 7.4.2). Da mesma forma que a lista de domínios, a lista de algoritmos auxilia o manuseio de algoritmos, no momento de selecioná-los.

O A4 possui um sistema de ajuda (*help*) para todos os widgets que formam o ambiente. Para solicitar uma ajuda, o usuário deve posicionar o cursor ou o *mouse* sobre o objeto desejado e teclar F1. A figura 7.4 é um exemplo da janela de *help* do A4.

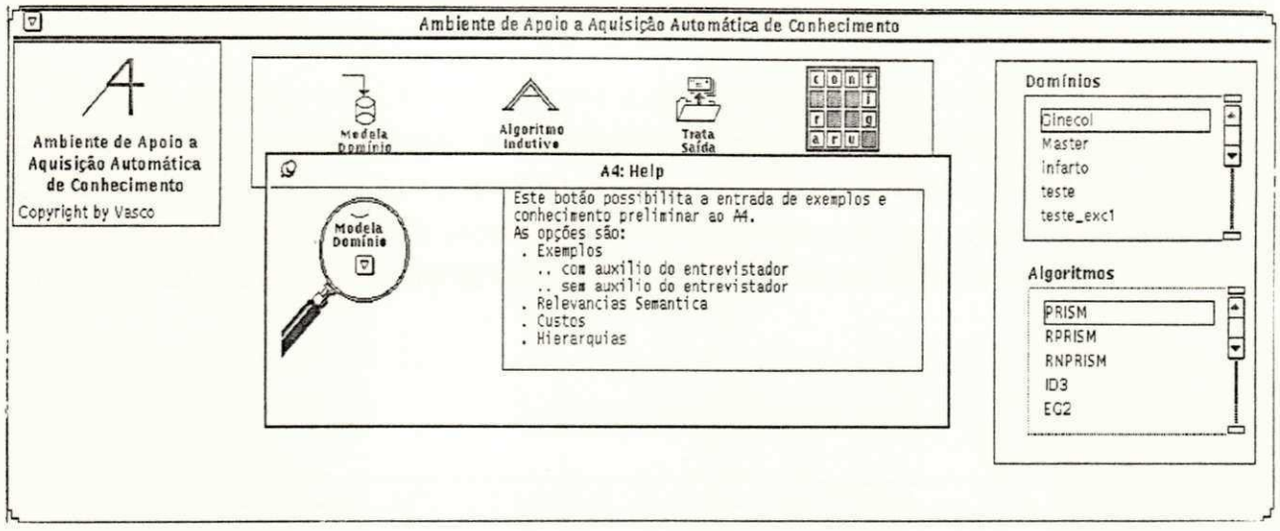


Figura 7.4 Exemplo de help no A4

A saída do A4 se dá através do abandono da janela principal, conforme já exposto em 7.3 e mostrado na figura 7.2.

Vamos examinar mais detalhadamente cada um dos objetos da janela principal.

7.4.1 Modela Domínio

A opção modela domínio deve ser acionada quando se deseja modelar o domínio a ser trabalhado em exemplos, relevâncias, custos ou hierarquias. O ícone *modela domínio*, ao ser acionado, tem o comportamento de mostrar o menu visto na Figura 7.5.

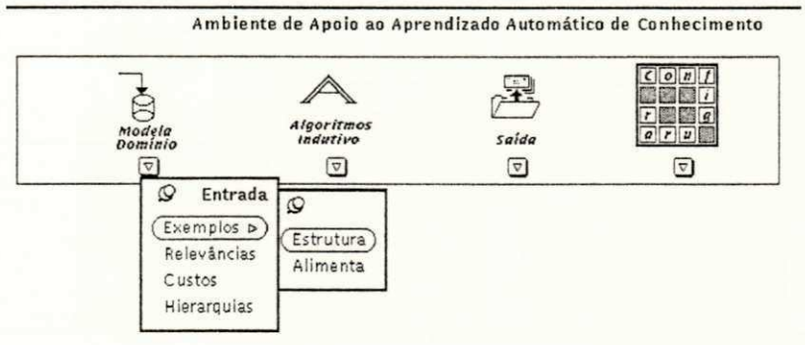


Figura 7.5 Em que modelar o domínio

7.4.1.1 Exemplos

A modelagem do domínio em exemplos pode ser feita de duas formas, como mostra o submenu na figura 7.5: com estruturação ou somente alimentação de informações. A opção com estruturação deve ser escolhida quando é necessário fazer uma definição da estrutura básica do exemplo. A opção de alimentação deve ser escolhida quando o domínio já

estiver bem estruturado e for necessário somente fazer a digitação dos casos.

Na opção com estruturação o usuário terá auxílio do entrevistador que aconselhará a estruturação e a apresentação dos exemplos. A figura 7.6 mostra o entrevistador em ação no domínio de problemas cardio vasculares. O entrevistador é formado por duas partes: na parte inferior fica registrada a última mensagem por ele enviada, aqui denominada mensagem corrente (em vermelho no vídeo) e na parte superior existe

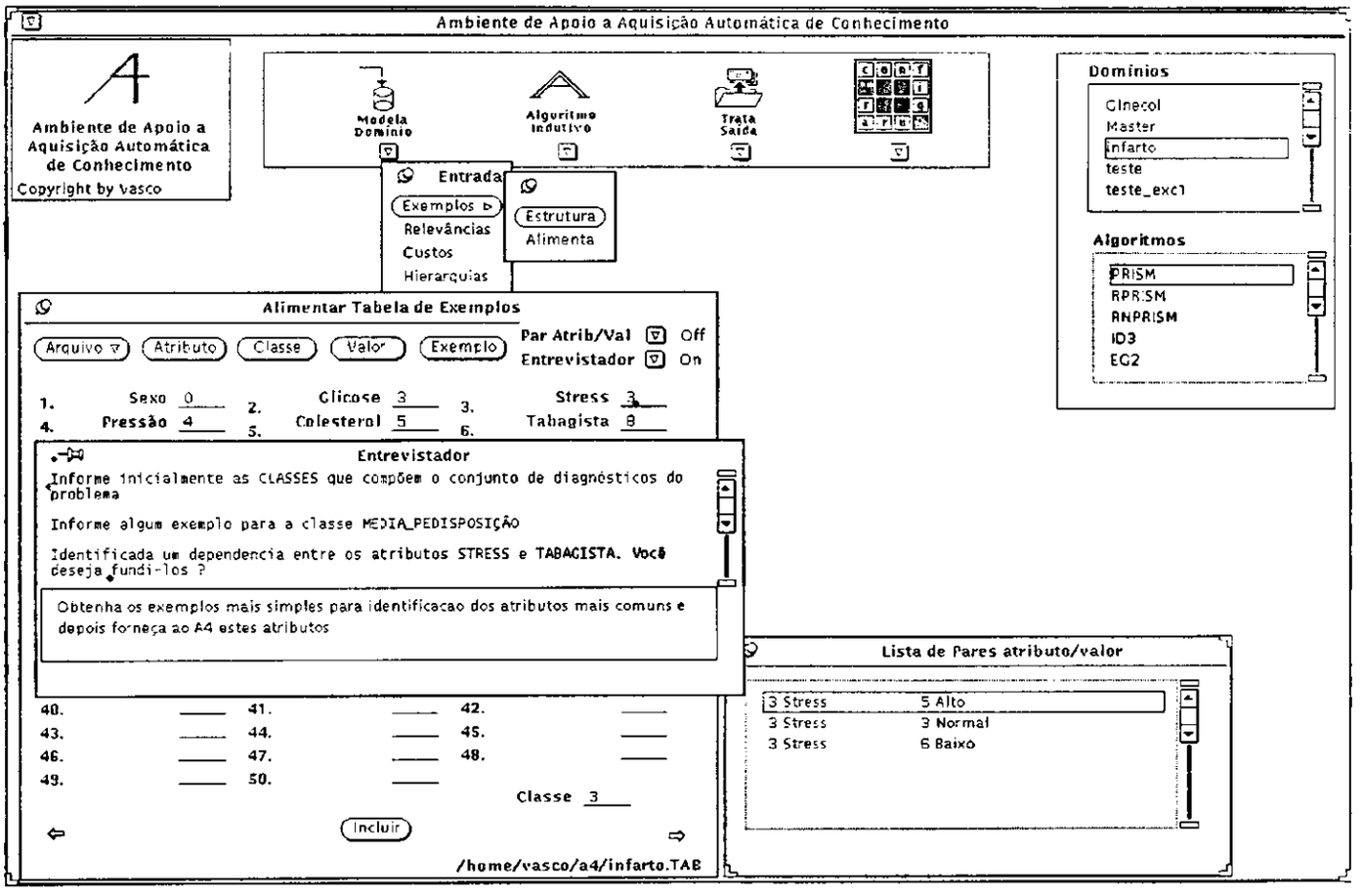


Figura 7.6 O entrevistador em ação

uma lista que guarda todas as mensagens anteriores a mensagem corrente.

Tanto na fase de estruturação como na fase de alimentação o A4 mostra a janela de

alimentação de exemplos. Esta janela é destinada ao recebimento dos casos. Na parte superior da janela de alimentação existem os botões de tratamento de arquivo, atributo, classe e exemplo. Além destes botões há dois botões de escolha que definem se a janela de pares atributo valor deve estar ativa ou não e um outro botão que indica o mesmo quanto ao entrevistador.

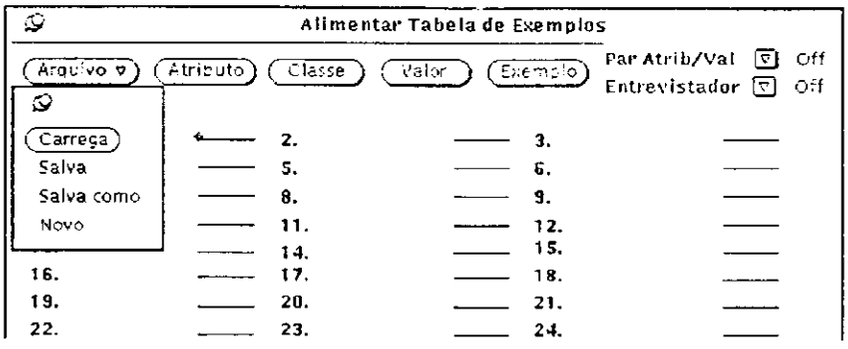


Figura 7.7 Opções de manuseio da tabela de exemplos

O botão **arquivo** deve ser acionado quando se deseja carregar, salvar, salvar como ou limpar a janela de alimentação para a criação de uma nova estrutura de exemplo. Essas quatro opções, compõem o menu subordinado ao ícone arquivo, como visto na figura 7.7, e são descritas a seguir:

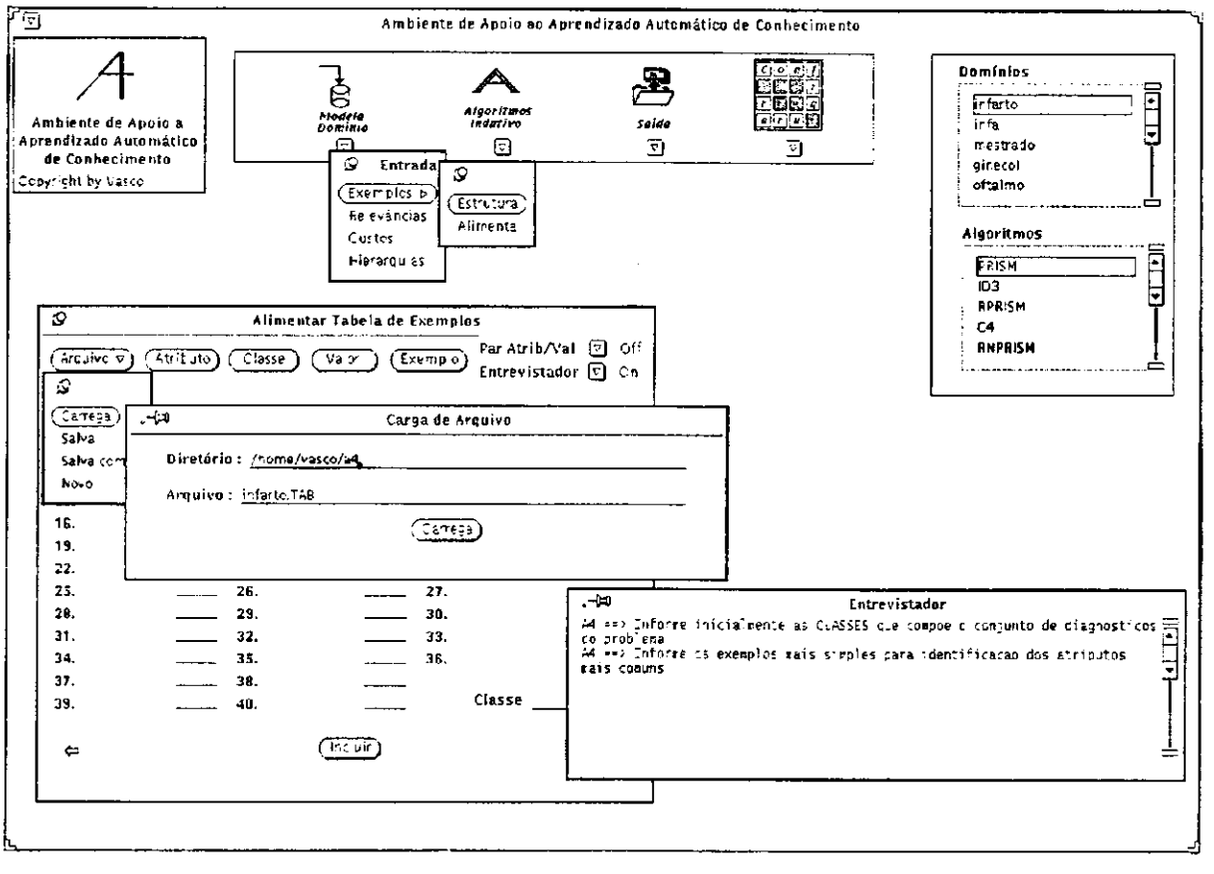


Figura 7.8 Carga de tabela de exemplos

» **Carrega:** Ao se optar por carregar um arquivo (tabela de exemplos) na área da janela de alimentação, o A4 mostra uma janela como a da figura 7.8 que recebe a informação do diretório e do nome da tabela de exemplos. O usuário deve digitar essas informações, sendo que o nome da tabela de exemplos pode ser escolhido através de seleção na lista de domínios a que ele se refere.

» **Salva**: Ao se optar por salvar um arquivo, as informações digitadas são gravadas no arquivo corrente (indicado no canto direito inferior da janela de alimentação). Caso não haja um arquivo corrente, o A4 apontará um erro na ação de salvar. É importante salientar que as informações digitadas na janela de alimentação de exemplos, bem como as manipulações na estrutura do exemplo, só são efetivadas quando se faz uso da opção salvar.

» **Salva Como**: A opção *salva como* se destina a salvar a tabela de exemplos que está na memória com um nome diferente do carregado no momento (se a tabela de exemplos corrente já possuir algum nome, este nome estará indicado no canto direito inferior da janela de alimentação de exemplos). O nome do arquivo a ser salvo é solicitado numa janela semelhante a mostrada na opção *carregar*. Logo após a confirmação do novo nome da tabela de exemplos, a lista de domínios é atualizada. O nome da tabela de exemplos deve obrigatoriamente ser informada com a extensão *.TAB* ao seu final, sob pena do A4 não reconhecer, como tabela de exemplos, um nome que não siga essa regra.

» **Novo**: A opção novo deve ser escolhida quando se deseja criar uma nova estrutura de exemplo, diferente da corrente. Esta opção limpa totalmente a área de trabalho. Em função disso, quando o usuário faz a escolha dessa opção o A4 pede a confirmação através de um *notice.*

O botão **atributo**, ao ser acionado, ativa a janela de tratamento de atributos. Esta opção objetiva facilitar o manuseio dos atributos que vão compor a estrutura do exemplo. A janela de tratamento de atributos, vista na figura 7.9, é composta por uma lista dos atributos que compõem o exemplo, as características do atributo (nome, tipo, nome abreviado e custo) e um conjunto de botões que definem ações a serem tomadas ou outros objetos que se relacionam com os atributos. Os botões da janela de atributo são:

» **Incluir**: Este botão deve ser acionado para incluir um novo atributo na estrutura do exemplo. O acionamento deste botão só deve ser feito, após a digitação do novo nome do atributo no campo de texto *nome do atributo* e do preenchimento das características do atributo. A lista de atributos é

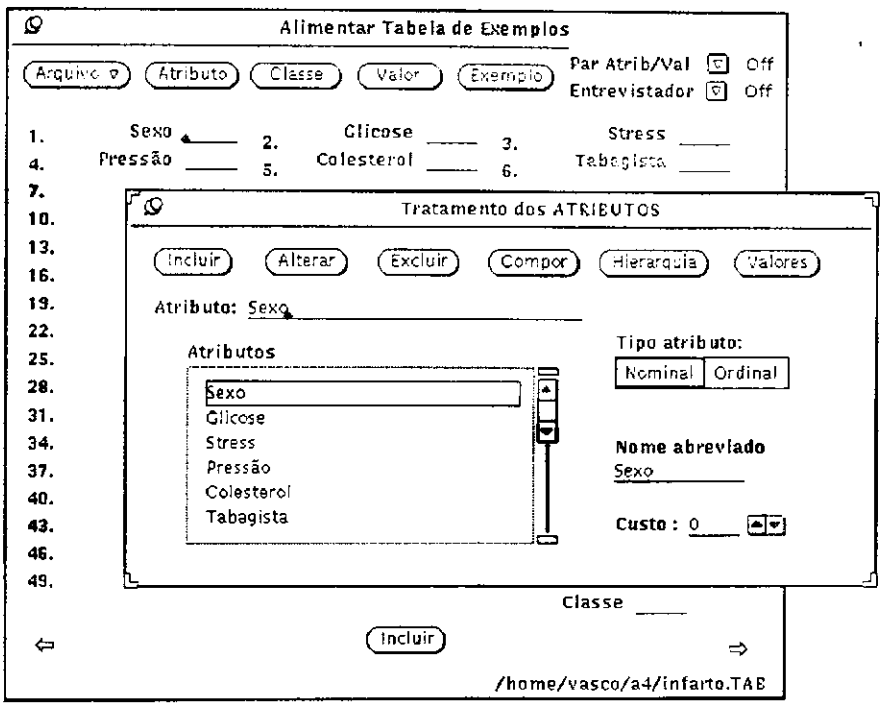


Figura 7.9 Janela de tratamento de atributos

imediatamente atualizada com o nome do novo atributo. O mesmo acontece com os *labels* da janela de alimentação de exemplos.

» **Alterar**: Este botão possibilita a alteração de informações do atributo, ou até mesmo de seu nome. Para isso, basta selecionar o atributo desejado na lista de atributos, realizar a modificação e depois acionar o botão alterar. Caso o nome do atributo selecionado seja modificado o A4 mostrará um *notice* que pedirá a confirmação dessa ação.

» **Excluir:** Este botão deve ser acionado para excluir um atributo da estrutura do exemplo. Para realizar a exclusão deve-se selecionar o atributo desejado na lista de atributos e acionar o botão excluir. É importante se certificar que a ação de excluir é realmente desejada, porque os exemplos digitados com aquele atributos serão modificados. O A4 informa esse fato e solicita a confirmação da ação.

» **Compor:** O botão de compor deve ser acionado quando se deseja fundir dois atributos. Para isso o usuário seleciona o primeiro atributo a ser composto na lista de atributos, e o A4 solicita o segundo (mensagem no canto inferior esquerdo da tela de tratamento de atributos). A figura 7.10 mostra, para um domínio ginecológico, a janela de composição de atributos mutuamente exclusivos que aparece após a informação do segundo atributo a ser composto. Se a composição for confirmada, os *labels* da tela de alimentação e a lista de atributos são automaticamente atualizados. Caso seja acionado o botão cancelar, na tela compor atributos, a situação anterior é recuperada.

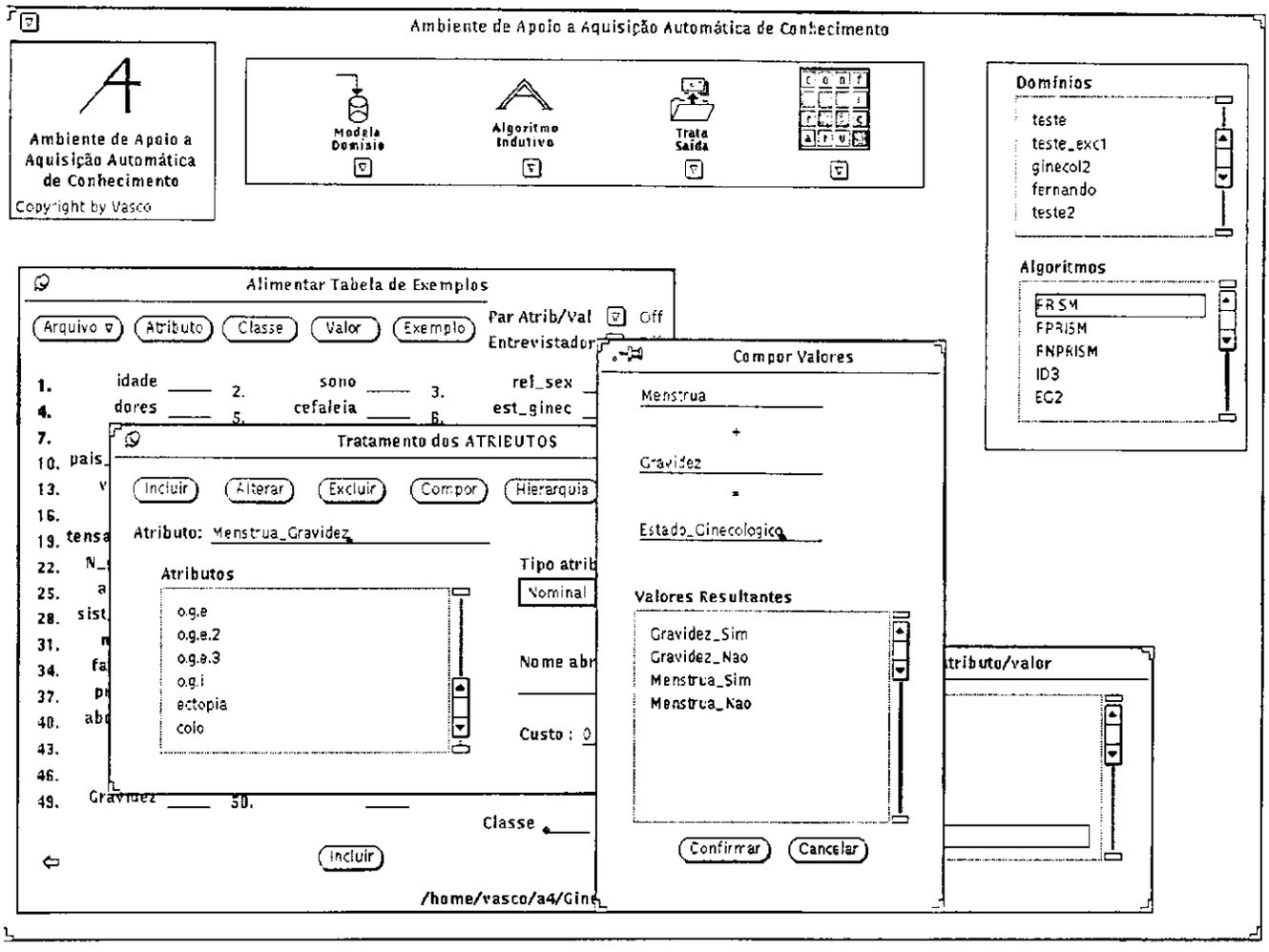


Figura 7.10 Janela de composição de atributos

» **Hierarquia**: Este botão liga a tela de tratamento de atributo à de tratamento de Hierarquias (ver em 7.4.1.4). Essa possibilidade permite ao usuário prover informações sobre hierarquias no exato momento do manuseio do atributo.

» **Valores**: Este botão, ao ser acionado, ativa a tela de tratamento de valores. O nome do atributo corrente é automaticamente mostrado no campo atributo da janela de tratamento de valores, bem como os valores que já estiverem cadastrados para esse atributo.

O botão **classe**, quando acionado, ativa a janela de tratamento de classes. Esta janela auxilia o tratamento das classes ou elementos de classificação do problema. A janela é composta por uma lista de classes, o nome da classe corrente e os botões que definem as ações que podem ser realizadas com os elementos de classificação.

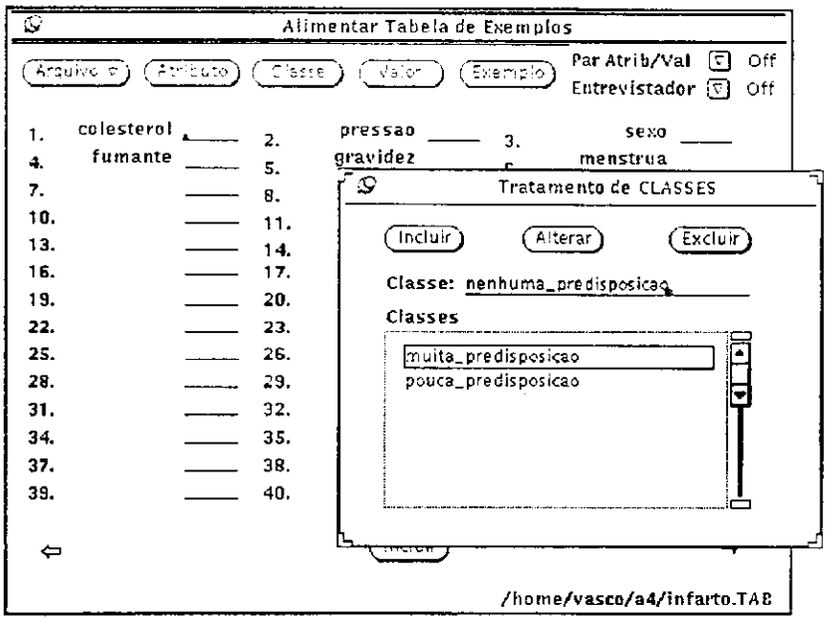


Figura 7.11 Janela de tratamento de classes

Os botões da janela de classe são incluir, alterar e excluir. A figura 7.11 mostra a janela de tratamento de classes.

- **Incluir**: Este botão deve ser acionado para incluir um elemento de classificação(classe). O acionamento deste botão só deve ser feito, após ter sido digitado, no campo de texto *nome da classe*, o novo nome da classe. A lista de classes é imediatamente atualizada com o nome da nova classe.
- **Alterar**: Este botão possibilita a alteração do nome da classe. Para alterar uma classe, deve-se selecionar na lista de classes aquela desejada, realizar a modificação e depois acionar o botão alterar.
- **Excluir:** Este botão deve ser acionado para excluir uma classe para um dado domínio. Para realizar a exclusão deve-se selecionar a classe desejada na lista de classes e acionar o botão excluir. É importante certificar-se se a ação de excluir é realmente desejada porque os exemplos digitados com aquela classe serão modificados. Em função disso, O A4 solicita a confirmação da ação.

O botão **Valor**, possibilita o tratamento de valores de um certo atributo. A janela de tratamento de valor, que pode ser vista na figura 7.12, possui dois campos de texto onde devem ser informado o nome do valor e do atributo a qual ele se refere. A lista de valores da figura 7.12 mostra quais os valores existentes para o atributo informado no campo de texto *atributo*. A janela de tratamento de valor possui três botões, quais sejam:

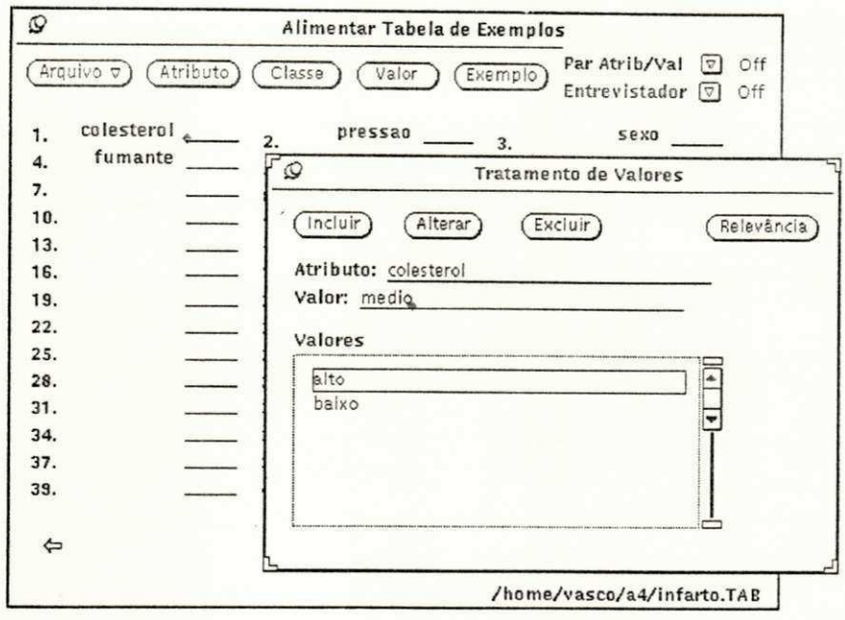


Figura 7.12 Janela de tratamento de valores

- **Incluir**: Este botão deve ser acionado para incluir um novo valor. O acionamento deste botão só deve ser feito, após a digitação do novo nome do valor no campo de texto *nome valor* e do nome do atributo a qual ele se refere no campo de texto *nome atributo*. A lista de valores é imediatamente atualizada com o nome do novo valor.
- **Alterar**: Este botão possibilita a alteração do nome de um valor. Para alterar um valor , deve-se selecionar na lista de valores aquele desejado, realizar a modificação e depois acionar o botão alterar.
- **Excluir:** Este botão deve ser acionado para excluir um valor para um dado atributo. Para realizar a exclusão deve-se selecionar o valor desejado na lista de valores e acionar o botão excluir. É importante certificar-se se a ação de excluir é realmente desejada porque os exemplos digitados com aquele valor serão modificados. Por este motivo, o A4 solicita a confirmação da ação.

O botão **Exemplo**, quando acionado, ativa a tela de tratamento de exemplos. Nessa tela pode-se visualizar a tabela de exemplos de uma forma colunar. Pode-se realizar rolagens tanto verticais como horizontais. A janela de tratamento de exemplos é composta por uma área de texto em que são apresentadas as tabelas de exemplo, e os botões que auxiliam o manuseio de exemplos. Esses botões são utilizados para :

- **Selecionar:** Nesta opção o usuário pode selecionar um subconjunto da tabela de exemplos corrente. Esse sub-conjunto pode ser selecionado em função das classes, pares atributo/valor ou pela numeração dos exemplos.

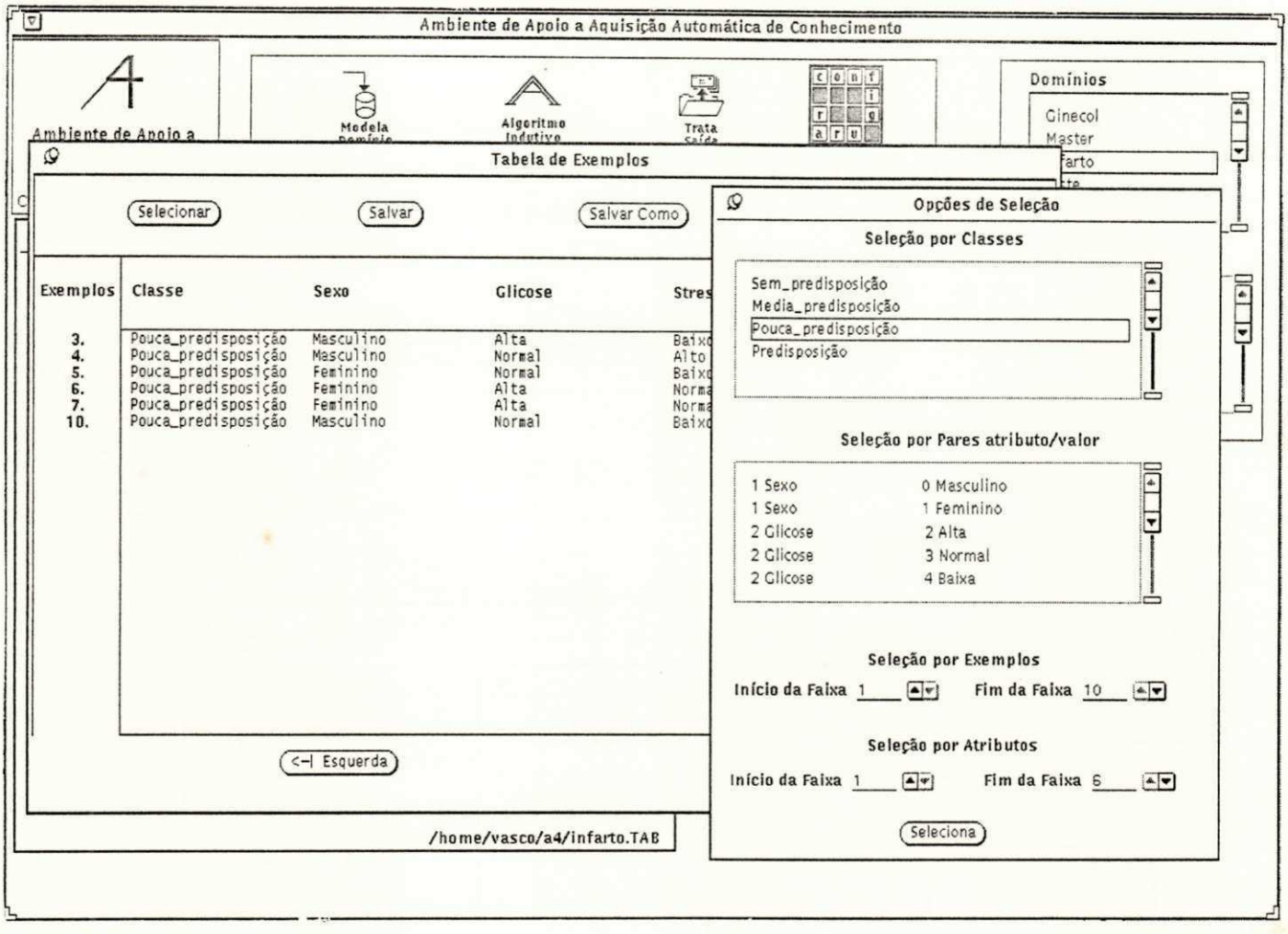


Figura 7.13 Manuseio de uma tabela de exemplos

A figura 7.13 mostra um exemplo, no domínio cardiológico, de uma seleção pela classe Pouca_predisposição. As condições de seleção podem ser escolhidas em conjunto. A opção de seleção pode ser utilizada para se visualizar somente um sub-conjunto dos atributos que formam a tabela de exemplos corrente.

- **Salvar Como**: O botão Salvar Como deve ser escolhido quando se deseja criar uma tabela de exemplos que seja um subconjunto da tabela de exemplos corrente.
- **Salvar:** A opção Salvar guarda o subconjunto selecionado dentro da tabela de exemplos corrente. Esta opção deve ser utilizada com cuidado porque, dependendo da seleção feita, alguns exemplos da tabela de exemplos corrente podem ser perdidos.
- **Excluir**: A opção excluir funciona de forma semelhante a seleção. Há a possibilidade de se excluir exemplos em função de condições

pré-estabelecidas.

7.4.1.2 Relevâncias

Ao se fazer a opção por relevâncias em um determinado domínio, o A4 mostrará o nome das matrizes de relevância para esse domínio. O usuário deve selecionar a matriz desejada e acionar o botão carregar. A matriz selecionada aparecerá na janela de

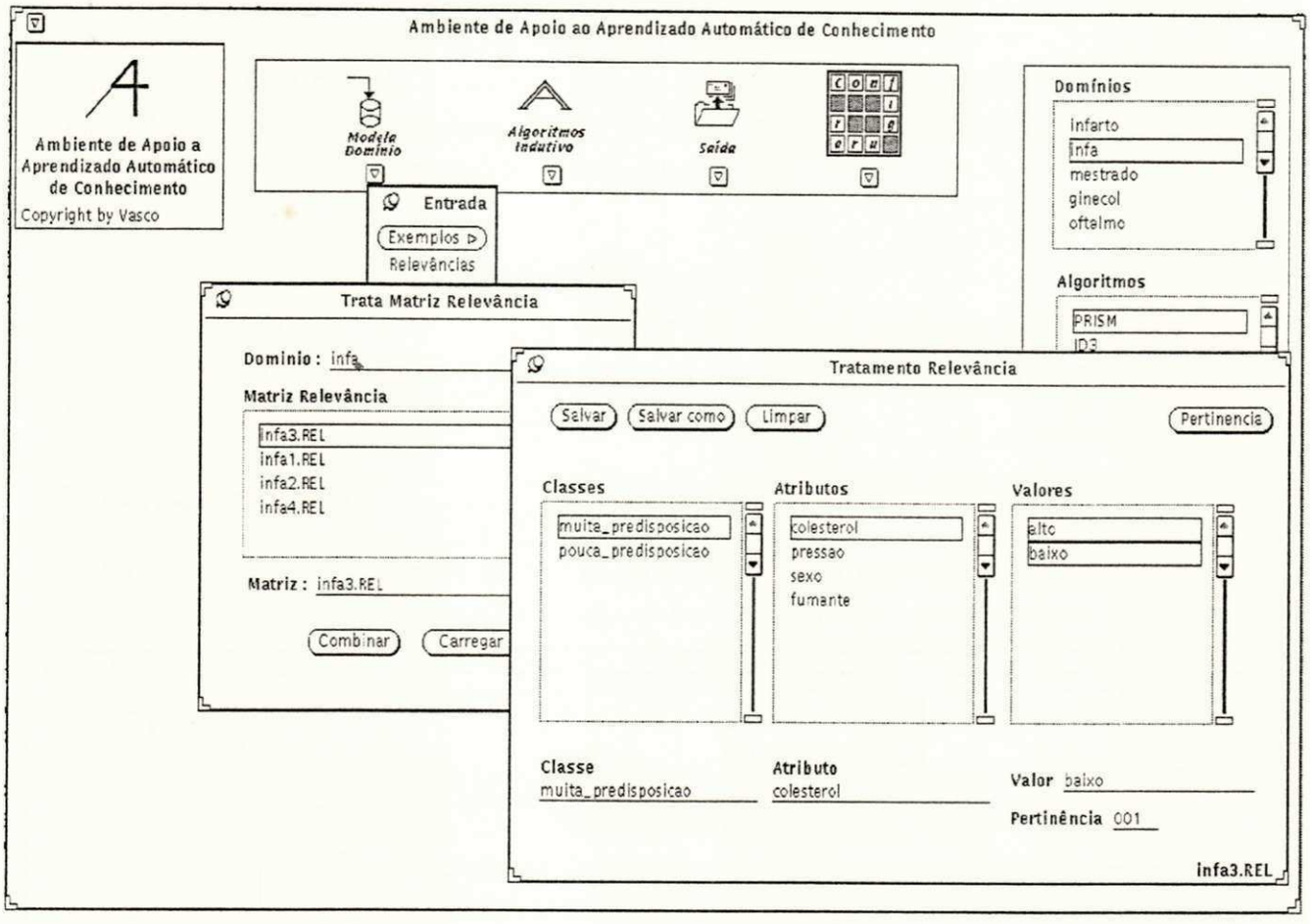


Figura 7.14 Janela de tratamento de relevâncias

tratamento de relevâncias, que contém as listas de classe, atributo e valores do atributo para o domínio selecionado. A figura 7.14 ilustra esse processo.

Para saber se um determinado valor de um atributo é relevante para uma determinada classe, basta selecionar a classe e o atributo desejados em suas listas correspondentes e observar quais valores do atributo estão selecionados na lista de valores (marcados em destaque). Para marcar um valor como relevante, o procedimento é semelhante. Após a seleção da classe e atributo, basta selecionar o

valor desejado na lista de valores.

A janela de tratamento de relevâncias possui ainda a informação de pertinência de uma relevância. Esta pertinência é informada para indicar o grau de incerteza da relevância do valor para uma determinada classe. Para informar o grau de pertinência basta digitar um valor no intervalo [0,1]. Onde 0 significa a ausência de relevância e 1 significa total crença na existência dela.

Os botões, de **salvar** e **salvar como**, existentes na janela de relevância devem ser usados respectivamente para salvar a matriz corrente ou salvá-la com um novo nome. Já o botão **pertinência** deve ser usado quando se deseja modificar o grau de pertinência de uma relevância, bastando para isto, seja digitado o novo grau de pertinência e então acionado o botão de pertinência.

7.4.1.3 Custos

A opção custo no menu do ícone *modela domínio* permite a informação de custos de um determinado atributo. A alimentação do custo de um atributo se dá na janela de tratamento de atributos.

7.4.1.4 Hierarquias

Ao se escolher hierarquias no menu do ícone *modela domínio*, o A4 apresenta os

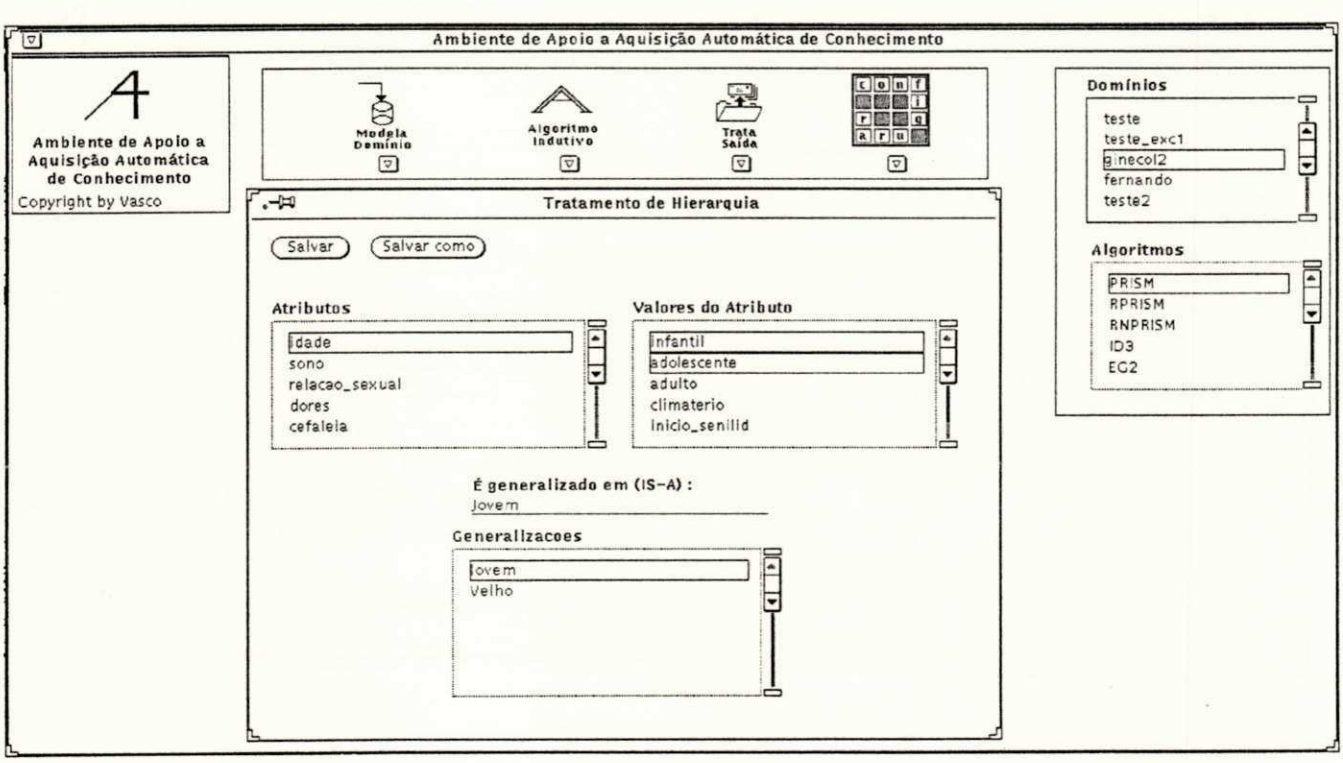


Figura 7.15 Janela de tratamento de hierarquias

nomes das hierarquias existentes para um determinado domínio. Como nas rotinas anteriores, a escolha do domínio pode ser feita com o auxílio da lista de domínios.

Ao se carregar um determinada hierarquia, a janela de tratamento de hierarquias, vista na figura 7.15, é mostrada. Esta janela possui três listas: a lista de atributos, a de valores de um atributo e a de generalizações ou hierarquias. Para consultar a existência de uma generalização deve-se selecionar nas suas respectivas listas o atributo e a generalização desejados, e então os valores do atributo que tiverem sido generalizados aparecem marcados. Para informar a existência de uma generalização deve-se selecionar a generalização e o atributo desejados e marcar na lista de valores aqueles que são generalizados. Caso seja necessário incluir um nome de generalização, deve-se digitar este nome no campo de texto abaixo do label *E generalizado em* e acionar a tecla <*enter*>.

7.4.2 Algoritmo Indutivo

A opção algoritmo indutivo deve ser acionada quando se pretende manusear algoritmos indutivos generalizadores. Após a escolha dessa opção o A4 mostra a janela de tratamento de algoritmos, vista na figura 7.16. Esta janela possui um conjunto de

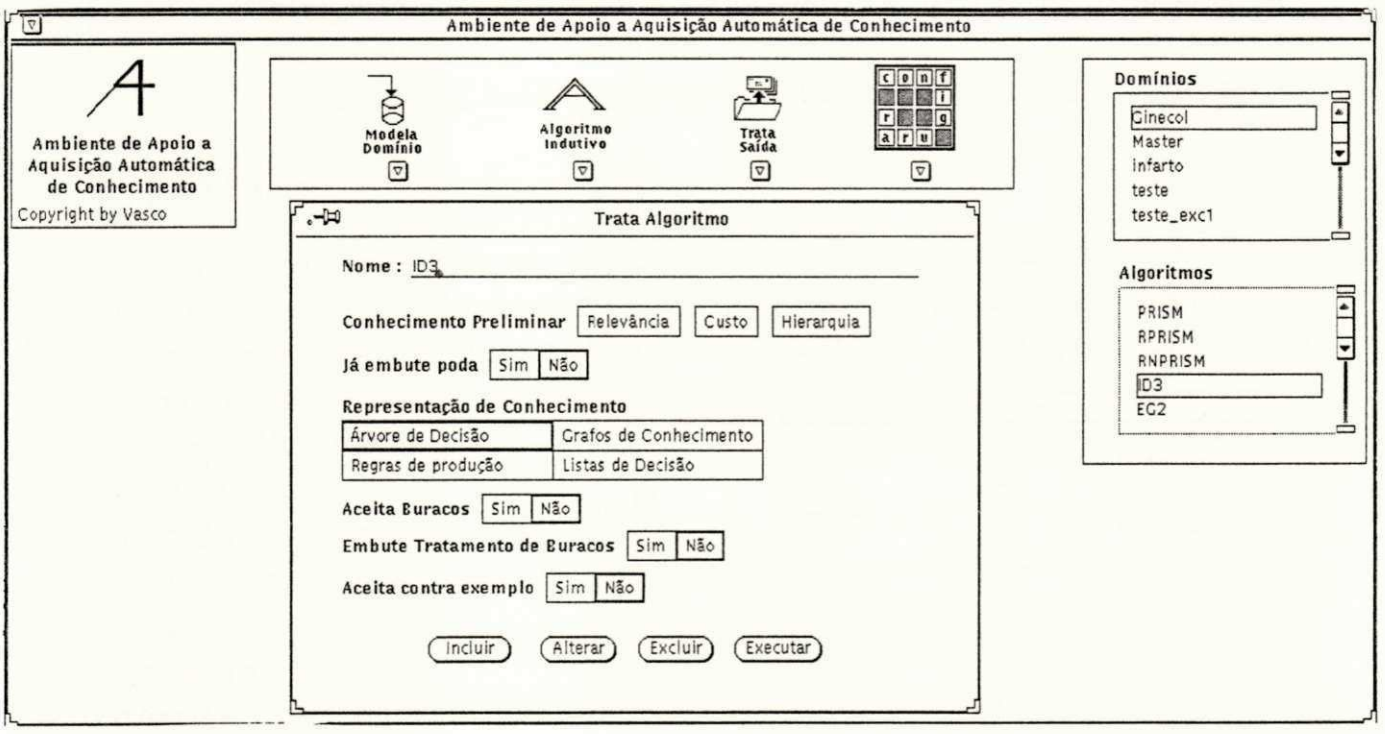


Figura 7.16 Janela de tratamento de algoritmos

campos de texto destinados ao recebimento e visualização das características de um determinado algoritmo. Além desses campos existem quatro botões que definem as ações que podem ser realizadas sobre algoritmos indutivos: incluir, alterar, excluir e executar.

- **Incluir**: Para incluir um algoritmo no A4 é preciso, somente, digitar suas características e acionar o botão de incluir.
- **Alterar:** O botão alterar deve ser acionado caso se tenha feito alguma modificação nas características do algoritmo indicado no campo nome da janela de algoritmos.
- **Excluir:** Ao ser acionado o botão excluir o A4 retira de sua lista de algoritmos o algoritmo indicado no campo texto nome da janela de tratamento de algoritmos.
- **Executar:** O acionamento do botão de execução faz com que o A4 ponha em execução o algoritmo indicado no campo nome da janela de tratamento de algoritmos. Caso este algoritmo não esteja cadastrado no A4, um *notice* informará o erro. O algoritmo selecionado para executar deverá trabalhar com os arquivos correntes do momento de sua execução. Os arquivos correntes estão definidos no arquivo de configuração do A4 e podem ser acessados pela opção **configurar**, a ser vista em 7.4.4. Antes da execução do algoritmo selecionado o A4 faz uma crítica nas entradas que este algoritmo utilizará, podendo com isso enviar, ao usuário, *notices* que alertam , por exemplo,a necessidade de tratar buracos e realizar discretizações.

7.4.3 Saída

Esta opção refere-se aos tratamentos que podem ser dados as saídas dos algoritmos indutivos. Conforme pode ser visto na figura 7.17, menu do ícone *Saída*, estão previstos os seguintes tratamentos: simplificação, validação, estratégias, formatação, combinação e listagem. No estágio atual do A4, somente a opção de listagem está implementada, onde se pode visualisar a saída gerada por um algoritmo generalizador. A

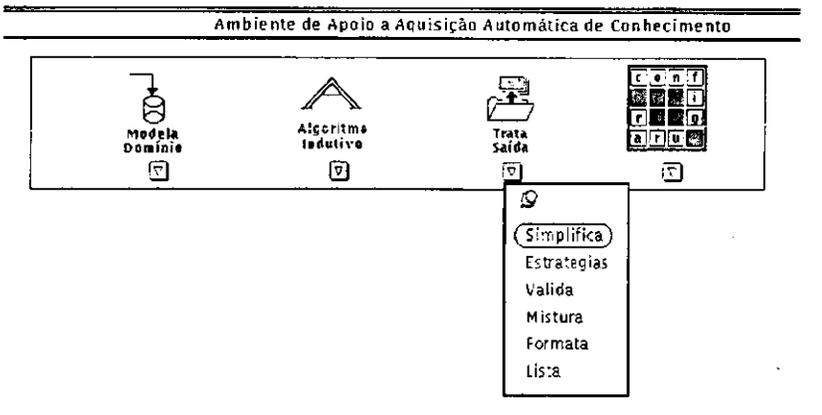


Figura 7.17 Menu de saída

opção listagem mostra uma janela com o conteúdo do arquivo indicado em saída corrente na janela de configurar.

7.4.4 Configurar

A opção de configuração possibilita a edição dos parâmetros que auxiliam o funcionamento do A4. Esses parâmetros são: informações dos arquivos correntes(tabelas de exemplos, hierarquias, relevâncias e saída de algoritmo) e a quantidade de exemplos usada pelo estruturador e entrevistado, para acionar os métodos de auxilio a estruturação e apresentação dos exemplos. A figura 7.18 mostra a janela de configuração com suas informações. A modificação de uma determinada configuração pode ser feita com a digitação das novas informações e em seguida o acionamento do botão de configurar.

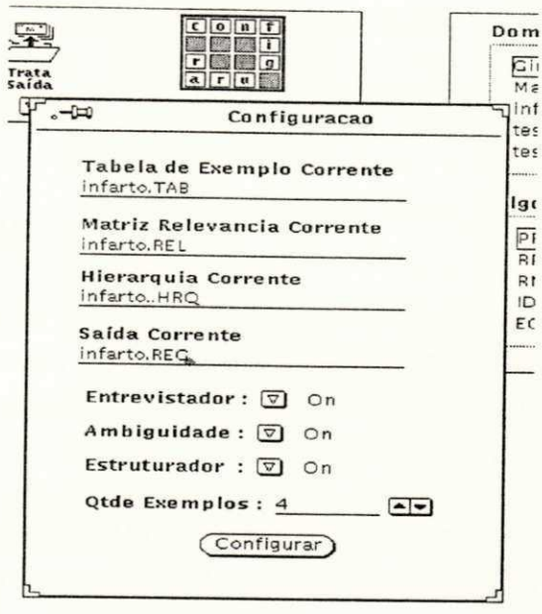


Figura 7.18 Janela de configuração

CAPÍTULO 8

Conclusão

Neste capítulo teceremos as considerações finais, para em seguida descrevermos os principais aspectos relevantes de nosso trabalho e por último, propormos alguns temas de futuros trabalhos que poderão ser desenvolvidos a partir deste.

8.1 Considerações Finais

Embora haja uma enorme proliferação de taxonomias definindo aquisição de conhecimento automatizada e de como ela pode ser dividida, a classificação das técnicas automatizadas em baseadas em entrevistas e baseadas em aprendizado automático é a mais adotada.

O que observamos como alternativa promissora para solucionar os problemas de AC, é a utilização de diferentes técnicas de uma forma integrada. Dessa integração, estão surgindo os ambientes de aquisição de conhecimento que utilizam tanto técnicas de aprendizado automático, nos seus diferentes paradigmas, como técnicas de AC baseadas em entrevistas.

O A4 segue a idéia de integração e busca atenuar as dificuldades que envolvem a tarefa de aquisição automática de conhecimento, mais especificamente a baseada em indução a partir de exemplos. Esta atenuação de dificuldades acontece em função do uso de informações semânticas nos algoritmos generalizadores, suporte a diferentes algoritmos e mecanização de procedimentos de refinamento das entradas. Além disto, o ambiente apresenta facilidades que elevam a eficácia das saídas devido à possibilidade de aplicação de diversos objetos tratadores, que pode ser feita de uma forma ortogonal às entradas e saídas de diferentes algoritmos.

O fato de possibilitar a aquisição de semântica sobre o domínio e utilização da mesma pelos objetos da arquitetura, incentiva o surgimento de aplicações que utilizam métodos indutivos em conjunto com métodos dedutivos. Essa idéia é considerada como uma ótima forma de minimizar a fraqueza de muitos dos algoritmos indutivos

baseados em exemplos, que é a falta de conhecimento sobre o domínio do problema [Núñez 91].

É interessante observar que a arquitetura do A4 privilegia e revigora os algoritmos mais simples, com menos funções embutidas, porque é mais fácil acoplar diversos tratamentos às suas saídas. Algoritmos que já trazem em si próprio tratamento para restrições, podem perder a flexibilidade de se adaptarem a outras formas de tratamento.

O ambiente como um todo visa facilitar a aplicação de técnicas de aprendizado automático provendo, com modularidade e flexibilidade, diversas ferramentas que tratem as limitações e restrições que essas técnicas impõem. Este tratamento é feito de uma forma simples e transparente. A administração das tarefas é realizada pelo gerente do ambiente. Ela é baseada em conhecimento dos componentes da arquitetura, o que possibilita uma perfeita sincronização e sinergia entre as partes do A4.

Um enfoque mais detalhado em torno da modelagem do mundo real, com vistas à organização de dados de treinamento a serem submetidos à algoritmos generalizadores indutivos, mostrou que é necessária uma boa modelagem do domínio, para então se obter resultados eficazes a partir de algoritmos indutivos.

O A4 possui a classe modela domínio que auxilia a modelagem do domínio e é fundamentada em métodos semi-automáticos de aquisição de conhecimento (baseados em entrevistas). Numa das formas de realizar essa modelagem, entrevistas dirigem a apresentação dos exemplos, selecionado os mais representativos. Estes exemplos selecionados se tornam um ótimo conjunto de treinamento para o algoritmo generalizador, tornando o aprendizado rápido com menos esforço dispendido. Com isso, pudemos concluir que um suporte automatizado à obtenção de casos e conhecimento preliminar sobre o domínio torna mais ágil e eficiente a tarefa de aquisição de conhecimento para sistemas baseados em conhecimento.

8.2 Relevância

No nosso entender, as principais contribuições deste trabalho incluem:

- Uma discussão sobre os métodos automatizados de aquisição de conhecimento, com uma análise crítica das principais técnicas utilizadas.

- Uma análise mais profunda sobre os problemas operacionais dos métodos indutivos de aquisição de conhecimento.
- A definição do Ambiente de Apoio a Aquisição Automática de Conhecimento – A4 para auxiliar o processo de AC indutiva.
- Algumas considerações sobre a filosofia de orientação a objeto e a descrição de uma metodologia de desenvolvimento de sistemas baseada nessa filosofia, usada no desenvolvimento do A4.
- A proposição de estratégias para automatizar a modelagem do domínio para os métodos de AC indutiva.
- A implementação da estrutura básica do A4 e das estratégias de modelagem do domínio na classe *modela domínio*. Através dessa implementação pudemos validar a proposta do A4 com casos reais.

8.3 Trabalhos Futuros

Acreditamos que o A4, longe de ser a solução para o problema de AC indutiva, é o começo de uma abordagem para atenuar as dificuldades do processo de AC. No A4 projetamos e implementamos uma configuração mínima. Podemos vislumbrar futuros projetos tanto no tocante a inclusão de novas ferramentas ao ambiente, como novas instanciações dos objetos já existentes. Alguns destes novos projetos poderão ser:

- Validar as heurísticas da modelagem do domínio com a aplicação de casos reais em diferentes domínios.
- Incorporar recursos gráficos ao objeto discretizador para auxiliar a escolha dos atributos nominais durante o processo de discretização.
- Desenvolver uma metodologia de comparação entre soluções, que, além de considerar aspectos como acurácia, tamanho da saída e clareza, contemple o aspecto semântico.
- Construir algoritmos indutivos que incorporem as diversas formas de conhecimento preliminar previstas pelo A4.
- Projetar e implementar objetos que realizem simplificações das saídas geradas pelos algoritmos.
- Projetar e implementar objetos que adicionem estratégias as saídas geradas pelos algoritmos generalizadores, considerando o conhecimento preliminar obtido para o domínio.

- Incorporar ao A4 objetos que implementem técnicas para combinar resultados de diferentes algoritmos generalizadores. Essas técnicas devem prever a combinação de resultados em diferentes formas de representação de conhecimento.
- Implementar objetos que realizem a transformação das diferentes formas de representação possíveis de se utilizar no A4 para alguns importantes *shells* do mercado.
- Prover objetos que auxiliem a validação pelo usuário dos resultados gerados. Para isto deve-se fazer uso das interfaces gráficas a fim de tornar esse processo o mais amigável possível.
- Transformar o gerente do A4 em um SBC, semelhante ao *hipercom* do ambiente integrado de aquisição de conhecimento *HOLOS*[Oliveira 90].

Referências Bibliográficas

[Andrae 84] Andrae, P.M.: *Constrainted limited generalization: Acquiring procedures from examples*. Proc. AAAI, Austin, August, 1984.

[Booch 91] Booch, G.:Object-Oriented Design with Applications. Benjamin/Cummings, 1991.

[Boose 84] Boose, J.H: *Personal Construct Theory and the Transfer Human Expertise*. Proc. National Conference of AI, Austin-Texas. 1984.

[Boose 86] Boose, J,: *Expertise Transfer for Expert Systems Desig*n. New York, Elsevier, 1986.

[Boose 88a] Boose, J.H.: *A Research Framework for Knowledge Acquisition Techniques and Tools*. Proc. EKAW 88,1988.

[Boose 88b] Boose, J.H.: *Uses of Repertory-Grid Centered Knowledge Acquisition Tools for Knowledge-Based Systems*,Proc. EKAW 88,1988.

[Boose 90] Boose, J.: *Knowledge Acquisition for Knowledge-Based Systems*, In Motoda, H., Mizoguchi, R., Boose, J., Gaines, B. (eds). IOS press, Tokyo, 1990.

[Buchanan 78] Buchanan, B.G., Feigenbaum, E.A.: *DENDRAL and Meta-DENDRAL: Their application dimension*. Artificial Intelligence,Vol 11, Nº 1,1978,pp.5-24.

[Buchanan 84] Buchanan, B.G., Shortliffe, E.H.:*Rule based expert system: The MYCIN Experiments of the Stanford Heuristics Programming Project*, Addison-Wesley Publ. Comp., Reading,MA,1984.

[Bussab 85] Bussab, W.O, Severo, C.P.: *Tábuas de estatística*. Harper & Row LTDA. 1985.

[Caianiello 84] Caianiello, E. R., Musso, G.: *Cybernetics Systems: Recognition, Learning, Self-Organization*. Research Studies

Press Ltd., New York,1984.

[Carbonell 86] Carbonell, J.G.: *Derivational analogy: A theory of reconstrutive problem solving and expertise acquisition*, in : Michalski and Mitchell(Eds), Machine Learning: An Artificial Intelligence Aproach 2, Morgan Kaufmann, Los Altos, CA, 1986.

[Carbonell 89] Carbonell, J.G.: *Paradigms for Machine Learnin*g. Artificial Intelligence, v.40, pp: 1-9, 1989.

[Carter 87] Carter, C., Catlett, J.: *Assessing Credit Card Applications Using Machine Learning.* IEEE Expert, fall, 1987.

[Cendrowska 88] Cendrowska,J.: *PRISM: An algorithm for inducing modular rules.* Knowledge-Based Systems. Boose, J. & Gaines. B. (eds). Vol 1. pp: 255-276. Academic Press,1988.

[Cirne 91] Cirne, W., Mongiovi, G.: *Indução Semântica de Regras Modulares*. Anais VIII SBIA. pp:143-148. Brasília, 1990.

[Cirne 92] Cirne, W.:*O Uso de Semântica na Melhoria dos Métodos Indutivos de Aquisição Automática de Conhecimento.* Dissertação de mestrado. Universidade Federal da Paraíba, Campus II, Depto de Sistemas e Computação,1992.

[Clark 89] Clark, P., Niblett, T.: *The CN2 Induction Algorith*m. Machine Learning, vol 3, nº 4, pp:261-284, 1989.

[DeJong 86] Dejong, G., Mooney, R.: *Explanation-based Learning.* Machine Learning, vol 1, pp: 145-176. 1986.

[Diederich 87] Diederich, J.,Linster, L.,Ruhmann, I,Uthmann, T.: *A Methodology for Integrating Knowledge Acquisition Techniques.* Proc EKAW 87. Reading University. 1987.

[Diederich 88] Diederich, J., L., Ruhmann, I, May, M.:*KRITON: a Knoowledge Acquisition Tool for Expert Systems.* Knowledge-Based Systems. Boose, J. & Gaines. B. (eds). Vol 2. Academic Press,1988.

[Dietterich 83] Dietterich, T.G., Michalski, R.S.: *A comparative review of*

selected methods for learning structural descriptions, in : Michalski, Carbonell and Mitchell(Eds), Machine Learning : An Artificial Intelligence Aproach, Tioga, Los Altos, CA, 1983.

[Donato 93] Donato, E. T., Mongiovi, G., Vasco, J.J.P.F.: *Uso de Semântica para melhoria de Aprendizado em um Modelo simbólico-conexionista*. Submetido ao X Simpósio Brasileiro de Inteligência Aritificial. Porto Alegre. 1993.

[Elomaa 89] Elomaa, T.,Holsti, N.: *An Experimental Comparison of Inducing Decision Trees and Decision Lists in Noisy Domains.* Proceedings of th ESWSL 89. Morik, K.(ed). Montpellier. França. Morgan Kaufman. 1990.

[Eshelman 86] Eshelman, L., McDermott, J.: *MOLE: a Knowledge Acquisition Tool That Uses Its Head.* Proc of the National Conference on Artificial Intelligence. Philadelphia, Pennsylvania, 1986.

[Eshelman 88] Eshelman, L.,Ehret, D.,McDermott J.,Tan, M.: *MOLE: a tenacious knowledge-acquisition tool*:knowledge-based system,Vol 2,Academic Press,1988.

[Feigenbaum 81] Feigenbaum, E. A. Expert Systems in 1980's. *The state of the art Report on Machine Intelligence.* A.Bond (ed). Pergamon-Infotech, 1981.

[Furtado 93] Furtado, M.E.S., Schiel, U.:*Uma metodologia para projeto de banco de dados temporal orientado a objetos.* Anais do VIII Simpósio Brasileiro de Banco de Dados. Campina Grande, 1993.

[Glasser 84] Glasser, H., Hankin, C., Till, D.: *Principles of functional programming*. Englewood Cliffs, NJ.Prentice-Hall, 1984

[Gomes 89] Gomes, F.: *Aprend: Um Sistema de Aquisição Automática de Conhecimento a PArtir de Exemplos.* Dissertação de mestrado. Universidade Federal da Paraíba, Campus II, Depto de Sistemas e Computação,1989.

[Gaines 88] Gaines, B.R: *Integration issues for Knowledge Suport Systems.* Knowledge Based Systems,vol 1,Academic Press,1988.

[Gammack 88] Gammack, J., Young, R. *Psychological tecniques for elicitation expert knowledge.* Cambridge University Press,1985.

[Genesereth 87] Genesereth, M.R, Nilsson, N.J.: *Logical Foundations of Artificial Intelligence.* Los Altos, CA. Morgan Kauffman, 1987.

[Gruber 89] Gruber, T.: *The Acquisition of Strategic Knowledge.* Academic Press. 1989.

[Hart 87] Hart, A., *Induction and Knowledge Elicitation.* In Knowledge Acquisition for Expert Systems, Kidd, A. (ed). Plenum Press, New York, 1987.

[Hinton 84] Hinton, G.E., Sejnowski, T.J., Ackley, D.H.: *Boltzmann machines: constraint satisfaction networks that learn.* Relatório técnico, Computer Science Department, Carnegie-Mellon University, Pittsburgh, PA, 1984.

[Holland 75] Holland, J.:*Adaptation in Natural and Artificial Systems.* University Michigan Press, Ann Arbor, MI, 1975.

[Holland 86] Holland, J.H.: *Escaping brittleness: The possibilites of general-purpose learning algorithms applied to parallel rule-based systems.* In : Michalsky and Mitchell(Eds), Machine Learning: An Artificial Intelligence Aproach 2, Morgan Kaufmann, Los Altos, CA, 1986.

[Kawaguchi 87] Kawaguchi, A.,Mizoguchi R.,Yamaguchi, T.,Kakusho, B.:*SISa shell for interview systems.* Principles of expert systems,1987.

[Kelly 55] Kelly, G.: *The psichology of personal constructs.*New York:Norton,1955.

[Kokar 86] Kokar, M.M.: *Determining arguments of invariants functions.* Machine Learning 4, 403-422, 1986.

[Langley 83] Langley, P., Bradshaw, G.L., Simon, H.A.: *Rediscovering chemistry with the Bacon system.* Machine Learning 1, 307-329,1983.

[Mizoguchi 91] Mizoguchi, R.: *Classification Knowledge Acquisition System with an Atribute Elicitation Facility.* Proc. Australian Workshop on Knowledge Acquisition for Knowledge-Based Systems. Polkobin, 1991.

[Mongiovi 90a] Mongiovi, G., Alencar, W., Baeny, X.: *Ginecol: Um Sistema de Diagnóstico Ginecológico.* Relatório Técnico. Universidade Federal da Paraíba. Depto de Sistemas e Computação. 1990.

[Mongiovi 90b] Mongiovi, G. Cirne, W.: *Um Algoritmo Baseado em Conhecimento Para Ampliar a Potencialidade do ID3.* Anais do VI simpósio Brasileiro de Inteligência Artificial, 1990.

[Mongiovi 93] Mongiovi, G., Vasco, J.J.P.F.: *Relevância Semântica Nebulosa para os Métodos Indutivos de Aquisição de Conhecimento.* Submetido ao X Simpósio Brasileiro de Inteligência Artificial, Porto Alegre, Outubro, 1993.

[Niblett 87] Niblett, T.: *Constructing Decision Trees in Noisy Domains.* In progress in Machine Learning, Proceedings of EWSL 87. Bled, Iugoslávia. Bratko, I. & Lavrac, N. (eds). Sigma Press. Wilmslow,UK, 1987.

[Nilsson 80] Nilsson, N.: *Principles of Artificial Intelligence.* Palo Alto, CA. Tioga. 1980.

[Núñez 91] Núñez, M.: *The Use of Background Knowledge in Decision Tree Induction.* Machine Learning, 6, pp:231-250, Kluwer Academics Publishers, Boston, 1991.

[Oliveira 89] Oliveira, J. e Mongiovi. G.: *Holos: Um Ambiente Integrado de Aquisição Automática de Conhecimento.* Anais do 1o Simpósio de Inteligencia Artificial y Robotica. Luján, Argentina, 1990.

[Pope 81] Pope, M.L.,Shaw, M.L.G.:*Personal construct psichology in education e learning.* In Shaw, M.L.G.,Ed. Recent Advances in Personal Constructs Technology. Academic Press,Londres,1981.

[Quinlan 79] Quinlan, J.R.: *Discovering rules by induction from large collection of examples.* In Michie, D. (ed), Expert Systems in the Microeletronic Age. Edinburgh University Press, 1979.

[Quilan 83] Quinlan, J.R.: *Learning Efficient Classification Procedures and their Application to Chess End Games.* In Machine Learning: An Artificial Intelligence Aproach. Michalsky, R., Carbonell, J., Mitchell, T. (eds). Vol 1. Morgan Kaufmann.1983.

[Quinlan 86] Quinlan, J.R.: *The Effect of Noise on Concept Learnin*g. In Machine Learning: An Artificial Intelligence Aproach. Michalsky, R., Carbonell, J., Mitchell, T. (eds). Vol 2. Morgan Kaufmann.1986.

[Quilan 87a] Quilan, J.R.: *Simplifying Decision Trees*. International Journal of Man-Machine Studies 27,3,221-234,1987.

[Quilan 87b] Quinlan, J.R., Compton, P.J., Horn, K.A., Lazarus, L.: *Inductive Knowledge Acquisition: a Case Study.* In applications of Expert Systems, Quinlan, J.R.(ed). Turing Institute Press. Sidney, 1987.

[Rosenblatt 58] Rosenblatt, F.: *The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain.* Psychological Review, Vol. 65, 386-407, 1958.

[Sammut 86] Sammut, C.: *Learning concepts by asking questions.* Machine Learning 2, 167-191, 1986.

[Samuel 63] Samuel, A.L.: *Some studies in machine learning using the game of checkers.* Feigenbaum and Feldman Eds. Computers and Thought. McGraw-Hill, New York, 1963.

[Schank 82] Schank, R.C.:Dynamic Memory: *A theory of reminding and Learning in Computers and People.* Cambridge University Press, Cambridge, 1982.

[Selfridge 59] Selfridge, O.G.: *Pandemonium: A Paradigm for Learning. Proc. of Symposium on Mechanization of Thought Processes.* In Blake, D. and Uttley, A. Eds. HMSO, London, 511-529, 1959.

[Shaw 80] Shaw, M.L.G.: *On becoming a Personal Scientist.*Academic Press.Londres,1980.

[Shaw 88a] Shaw, M.L., Gaines, B.:*KITTEN: Knowledge Initiation and Transfer Tools for Experts and Novices.* Knowledge-Based Systems. Vol 2. Boose, J. & Gaines, B. (eds). Academic Press, 1988.

[Shaw 88b] Shaw, M.L., Woodward, J.B.: *Validation of Knowledge Support System.* Proc EKAW 88. 1988.

[Shepard 82] Shepherd, E.,Watson, J.P.:*Personal Meanings.* Ed. John Wiley. Londres,1982.

[Soult 88] Soult, J.,Caplain, G.,Marcus, S.,McDermott, J.:*Toward Automating Recognition of Differing Problem-Solving Demands.* Proc. EKAW 88. 1988.

[SUN 91] *XVIEW Programming Language Manual.* Sun MicroSystems, 1991.

[Tsypkin 72] Tsypkin, J.Z.: *Foundations of the Theory of Learning Systems*(in Russian), Publisher Nauka, Moscow, 1972.

[Van de Velde 89] Van de Velde, W.: *IDL, or Taming the Multiplexer.* Proc. Of the Fourth European Working Session on Learning. Montpellier, December, 1989.

[Winston 75] Winston, P.: *Learning structural descriptions from examples.* In : Winston (Ed.), The Psychology of Computer Vision, McGraw-Hill, New York, 1975.

[Vasco 92a] Vasco, J.J.F., Mongiovi, G., Cirne Filho, W., Donato Jr, E.T.: *A4 – Ambiente de Apoio a Aquisição Automática de Conhecimento.* Anais do IX SBIA (Simpósio Brasileiro de Inteligência Artificial), Rio de Janeiro, 1992.

[Vasco 92b] Vasco, J.J.F, Mongiovi, G., Cirne, W., Donato, E.: *Modelagem do Domínio para Aquisição de Conhecimento Indutivo.* Anais do IX SBIA(Simpósio Brasileiro de Inteligência Artificial), Rio de Janeiro, 1992.

[Vasco 93a] Vasco, J.J.F., Mongiovi, G., Cirne Filho, W., Donato Jr, E.T.: *Modeling the Real World for Knowledge Acquisition Inductive Methods.* Computer Science International Congress. Mendoza, Argentina. Junho, 1993.

[Vasco 93b] Vasco, J.J.F, Mongiovi, G., Cirne, W., Donato, E.: *Um Ambiente para Aquisição Automática de Conhecimento a Partir de Exemplos.* Anais da IX Jornada Latino Americana de Informática. Buenos Aires, Argentina. Agosto,1993.

Apêndice A

Métodos Reutilizáveis pelos Projetistas de Algoritmos

Classe *Exemplo*

int QualQtdeExemplos();	// Retorna a quantidade de exemplos da tabela de exemplos corrente
int QualQtdeAtributos();	// Retorna a quantidade de atributos da tabela de exemplos corrente
int QualQtdeClasse();	// Retorna a quantidade de classes na tabela de exemplos corrente
int QualQtdeAtribVal();	// Retorna a quantidade de pares atributo/valor na tabela de exemplos corrente
int freq_classe(int cl);	// Retorna a frequência de aparecimento da classe *cl* na tabela de exemplos corrente. Em caso de erro retorna -1.
int freq_atributo(int at);	// Retorna a frequencia em que o atributo *at* apareceu com valores na tabela de exemplos corrente.
int freq_valor(int at, int val);	// Retorna a frequencia de aparecimento do valor *val* para o atributo *at* na tabela de exemplos corrente
int FreqExemplo(int ex);	// Retorna a frequencia do exemplo *ex* na tabela de exemplo corrente.
int CustoAtributo(int at);	// Retorna o custo do atributo *at*. Em caso de erro ou ausência retorna -1.

int ValAtribExemplo(int ex, int at);	// Retorna o valor do atributo *at* no exemplo *ex*.. Caso haja buraco o valor retornado é -1 e em caso de erro -2.
int ClassedoExemplo(int ex);	// Retorna o valor da classe para o exemplo ex.
char *nome_atributo(int at);	// Retorna um ponteiro para o nome do atributo de codigo *at*.. Caso *at* não exista, retorna um ponteiro para um " " (nome em branco)
char *nome_valor(int val);	// Retorna um ponteiro para o nome do valor de có- digo *val*. Caso *val* não exista, retorna um ponteiro para um " " (nome em branco)
char *nome_classe(int cl);	// Retorna um ponteiro para o nome da classe de código *cl*. Caso *cl* não exista, retorna um ponteiro para um " " (nome em branco)
int código_atributo(char *nomeat);	// Retorna o código do atributo de nome *nomeat*. Caso *nomeat* não exista, retorna -1.
int código_classe(char *nomecl);	// Retorna o codigo da classe de nome *nomecl*. Caso *nome*cl não exista, retorna -1.
int código_valor(char *nomeval);	// Retorna o código do char *nomeval); valor de nome *nomeval*. Caso *nomeval* não exista, retorna -1.

Exemplo de Utilização:

```
#include A4exem.H
Exemplo *ex;
..
..
..
// Imprime os valores de um tabela de exemplo
```

```
for (int i=0; i<QualQtdeExemplos;i++)
   for (int j= 0; j< QualQtdeAtributos;j+ + )
      fprint("Valor Atributo = = %s", ex-nome_valor (ex-ValAtribExemplo (i, j)));
```

Classe *Relevância*

int RelevanciaCondClasse(int val, int at, int cl);	// Retorna o grau de relevância de um valor *val* de um atributo *at* para uma classe *cl.* Senão retorna zero.

Exemplo de utilização

```
#include A4rel.H
#include A4exem.H
Relevância *rel;
Exemplo *ex;
..
..
..

// Se o valor 1 do atributo 1 for relevante a classe 1 então o nome do valor, do atributo e da
classe é impresso

if (rel->RelevanciaCondClasse(1,1,1))
   printf("Valor %s do Atributo %s é relevante para a classe %s, ex-> nome_valor (1),
ex->nome_atributo(1), ex-nome_classe(1));
```

Classe Hierarquia

int codigo_generalizacao(char *nomegn);	// Retorna o codigo do nome da generalização *nomegn.* Caso *nomegn* não exista, retorna -1
char *nome_generalizacao(int codgn);	// Retorna um ponteiro para o nome da generalização de código *codgn.* Caso *codgn* não exista, retorna -1
int QualQtdeValporGen(int gen, int atrib);	// Retorna a Quantidade de valores generalizados pela generalização *gen* para o atributo *at.*

int QualGenAtribVal(int at, int val);	// Retorna a generalização para o par atributo/valor *at / val.*

Exemplo de Utilização

```
#include A4hiera.H
#include A4exem.H
Hierarquia *hrq;
Exemplo *ex;
..
..
..

// Imprime o nome da generalização do valor 1 e do atributo 1

   printf("Valor %s do Atributo %s é generalizado por %s, ex-> nome_valor (1),
ex->nome_atributo(1),hrq-> nome_generalização(hrq->QualGenAtribVal(1, 1)));
```

Classe Algoritmo

virtual void executa();	// Executa o corpo do algoritmo generalizador que estiver instanciado
float EntropiaAtributo(int at);	// Calcula o valor da entropia do atributo *at* na tabela de exemplos corrente.
int AtributoMenorEntropia();	// Retorrna o atributo de menor atributo da tabela de exemplos corrente.
void FormaSubConjunto(int at, int val);	// Retira da tabela de exemplos corrente os exemplos que não tem valor *val* para o atributo *at*..
void RecuperaSubConjunto(int at, int val);	// Recupera para a tabela de exemplos corrente os exemplos que não tem valor val para o atributo at.
void ExcluiExMapeado(int at, int val, int cl);	// Exclui da tabela de exemplos corrente os exemplos em que o atributo *at* tem valor *val* e que conclua a classe *cl*.

int FreqCondClasse(int at, int val, int cl);	// Retorna a frequencia de aparecimento da condição atributo *at* e valor *val* concluindo a classe *cl*.

Exemplo de Utilização

```
#include A4alg.H
#include A4exem.H
Algoritmo *alg;
Exemplo *ex;
..
..
..

// Calcula a Entropia do atributo idade

float at = alg->EntropiaAtributo(ex->codigo_atributo("Idade"));
```

Apêndice B

Como Incluir Algoritmos no A4

Infelizmente, a linguagem C++ por ser uma linguagem basicamente estática, característica herdada de sua precursora linguagem C, impossibilita a inclusão dinâmica de classes no ambiente A4. Em função disso o projetista de novos algoritmos é obrigado a atualizar o código fonte do ambiente para incluir a classe de algoritmos que ele desenvolveu. O que buscamos no A4 foi minimizar os efeitos colaterais desse processo, simplificando-o ao máximo.

Quando do desenvolvimento de um novo algoritmo o projetista deverá criar uma nova classe que comportará o corpo do algoritmo. Esta nova classe deve obrigatoriamente ser subclasse da classe algoritmo ou de outras subclasses já definidas no ambiente. A figura B.1 mostra a hierarquia inicial que o A4 fornece. Esta nova classe deve possuir o método **executa()**, no qual deve estar definido o corpo básico do algoritmo.

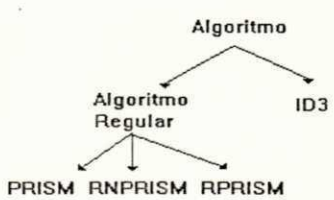


Figura B.1 Hierarquia básica de algoritmos em A4

Com o algoritmo desenvolvido seguindo essas normas, o projetista deve editar a implementação da classe *Gerente* (arquivo A4gerente.C) e incluir, dentro do método inicializa_algoritmo, uma mensagem ao método inclui_algoritmo passando, como parâmetro, um ponteiro para o algoritmo a ser incluído. Ainda na implementação da classe *Gerente* é necessário incluir uma cláusula *#include* com o nome da parte de interface da classe a ser incluída (classenova.H). Um exemplo desse processo segue a seguir:

Arquivo A4gerente.C

```
#include "NovoAlg.H"
..
..
..
void Gerente::inicializa_algoritmo()
 {
 ..
 ..
 ..
inclui_algoritmo(new *novo_algoritmo);
}
```

Feita essa alteração o projetista deve editar o arquivo Makefile do diretório */home/A4* e incluir o nome da classe que ele desenvolveu na clausula SOURCES.C.

Após essas modificações, o projetista pode recompilar o ambiente, executando o comando make dentro do diretório */home/A4* e utilizar a nova versão do ambiente. É importante lembrar que para utilizar esse novo algoritmo, ele deve estar catalogado no ambiente. Isso é feito na tela de tratamento de algoritmos (ver capítulo 7, figura 7.16) com a opção de inclusão do algoritmo e informado as suas características básicas.

UFPb/BIBLIOTECA/PRAI

APÊNDICE C

O Processo de Discretização no A4

O processo de discretização no A4 é realizado através de análises estatísticas da ocorrência dos valores contínuos de um atributo. Nessa análise, para cada exemplo, consideramos a relação existente entre o valor do atributo a ser discretizado e o elemento de classificação. A partir de toda a tabela de exemplos, através de médias aritméticas dos valores do atributo contínuo e do cálculo de intervalos de confiança para cada elemento classificador, o A4 sugere un conjunto de intervalos no qual pode ser discretizado o atributo.

É importante salientar que a solução adotada no A4, longe de ser a solução para o problema de discretização de atributos com valores contínuos, constitui-se apenas uma forma possível de atacar o problema. Consideramos importante a existência do mecanismo por nós definido, pois ele provê uma considerável ajuda ao usuário. Mas percebemos também, que há algumas falhas neste mecanismo, dentre as quais consideramos a principal, o fato do número de intervalos gerados ser definido em função do número de classes do problema.

O processo de discretização utilizado pelo A4 pode ser definido da seguinte forma.

Sejam:

TE - uma tabela de exemplos com Nex exemplos;

$C = \{c_1, c_2, c_3, \ldots, c_n\}$ - O conjunto de Ncl elementos classificadores;

n_i - o número de exemplos que concluem a classe i;

A - Um atributo ordinal que pertence a TE e deve ser discretizado;

$VC_i = \{V_{i1}, V_{i2}, \ldots, V_{ini}\}$ - O conjunto de valores do atributo A em exemplos que concluem o elemento de classificação i

Calculamos a média aritmética α_i, para cada elemento de classificação, como sendo:

$$\alpha_i = \left(\sum_{j=1}^{n_i} V_{ij} \right) / n_i$$

A partir das médias aritméticas α_i para cada classe calculamos um intervalo de confiança IC_i com $[IC_{imin}, IC_{imáx}]$, dado por,

$$IC_{min} = \alpha_i - Ts\,(Q/n_i)^{1/2}$$

$$IC_{máx} = \alpha_i + Ts\,(Q/n_i)^{1/2}$$

Onde,

Ts é o valor obtido na tabela t-student [Bussab 85] com (Nex - Ncl) graus de liberdade;

$$Q = (1 / Nex\text{-}Ncl) \sum_{i=1}^{Ncl} \sum_{j=1}^{n_i} [Vij - \alpha_i]^2$$

A partir dos Ncl intervalos de confiança(IC) calculados partimos para o cálculo dos intervalos finais(IF). Primeiramente, todos os limites inferiores e superiores são ordenados. Com isso obtemos um conjunto de limites L={l1, l2, l3 ..., ln} onde l1<l2<l3<...<ln. Os intervalos finais são formados a partir da ordem deste conjunto. Ou seja o conjunto IF de intervalos finais é IF={[l1,l2[,[l2,l3[[l3,l4[,...,[ln-1,ln]}.

Após a formação do conjunto de intervalos finais podemos utilizar algumas heurísticas para refinar esse conjunto obtido. A primeira heurística é definir os limites mínimo e máximo do primeiro e último intervalo respectivamente. Isso é feito através do cálculo do menor e do maior valor existente na tabela de exemplo. Uma outra heurística nos permite eliminar alguns intervalos, após verificarmos se existem intervalos que não englobem nenhum valor da tabela de exemplos. Nesse caso esses intervalos são desnecessários.

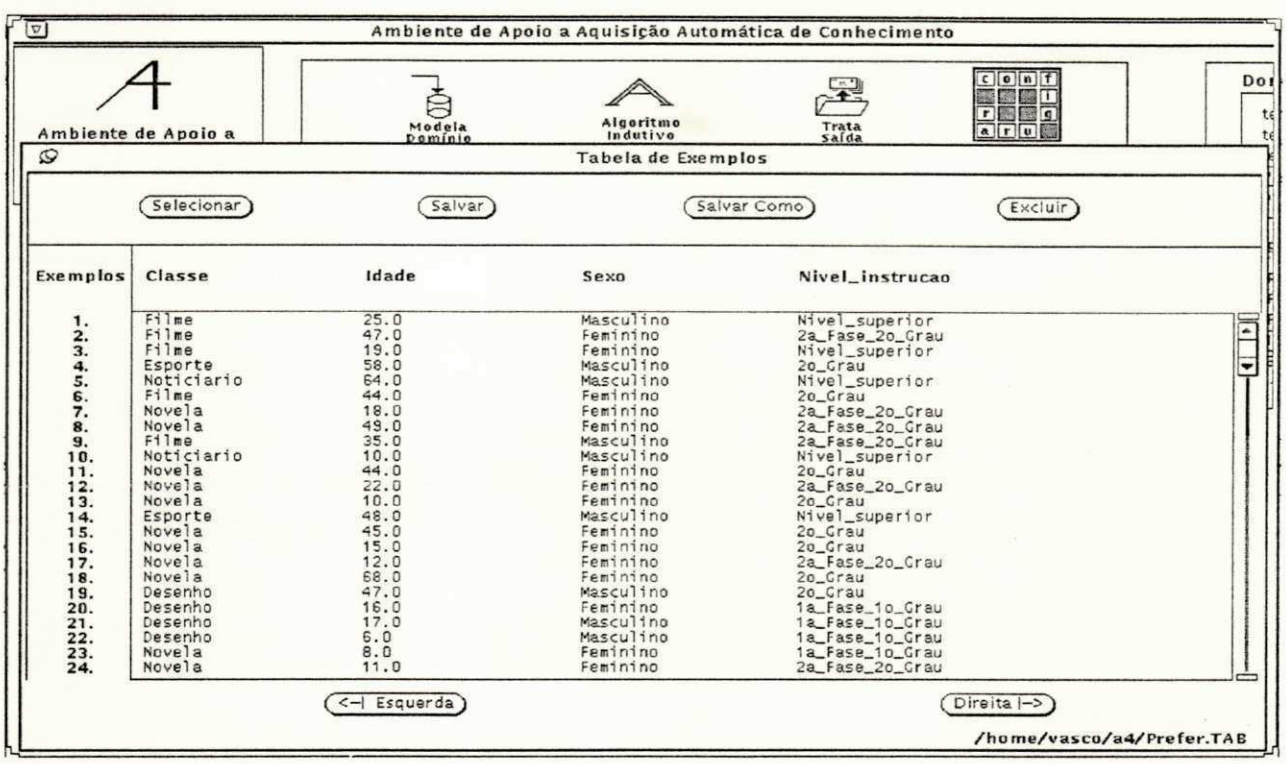


Figura C.1 Tabela de exemplos com atributo ordinal

A seguir mostraremos como o processo de discretização é feito no A4.

A figura C.1 nos mostra a tabela de exemplos a ser trabalhada. Essa tabela de exemplos representa o domínio de preferências de programas de televisão. O atributo idade esta definido como ordinal.

Quando o A4 é solicitado para executar um algoritmo generalizador, a existência de atributos contínuos será detectada e a rotina de discretização é ativada. Após a aplicação do processo de discretização, tal como foi descrito anteriormente, o A4 mostra os intervalos finais. A figura C.2 mostra os intervalos gerados. Neste momento o usuário tem a opção de aceitar, rejeitar ou modificar os intervalos gerados.

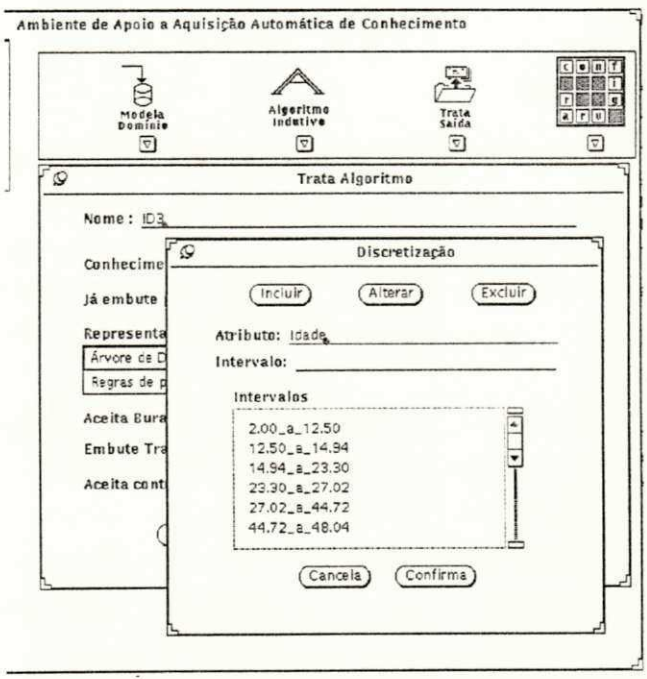
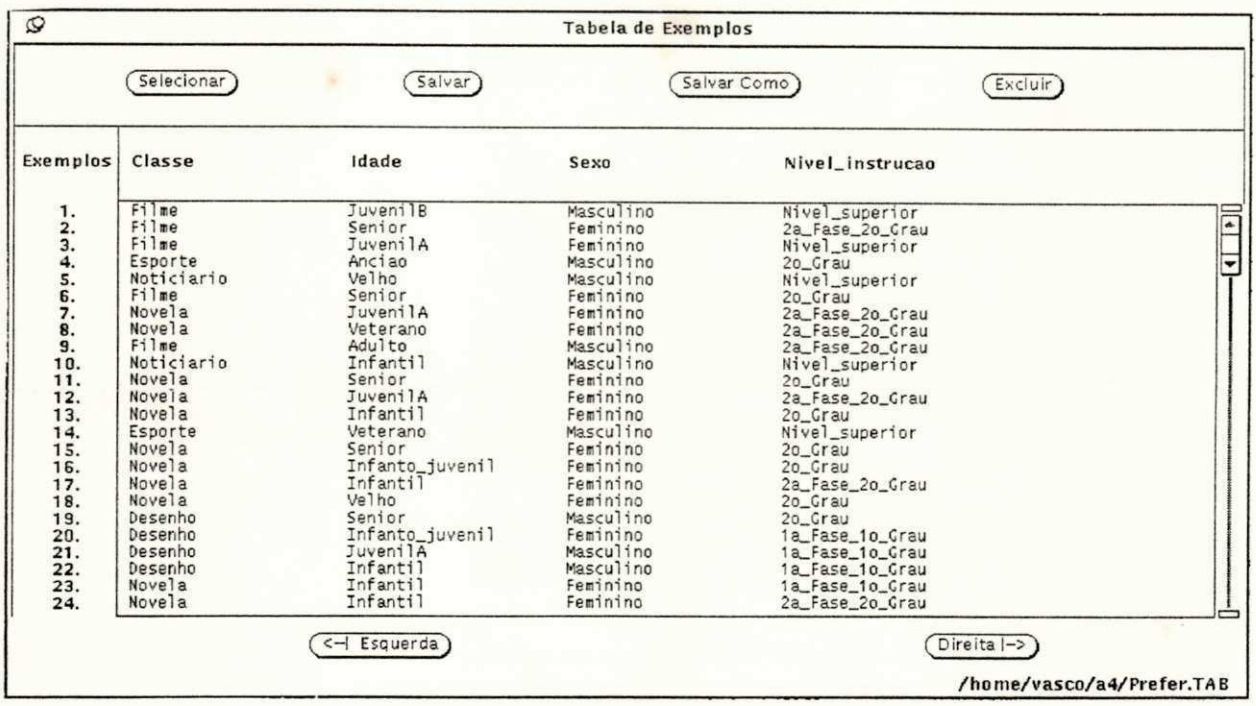


Figura C.2 Intervalos sugeridos para o atributo idade



Exemplos	Classe	Idade	Sexo	Nivel_instrucao
1.	Filme	JuvenilB	Masculino	Nivel_superior
2.	Filme	Senior	Feminino	2a_Fase_2o_Grau
3.	Filme	JuvenilA	Feminino	Nivel_superior
4.	Esporte	Anciao	Masculino	2o_Grau
5.	Noticiario	Velho	Masculino	Nivel_superior
6.	Filme	Senior	Feminino	2o_Grau
7.	Novela	JuvenilA	Feminino	2a_Fase_2o_Grau
8.	Novela	Veterano	Feminino	2a_Fase_2o_Grau
9.	Filme	Adulto	Masculino	2a_Fase_2o_Grau
10.	Noticiario	Infantil	Masculino	Nivel_superior
11.	Novela	Senior	Feminino	2o_Grau
12.	Novela	JuvenilA	Feminino	2a_Fase_2o_Grau
13.	Novela	Infantil	Feminino	2o_Grau
14.	Esporte	Veterano	Masculino	Nivel_superior
15.	Novela	Senior	Feminino	2o_Grau
16.	Novela	Infanto_juvenil	Feminino	2o_Grau
17.	Novela	Infantil	Feminino	2a_Fase_2o_Grau
18.	Novela	Velho	Feminino	2o_Grau
19.	Desenho	Senior	Masculino	2o_Grau
20.	Desenho	Infanto_juvenil	Feminino	1a_Fase_1o_Grau
21.	Desenho	JuvenilA	Masculino	1a_Fase_1o_Grau
22.	Desenho	Infantil	Masculino	1a_Fase_1o_Grau
23.	Novela	Infantil	Feminino	1a_Fase_1o_Grau
24.	Novela	Infantil	Feminino	2a_Fase_2o_Grau

Figura C.3 Tabela de exemplos com atributo discretizado

Após a confirmação de aceitação da discretização, o A4 atualiza a tabela de exemplos original trocando todos os valores contínuos para os seus novos valores nominais. Depois dessa atualização o usuário pode modificar o nome dos valores para nomes que sejam mais repsentativos para o domínio. A figura C.3 mostra a tabela de exemplos do domínio de preferências após a tarefa de discretização e após a mudança do nome dos valores dos atributos.