UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE CIÊNCIAS E TECNOLOGIA
COORDENAÇÃO DE PÓS-GRADUAÇÃO EM INFORMÁTICA

O Uso da Semântica na Melhoria dos Métodos Indutivos de Aquisição Automática de Conhecimento

Walfredo da Costa Cirne Filho

Campina Grande
Dezembro / 1992

Walfredo da Costa Cirne Filho

O Uso da Semântica na Melhoria dos Métodos Indutivos de Aquisição Automática de Conhecimento

Dissertação apresentada no curso de
MESTRADO EM INFORMÁTICA da
Universidade Federal da Paraíba, em
cumprimento às exigências para
obtenção do grau de Mestre

Área de Concentração: Ciência da Computação

Giuseppe Mongiovi
Orientador

Campina Grande
Dezembro / 1992

C578u Cirne Filho, Walfredo da Costa
O uso da semantica na melhoria dos metodos indutivos de aquisicao automatica de conhecimento / Walfredo da Costa Cirne Filho. - Campina Grande, 1992.
64 f. : il.

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

1. Banco de Dados - 2. Algoritmo - 3. Semantica nos Algoritmos 4. Dissertacao I. Mongiovi, Giuseppe, M.Sc. II. Universidade Federal da Paraiba - Campina Grande (PB) III. Título

CDU 004.63(043)

O Uso de Semântica na Melhoria dos Métodos Indutivos de Aquisição Automática de Conhecimento

Walfredo da Costa Cirne Filho

Dissertação aprovada em 02/12/1992

Giuseppe Mongiovi, M.Sc.
Orientador

Maria de Fatima Camelo, M.Sc.
Componente da Banca

Hélio de Menezes Silva, M.Sc.
Componente da Banca

João Marques de Carvalho, Ph.D.
Componente da Banca

Campina Grande, PB
Dezembro/1992

Resumo

Neste trabalho, investigamos como o uso de semântica adquirida do especialista pode melhorar a qualidade das respostas fornecidas pelos algoritmos de aprendizado automático baseados em indução. Inicialmente mostramos como introduzir o uso de semântica nos algoritmos da família TDIDT (os mais conhecidos algoritmos indutivos). Todavia, mesmo com semântica, ainda temos um problema de natureza sintática devido à forma utilizada pelos algoritmos TDIDT para representar o conhecimento adquirido (Árvores de Decisão). Na busca da solução desse problema, apresentamos o algoritmo PRISM, que resolve o problema sintático embora também padeça do problema semântico como os algoritmos TDIDT. Finalmente, propomos o algortimo RPRISM, que resolve o problema sintático como o PRISM e o semântico, via uso de conhecimento obtido do especialista.

Agradecimentos

Gostaria sinceramente de agradecer a todas as pessoas que direta ou indiretamente tornaram este trabalho possível. Mesmo sabendo da impossibilidade de lembrar de todos e, portanto, de não esquecer de ninguém, vou agradecer explicitamente às pessoas que mais contribuiram com o trabalho. De antemão, peço desculpas às pessoas que deveriam estar aqui citadas e foram esquecidas. A memória humana é fraca.

Agradeço inicialmente a toda minha família, em particular aos meus pais (Walfredo e Célia) e aos meus irmãos (Márcia, Gustavo e Henrique), pelo apoio e compreensão. A Ana, minha namorada, pela compreensão e pelo companherismo. Que dure para sempre.

A Giuseppe (meu inspirado orientador); a Albégio e Alexandre (pela ajuda na digitação da dissertação); a Vasco e Tojal (pelas inestimáveis correções); a Andrezinho (pela ajuda na editoração) e a Zé Ricardo (pelas transparências), meus agradecimentos pela ajuda direta neste trabalho.

Aos meus amigos, companheiros de tantas cervejas e orientações. Agradecimentos orgásticos a Marcos Sebastian, Luiz, Simonex, Guilherme, Bolão, Guto, Gedeon, Jacques, Tárcio, Jarley, Israel, Peter, René, Tojal, Ana, Alexandre, Derly, Márcia, Fernando "Mande", Fernando, Elmar, Soninha, Luis Maurício, Ari, Bics, Maysa, Roberta, Trícia, Nicolaas, France, Vasco, Fubica, Erico e Plínio. Agradecimentos em separado para minha verdadeira turma do 2º grau (Sérgio, Mark e Márcio) que muito ajudaram no meu amadurecimento.

Agradecimento especial a todos os professores que tive até hoje, pois muito influenciaram no que sou hoje. Em particular a Jacques, Derly, Júlio, Peter, Giuseppe e Hélio.

Agradecimentos ainda a todos os meus colegas de mestrado e gradução pelo companheirismo e aos meus colegas de trabalho na Industrial Cirne pela compreensão e pelo apoio.

Por fim, agradeço a todos que gostam de mim e também àqueles que passarão a gostar, pois a vida é bela, esta tarde de domingo está lindíssima e, como dizem, "vai dar tudo certo".

Sumário

1. Introdução

1.1. Inteligência Artificial

1.2. Sistemas Baseados em Conhecimento

1.3. Objetivos da Dissertação

1.4. Estrutura da Dissertação

2. Aquisição de Conhecimento

2.1. Métodos Cognitivos

2.2. Métodos Automáticos

2.2.1. Apendizado Automático em Redes Neurais Artificiais

2.2.2. Aprendizado Automático por Analogia

2.2.3. Aprendizado Automático a partir de Instrução

2.2.4. Apendizado Automático por Indução

2.2.4.1. Tipos de Indução

2.2.4.2. Árvores de Decisão

3. A Família de Algoritmos TDIDT

3.1. Procedimento Geral

3.2. O Algoritmo ID3

3.3. Análise dos Algoritmos TDIDT

4. O Problema Semântico e sua Solução via Árvores de Decisão

4.1. O Problema Semântico

4.2. Árvores de Decisão Semânticas

4.2.1. Matriz de Relevância

4.2.2. Expansão Semântica de Árvores de Decisão

4.2.3. Construção Direta de Árvores de Decisão Semânticas

5. O Problema Sintático e sua Solução Inicial

5.1. O Problema Sintático

5.2. Indução de Regras Modulares

5.2.1. Onde Mudar os Algoritmos TDIDT

5.2.2. O Algoritmo PRISM

5.2.3. Exemplo com o Algoritmo PRISM

6. Indução Semântica de Regras Modulares

6.1. O Algoritmo RPRISM

6.2. Uma Aplicação em Medicina

7. Protótipo do Aquisitor Semântico

7.1. Arquitetura Interna

7.2. Modo de Utilização

7.2.1. Definição das Entradas

7.2.2. Executando o Protótipo

8. Conclusões e Trabalhos Futuros

8.1. Conclusões

8.2. Trabalhos Futuros

Referências Bibliograficas

Apêndice A. O Algoritmo IDL

A.1. Relevância Topológica

A.2. A Técnica de Transformação

A.3. O Algoritmo

Apêndice B. O Algoritmo C4

B.1. O Método de Poda do Algoritmo C4

B.2. Um Exemplo do Algoritmo C4

B.3. Análise Crítica do Algoritmo C4

B.4. Outros Métodos de Poda

1. Introdução

1.1. Inteligência Artificial

A primeira questão que nos vem a mente quando começamos a trabalhar em uma determinada área é como vamos definí-la. Todavia, parece que as tentativas de definir clara e precisamente termos complexos e de utilização variada são um simples exercício de retórica. Em geral, não se chega a lugar nenhum. Contudo, tentaremos, pelo menos, delinear quais os objetos de estudo da Inteligência Artificial (IA). Um bom conceito, embora não universalmente aceito, é:

A inteligência artificial estuda como fazer os computadores realizarem bem certas tarefas que, embora complicadas para máquinas, são simples para seres humanos [Rich 83].

Um exemplo típico de uma atividade simples para o ser humano e complicada para a máquina é o ato de falar (e de entender a fala).

É certo que este é um conceito bastante informal e sujeito a modificações na medida em que a ciência da computação for avançando. Porém, o lento progresso da IA no tocante a fazer os computadores solucionarem tarefas "difíceis", torna este conceito, pelo menos nos próximos anos, um bom delimitador da área.

1.2. Sistemas Baseados em Conhecimento

O conhecimento tem importância vital para várias das técnicas usadas em IA. Para alguns sistemas, o conhecimento é tão importante que eles são ditos Sistemas Baseados em Conhecimento (SBCs). Nos SBCs é dado um tratamento explícito e independente ao conhecimento, em oposição à abordagem de fundir o conhecimento usado com o controle, típica dos sistemas convencionais. Esta separação se faz necessária porque, nos sistemas baseados em conhecimento, o volume de conhecimento é tal que pulverizá-lo pelo sistema impossibilitaria sua construção e manutenção.

Os SBCs mais conhecidos são os sistemas especialistas. Um sistema especialista lida com problemas complexos do mundo real que requerem a interpretação de um especialista. Para isto é utilizado um modelo computacional do raciocínio humano.

Em um sistema baseado em conhecimento, o conhecimento, devidamente representado, constitui a Base de Conhecimento (BC). Aquisição de Conhecimento (AC), por sua vez, é o processo que permite a transferência do conhecimento sobre a solução de um determinado problema para a base de conhecimento. Além disso, precisamos de algoritmos que usem o conhecimento representado na BC. Estas tarefas estão a cargo do Engenheiro do Conhecimento (EC), que é o profissional (geralmente especializado em computação e em IA) a conceber e construir os sistemas baseados em conhecimento.

Portanto, a construção de um SBC envolve basicamente três tarefas: a aquisição, a representação e o uso do conhecimento. Porém, a aquisição de conhecimento revelou-se ser o "gargalo" desse processo [Feigenbaum 81], o que a torna de suma importância.

1.3. Objetivos da Dissertação

Devido a lentidão do processo de aquisição de conhecimento, vários métodos automáticos para esta tarefa foram propostos. Dentre eles, há aqueles que funcionam fazendo indução por computador, nos quais se destacam os da família TDIDT (*Top Down Induction of Decision Trees*) [Quinlan 86].

A meta principal dos algorítmos TDIDT é mapear um conjunto de exemplos em uma árvore de decisão de tamanho mínimo (altura e largura). Obviamente que, do ponto de vista puramente quantitativo, este objetivo é válido, pois implica em uma BC reduzida. Entretanto, em geral, para serem utilizados em SBCs, os resultados obtidos pelos algorítmos TDIDT apresentam dois problemas críticos: um sintático e outro semântico.

Os algorítmos TDIDT, por definição, geram sua saída na forma de uma árvore de decisão. As árvores de decisão constituem um problema porque não equivalem a um conjunto de regras modulares, no sentido que sempre há um atributo presente em todas as regras. Isto permite, principalmente se existerem atributos com um grande número de valores, o aparecimento de anomalias, pois nem todos os valores daquele atributo são relevantes para a conclusão dos elementos de classificação. Isto caracteriza o problema sintático.

A árvore de decisão gerada pelos algorítmos da família TDIDT é uma estrutura despida de informação semântica. Ela apenas mapeia, de forma reduzida, o conjunto de exemplos. Devido a este fato, pode ocorrer que condições irrelevantes apareçam na árvore. Isto ocorre porque, em aplicações reais, nunca trabalhamos com um conjunto completo de exemplos (i. e., um conjunto que cubra todas as combinações possíveis de condição x classe), o que permite o aparecimento de

condições irrelevantes por simples coincidência. Uma circunstância em que este problema frequentemente aparece é quando o conjunto de exemplos contém casos raros (casos que denotam classes que ocorrem apenas em pouquíssimos exemplos). Este é o problema semântico.

O problema sintático foi apresentado por Cendrowska [Cendrowska 88] e o semântico foi descoberto durante a pesquisa que viabilizou esta dissertação [Mongiovi 90].

Os objetivos principais desta dissertação são estudar a fundo o problema semântico (sugerindo alternativas que permitam contorná-lo) [Mongiovi 91] e propor uma solução para resolver simultaneamente ambos os problemas [Cirne 91a].

Esta solução foi denominada algoritmo RPRISM [Cirne 91a]. Ela é baseada em informação semântica e inspira-se tanto na solução apresentada para o problema sintático (algoritmo PRISM [Cendrowska 88]), quanto nas soluções propostas para o problema semântico (algorítmos ID3X [Mongiovi 90], ADEX [Cirne 90] e IDRT [Mongiovi 91]).

1.4. Estrutura da Dissertação

O capítulo 2 é uma breve resenha sobre a área de aquisição de conhecimento. Atenção especial será dada à indução por computador.

O capítulo 3 descreve a família de algoritmos TDIDT.

O capítulo 4 apresenta detalhadamente o problema semântico, bem como dois algoritmos propostos para solucioná-lo: o ADEX e o IDRT. Eles produzem o que chamamos de árvores de decisão semânticas.

O capítulo 5 expõe o problema sintático que é apresentado como sendo o ponto fraco das árvores de decisão (inclusive das semânticas). Além disso, descrevemos o algoritmo PRISM, que soluciona o problema sintático induzindo regras modulares. É também mostrado porque o PRISM não resolve o problema semântico.

O capítulo 6 expõe o RPRISM, algoritmo que resolve ambos os problemas (o sintático e o semântico) simultaneamente.

O capítulo 7 contém aspectos do projeto e da implementação de um protótipo, escrito em Prolog, do sistema AS (Aquisitor Semântico). O AS ajuda na

AC fazendo indução a partir de exemplos e de informações semânticas fornecidas pelo especialista. Os algoritmos ID3, ADEX, IDRT, PRISM e RPRISM estão disponíveis no protótipo.

As conclusões, bem como sugestões para futuros trabalhos, estão no capítulo 8.

2. Aquisição de Conhecimento

Os métodos de aquisição de conhecimento podem ser divididos em cognitivos e automáticos [Boy 87]. Os métodos cognitivos são caracterizados pela presença intensiva do engenheiro de conhecimento, que é encarregado de extrair e formalizar o conhecimento do(s) especialista(s). Os métodos automáticos são aqueles onde a máquina consegue obter, do ambiente que a envolve, o conhecimento necessário para o funcionamento do SBC.

2.1. Métodos Cognitivos

Os métodos cognitivos para AC têm grande semelhança com as técnicas de análise de sistemas tradicionais e exigem do EC o domínio completo de tais técnicas. Além disso, o EC tem que ser um bom conhecedor de IA, deve manter um intenso relacionamento com o(s) especialista(s) e até entender um pouco sobre o domínio. Devido a essas exigências, o processo de AC baseado em métodos cognitivos é lento e dispendioso. De fato, a experiência mostra que, mesmo quando há disponibilidade de especialistas no domínio, um EC consegue acrescentar por dia apenas de uma a quatro regras na BC [Quinlan 86] [Shapiro 87].

Os métodos cognitivos podem ser divididos em quatro grandes classes: entrevistas, *brainwriting*, análise de protocolos e observação direta [Boy 87].

As entrevistas são sessões nas quais o especialista é inquerido pelo EC acerca do domínio em questão. As entrevistas, em geral, são pouco produtivas devido a razões como:

i) O vocabulário usado pelo especialista nas suas explanações geralmente é diferente do usado pelo EC.

ii) Fatores humanos envolvidos no processo como humor, empatia e hostilidade.

iii) Inexistência de uma metodologia "padrão" para conduzir as entrevistas.

iv) Dificuldades encontradas pelo especialista para enunciar seu conhecimento. Estas dificuldades aparecem porque o conhecimento é compilado (o longo processo de aprendizagem faz com que o conhecimento se re-estruture de forma sucinta ao longo do tempo), subjetivo (cada especialista possui uma visão

particular do domínio), volátil (uma afirmação que hoje é aproveitada como conhecimento, amanhã pode deixar de ser aproveitada total ou parcialmente) e formado por várias partes (diversos especialistas detêm, cada um, uma parte do conhecimento).

Vale salientar que, apesar das muitas limitações, as entrevistas continuam sendo o método mais utilizado, até pela falta de alternativas significativamente superiores.

Brainwriting é um método que permite obter conhecimento de um grupo de especialistas. Ele consiste basicamente na definição de regras para discussão entre os especialistas e propõe a substituição da comunicação oral pela escrita [Warfield 71]. As principais vantagens desse método são:

i) A produtividade do grupo aumenta, devido ao trabalho em paralelo.

ii) O domínio de personalidades fortes é eliminado.

iii) O silêncio e a ausência de crítica verbal propiciam uma reflexão franca e uma atmosfera criativa.

iv) As idéias minoritárias não são desprezadas e a responsabilidade pelo trabalho é dividida pelo grupo.

v) O ato de escrever permite evidenciar os mecanismos de análise e resolução de problemas.

Bons resultados têm sido obtidos com esta abordagem, porém ela ainda é demasiada sujeita a fatores humanos.

A análise de protocolos consiste na minuciosa observação do especialista em ação na solução de problemas reais. A partir daí tenta-se formalizar o processo que levou o especialista à solução. A análise de protocolos pode ser retrospectiva (quando o especialista é filmado resolvendo o problema e, posteriormente, é requisitado a explicar seu comportamento) ou concorrente (quando pede-se ao especialista para "pensar em voz alta") [Varejão 91]. O principal problema desta abordagem é que o conhecimento assim obtido é pouco geral.

Os métodos de observação direta são generalizações dos métodos anteriores. A aquisição é feita a partir da observação de de documentos e manuais, entrevistas, estudo de ações e tarefas efetuadas pelo(s) especialista(s), etc. Devido a sua

generalidade, bons resultados já foram conseguidos com este método [Boy 86], a custa, porém, de uma baixa produtividade.

Note que a AC cognitiva não exclui totalmente o computador. Sistemas de edição e teste de consistência da base de conhecimento, tais como o Teiresias [Davis 82], têm sido propostos e utilizados proporcionando ganhos de produtividade para o EC.

2.2. Métodos Automáticos

Uma área também bastante importante da inteligência artificial é a de Aprendizado Automático (AA). Esta área se preocupa com todas as formas pelas quais uma máquina pode captar conhecimentos do ambiente. Obviamente, quando este conhecimento obtido pela máquina puder ser apresentado como uma BC, temos um método automático de aquisição de conhecimento (ou seja, uma técnica de AA usada na AC) [Cirne 91b]. A figura 2.1 mostra a relação entre aprendizado automático e aquisição de conhecimento.

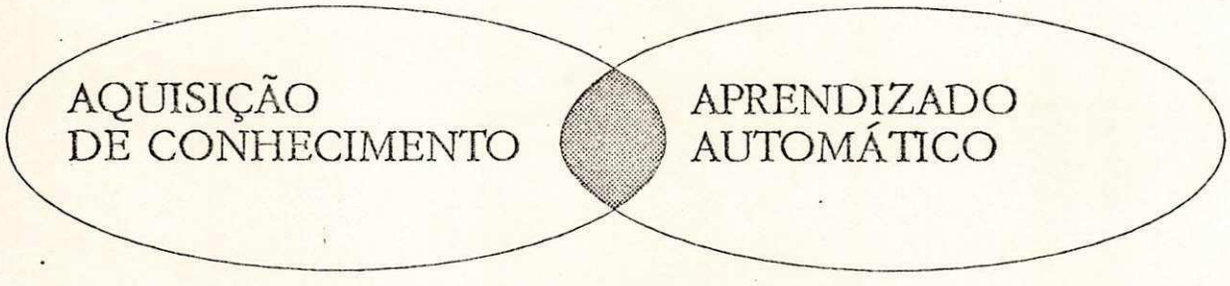

Figura 2.1 -- Relação entre AA e AC

As técnicas de AA mais comumente usadas em AC são [Cirne 91b]:

i) AA em Redes Neurais Artificiais

ii) AA por Analogia

iii) AA a partir de Instrução

iv) AA por Indução

2.2.1. Aprendizado Automático em Redes Neurais Artificiais

As redes neurais artificiais funcionam tentando imitar diretamente a maneira pela qual o cérebro humano trabalha. Então, nada mais natural que as redes neurais aprendam de maneira análoga ao cérebro, isto é, o aprendizado é feito através de alterações na rede [Soucek 89]. A apresentação de padrões entrada/saída, sem nenhuma preocupação em definir o processamento que transforma a entrada na saída, é suficiente para ensinar uma rede.

Em alguns tipos de redes neurais, onde os elementos de processamento (neurônios) representam conceitos, é possível obter uma BC explícita [Machado 90]. Este procedimento caracteriza uma forma de AC automática.

2.2.2. Aprendizado Automático por Analogia

A técnica de AA por Analogia se baseia na transferência de conceitos de um objeto (origem) para outro (meta), a partir da informação que eles são semelhantes. Obviamente, é necessário muita heurística para saber quais conceitos devem ser utilizados dentre os disponíveis. Um exemplo do uso desta técnica pode ser encontrado em um sistema proposto por Winston que representa os objetos por *frames* e os conceitos por *slots* [Winston 79]. A principal heurística do sistema consiste em usar os conceitos que assumem valores extremos no objeto origem.

2.2.3. Aprendizado Automático a partir de Instrução

AA a partir de Instrução é uma "imitação" da maneira pela qual um aluno aprende de seu instrutor. Ela consiste na transformação de conhecimento e heurísticas expressos em uma linguagem de entrada para uma linguagem usada internamente. Um exemplo de uma ferramenta com essas caracteristicas é o ETS (*Expertise Transfer System*) [Boose 84]. O ETS é baseado em teorias psicológicas (*Personal Construct Theory*). Esta ferramenta descreve o relacionamento entre os elementos de classificação do conhecimento (i. e., os conceitos a serem aprendidos) e suas características através de redes de repertório (*rating grid*). Posteriormente, o ETS evoluiu para o sistema AQUINAS [Boose 87]. AQUINAS permite lidar com o conhecimento hierarquicamente estruturado. Esses sistemas entrevistam exaustivamente o perito e praticamente dispensam a presença do EC. Entretanto, não fazem uso direto de exemplos do domínio analisado.

2.2.4. Apendizado Automático por Indução

Dentre todas as técnicas de aprendizado automático usadas em aquisição de conhecimento, a indução por computador tem sido a mais explorada.

Não há uma definição universalmente aceita de indução. Entretanto, todas as definições propostas encaram indução como uma forma de raciocínio não-dedutivo que obtém uma explicação geral, chamada conclusão, de um conjunto de fatos concretos observados no mundo real, chamado Conjunto de Treinamento (CT). Como raciocínio não-dedutivo, a validade do CT de uma inferência indutiva não garante a validade da conclusão. Contudo, as conclusões indutivas são mais prováveis que uma assertiva qualquer e, por isso, a indução tem grande importância, principalmente para as ciências experimentais. Uma boa resenha sobre os problemas da definição de indução e suas implicações filosóficas pode ser encontrada em [Boman 89].

Vale a pena ressaltar que em alguns casos uma conclusão indutiva pode ter sua validade assegurada. Isso acontece quando a conclusão for obtida de um conjunto de treinamento que descreva exaustivamente todos os casos individuais. Desejamos, porém, soluções que não precisem de todos os dados possíveis para chegar a algum resultado porque, mesmo em problemas "de brinquedo", esta quantidade de informação pode ser intratável.

Uma boa definição de indução, embora muito geral, foi apresentada por Genesereth e Nilsson [Genesereth 86]. Nela, as premissas são divididas em uma teoria base T e em um conjunto de treinamento C, que será generalizado. Esta divisão é feita de tal forma que T não implique logicamente C, i. e., T |=/ C.

Dada uma teoria base T e um CT C, dizemos que a assertiva i é uma conclusão indutiva de T e C (denotado por T U C |< i) se e somente se:

i) A conclusão indutiva for consistente com a teoria base e com o CT, i. e., T U C |=/ ~i.

ii) A conclusão indutiva explica o CT, i. e., T U {i} |= C.

Um outro tipo de inferência não-dedutiva é a abdução. A abdução consiste numa regra de inferência análoga ao Modus Ponens, qual seja: "se a -> b e b então a" [Charniak 85]. A abdução é um caso particular de indução fazendo T = {a -> b}, C = {b} e i = a e, portanto, não será considerada em particular. Porém, alguns trabalhos em AA definem indução de maneira menos geral de forma a não englobar abdução [Pingand 89].

2.2.4.1. Tipos de Indução

No aprendizado usando indução, há dois tipos de generalização: parte-global e instância-classe (veja a figura abaixo).

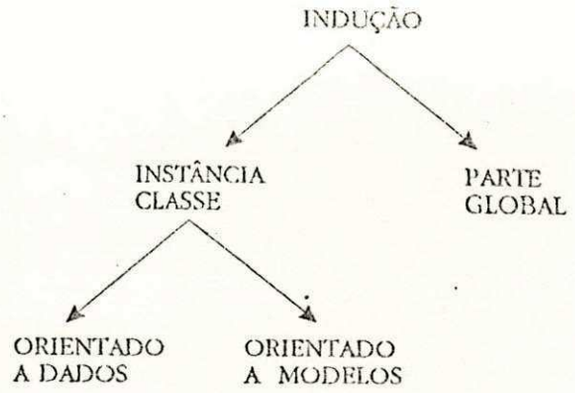

Figura 2.2 -- Classificação dos Sistemas de Indução

Na generalização parte-global, o objetivo é conceituar um objeto a partir de exemplos que representam partes do objeto. O sistema SPARC [Dietterich 86] se utiliza de indução parte-global para gerar regras.

Instância-classe caracteriza um sistema de aprendizado que induz uma descrição geral de uma classe quando informamos uma coleção de instâncias (ou exemplos) daquela classe. As instâncias podem ser representações de objetos físicos, sons, imagens, ações ou sintomas, por exemplo.

A generalização instância-classe permite duas formas de se chegar às descrições gerais de um objeto: orientada a dados (*data-driven*) e orientada a modelos (*model-driven*) [El-Khomi 88]. Nos métodos orientados a dados, o conjunto de treinamento não possui, *a priori*, nenhuma descrição geral das classes envolvidas. Já nos métodos orientados a modelos, o conjunto de treinamento passou previamente por um processo de generalização. Este processo anterior pode ser o fornecimento de regras de uma forma cognitiva. O programa INDUCE [Dietterich 81] se utiliza de um método orientado a modelos para generalizar regras.

2.2.4.2. Árvores de Decisão

Grande parte das soluções propostas para fazer indução por computador geram como resultado uma Árvore de Decisão (AD), um formalismo bastante conhecido e utilizado para representar conhecimento classificatório. Uma AD é uma árvore onde as folhas são os elementos de classificação (i. e., as classes), os nós não-terminais representam os atributos dos fatos concretos observados no mundo real, e os ramos denotam os valores destes atributos. Ela é construída de forma que, ao percorrer o caminho da raiz para uma das folhas, identificamos condições (pares ATRIBUTO = valor) suficientes para classificar a folha.

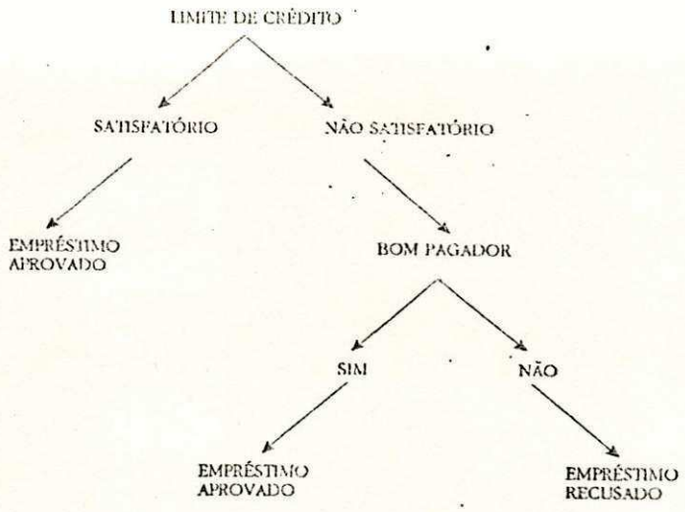

Figura 2.3 -- Exemplo de uma AD

Por exemplo, na AD da figura 2.3, as condições LIMITE-DE-CRÉDITO = não-satisfatório e BOM-PAGADOR = sim são suficientes para afirmar que o pedido de empréstimo será aprovado.

Há várias técnicas de aprendizado automático que permitem a construção de ADs a partir de conjuntos de treinamento. Na indução de árvores de decisão, um elemento do CT geralmemte representa um exemplo ou evento do mundo real e é descrito como uma coleção de condições, a qual está associada uma classe. A maior parte dos algoritmos de indução que são baseados em ADs fazem parte da família TDIDT [Quinlan 86]. Estes algorítmos são caracterizados por construir a AD da raiz para as folhas. Eles fazem a divisão sucessiva do conjunto de treinamento usando o melhor (por exemplo, o mais informativo) atributo como divisor. O processo é repetido recursivamente até que um teste de parada seja satisfeito (tipicamente, até que todos os exemplos da folha sejam de uma única classe) [Van de Vilde 89].

Mapeamento em Regras

É comum mapear árvores de decisão em regras, para possibilitar o uso de ferramentas baseadas em regras para construção de SBCs. Além do mais, o uso de regras facilita a explanação, pois não é mais necessário mostrar toda a AD quando uma explicação for solicitada. Quando utilizamos regras, apenas aquelas utilizadas são exibidas.

O mapeamento de uma AD em regras pode ser feito de forma trivial. São construídas tantas regras quantas folhas houverem na árvore. Cada folha é o consequente de uma regra que tem como antecedente a conjunção dos pares NÓ = ramo encontrados no caminho que vai da raiz à folha em questão. Note que nem

sempre há um mapeamento que transforme um conjunto de regras em uma **AD** (maiores detalhes no capítulo 6).

Neste trabalho, frequentemente usaremos regras mapeadas de **ADs** porque, devido a possibilidade de nomea-las, é mais fácil desenvolver nossa argumentação com as regras do que com a árvore propriamente dita. Entretanto, a argumentação não perde generalidade, pois as regras obtidas trivialmente de uma dada **AD** nada mais são do que uma outra forma de representar a árvore.

Algoritmos que Geram Árvores de Decisão

Há duas abordagens usadas pelos métodos de indução orientados a dados que usam **ADs**: a incremental e a TDIDT.

Os algoritmos incrementais funcionam transformando a **AD**, inicialmente vazia, a cada exemplo obtido. Como exemplos de métodos incrementais temos o ID4 [Schlimmer 86], o ID5 [Utgoff 88], o ID5R [Utgoff 89] e o IDL [Van de Vilde 89].

Para os algoritmos TDIDT, é necessário que todos os exemplos estejam disponíveis *a priori*. Eles geram a **AD** recursivamente, da raiz para as folhas, escolhendo, a cada passo, o "melhor" atributo (por exemplo, o mais informativo). Os algoritmos TDIDT mais famosos são o ID3 [Quinlan 83], o C4 [Quinlan 87] e o CART [Breiman 84].

O problema sintático (veja capítulo 4) provêm do uso de **ADs** como forma de representar o conhecimento adquirido. Portanto, tanto os algoritmos incrementais quanto os TDIDT dele sofrem, embora a forma pela qual eles constroem a árvore seja diferente. Já o problema semântico, pode ser tão bem resolvido tanto pela abordagem TDIDT quanto pela incremental. Devido a isto, por simplicidade, trataremos apenas dos algoritmos TDIDT no restante desta dissertação. Todavia, para tornar este trabalho mais completo e informar melhor o leitor interessado em algoritmos incrementais, o apêndice A traz a descrição de um desses algoritmos (o IDL).

3. A Família de Algoritmos TDIDT

Os diversos algoritmos da família TDIDT constroem **ADs** para tarefas de classificação. Estas árvores são construidas a partir da raiz, descendo até as folhas. Daí o nome TDIDT.

O conhecimento representado nas **ADs** permite classificar objetos ou situações em classes. De posse de vários exemplos que descrevem situações do mundo real, o **EC** obtém o conjunto de treinamento usando parte dos exemplos. O **CT** é usado para gerar a **AD**. Os exemplos restantes constituem o conjunto de teste, que é usado para avaliar a **AD** obtida.

O conjunto de treinamento pode ter diferentes origens. Ele pode vir de uma base de dados existente (por exemplo, um fichário médico com o quadro histórico de vários pacientes). Alternativamente, os exemplos podem ser um conjunto tutorial cuidadosamente preparado por um especialista (por exemplo, os casos clássicos de um determinado domínio).

Cada exemplo está definido por um conjunto de condições (pares ATRIBUTO = valor) e o valor da classe correspondente. Cada atributo a_i pode assumir um número finito de valores v_{i1}, ..., v_{ini}. Obviamente, o número de atributos também é finito.

Cada nó não-terminal de uma **AD**, representa o teste de um atributo e tem tantos filhos quanto forem os valores que o atributo representado puder assumir. Os ramos que ligam o atributo representado aos seus filhos denotam esses valores. Cada folha é rotulada com a classe que descreve os exemplos que contém os pares ATRIBUTO = valor do caminho que liga a folha à raiz.

3.1. Procedimento Geral

Os algoritmos TDIDT começam com uma **AD** vazia, que é expandida gradualmente até classificar todos os exemplos do **CT**. O procedimento geral é o seguinte [Castiñeira 90]:

DADOS
um conjunto de treinamento **D**;
uma condição de parada **t(D)**; e
uma função de avaliação **aval(D, a)**
SE todas as instâncias em **D** satisfazem a condição de término **t(D)** então
RETORNE o valor da classe
SENÃO
PARA CADA atributo **a**, CALCULE o valor da função **aval(D, a)**
SEJA a_m o atributo que possui o melhor valor de **aval(D, a)**
DIVIDA o conjunto **D** em subconjuntos com valores de atributo $v_{m1}, ..., v_{m,nm}$ usando o atributo a_m
APLIQUE recursivamente o algoritmo a cada subconjunto de treinamento D_k $(1 \leq k \leq n_m)$

O critério de parada t pode ser definido tanto para construir **ADs** que classificam todos os elementos do conjunto de treinamento em domínios determinísticos, quanto para decidir pela não expansão da árvore quando os exemplos fornecidos forem insuficientes. A função de avaliação **aval** denota quão bom é um atributo, no sentido do atributo levar à menor **AD** possível. Várias funções de avaliação foram propostas, visando obter **AD**s tão pequenas quanto possível. Temos o cálculo de entropia, a qui-quadrado, a estatística G, o índice de diversidade GINI, a medida proporcional de ganho [Mingers 89a]. Dentre elas, a mais utilizada é o cálculo de entropia [Klir 88].

3.2. O Algoritmo ID3

O algoritmo mais conhecido da família TDIDT é o ID3 (*Induction of Decision Trees*) [Quinlan 83]. Além disso, ele é um dos precursores da família e é um algoritmo simples. Devido a estas características e tendo em vista que o raciocínio aqui desenvolvido não é dependente das particularidades de nenhum algoritmo TDIDT, o ID3 será o único algoritmo TDIDT descrito no corpo desta dissertação. Entretanto, para fornecer mais subsídios ao leitor interessado nos algoritmos TDIDT, o apêndice B apresenta o algoritmo C4, que foi derivado do ID3 e lhe é superior, porque provê mecanismos de poda de árvores [Quinlan 87].

O ID3 encerra a construção da **AD** quando todos os elementos do **CT** pertencerem à mesma classe. A função de avaliação usada baseia-se no cálculo de entropia. A entropia mede o grau de incerteza de um conjunto de eventos. Assim, quanto menor a entropia de um dado conjunto de eventos, menor será seu grau de incerteza. O objetivo do ID3 é, portanto, obter a entropia nula, i. e., a eliminação da incerteza.

A função de avaliação H de um atributo a é a média ponderada das entropias das partições do conjunto de treinamento segundo a (ou seja, a cada valor de a corresponde uma partição). Em fórmulas, temos:

$$H(a) = \sum_{i=1}^{nv} \frac{n_{Vi}}{n} H(a=v_i)$$

onde: n_V é o número de valores do atributo a
n_{Vi} é o número de exemplos em que o atributo a possui o valor v_i
n é o número de exemplos no conjunto de treinamento

Por sua vez, a entropia da partição do CT formada por todos os casos em que a possui o valor v_i é dada por:

$$H(a=v_i) = \sum_{j=1}^{m} -p(c_j|v_i)\log_2 p(c_j|v_i)$$

onde: m é o número de classes no conjunto de treinamento
$p(c_j|v_i)$ é a probabilidade de se ter, no conjunto de treinamento, um exemplo com classe c_j, dado que a assume valor v_i.

Por exemplo, para o CT da tabela 3.1, o cálculo da entropia do atributo DIABÉTICO seria feito como segue:

$H(\text{DIABÉTICO} = \text{sim}) = -1/4.\log_2 1/4 - 3/4.\log_2 3/4 = 0,8113$
$H(\text{DIABÉTICO} = \text{não}) = -3/4.\log_2 3/4 - 1/4.\log_2 1/4 = 0,8113$
$H(\text{DIABÉTICO}) = 4/8.0,8113 + 4/8.0,8113 = 0,8113$

	COME	VEGETARIANO	IDADE	DIABÉTICO	CLASSE
ex1	pouco	sim	velho	sim	MAGRO
ex2	médio	sim	velho	não	MAGRO
ex3	muito	não	velho	sim	GORDO
ex4	pouco	não	velho	não	MAGRO
ex5	médio	não	jovem	sim	GORDO
ex6	pouco	sim	jovem	não	MAGRO
ex7	muito	não	velho	não	GORDO
ex8	médio	não	jovem	sim	GORDO

Tabela 3.1 - Conjunto de Treinamento

A AD gerada pelo ID3, correspondente ao conjunto de treinamento da tabela 3.1, está na figura 3.2. Abaixo de cada classe estão os números dos exemplos que foram mapeados pelo ramo que começa na raiz e vai até a classe.

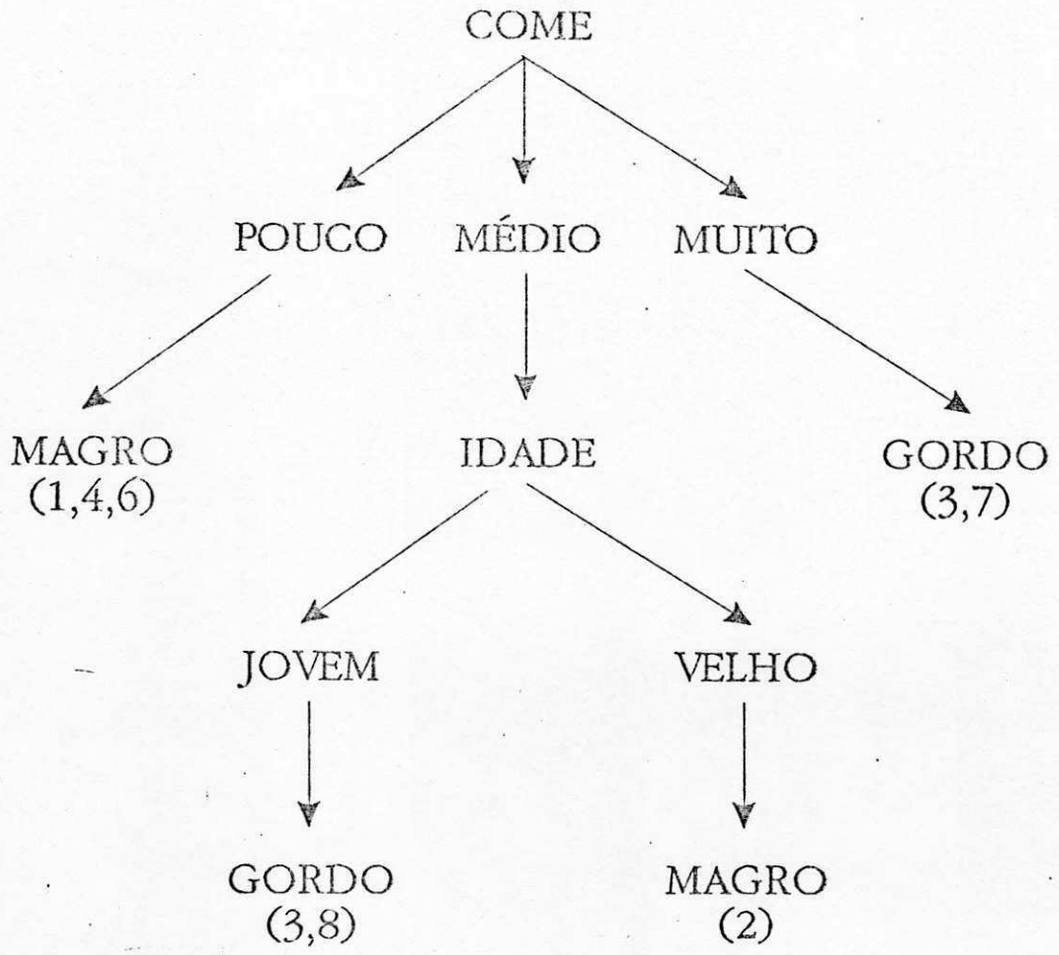

Figura 3.2 - AD gerada pelo ID3

As regras correspondentes à árvore, utilizando o mapeamento trivial, são:

r1) se COME = pouco **então MAGRO**
r2) se COME = médio e IDADE = jovem **então GORDO**
r3) se COME = médio e IDADE = velho **então MAGRO**
r4) se COME = muito **então GORDO**

A vantagem de usar uma boa função de avaliação está no tamanho da **AD** obtida. Como comparação, observe a árvore da figura 3.3, que foi gerada, usando como função de avaliação uma função aleatória, a partir dos mesmos dados de entrada usados pelo ID3 para gerar a **AD** acima (i. e., o **CT** da tabela 3.1).

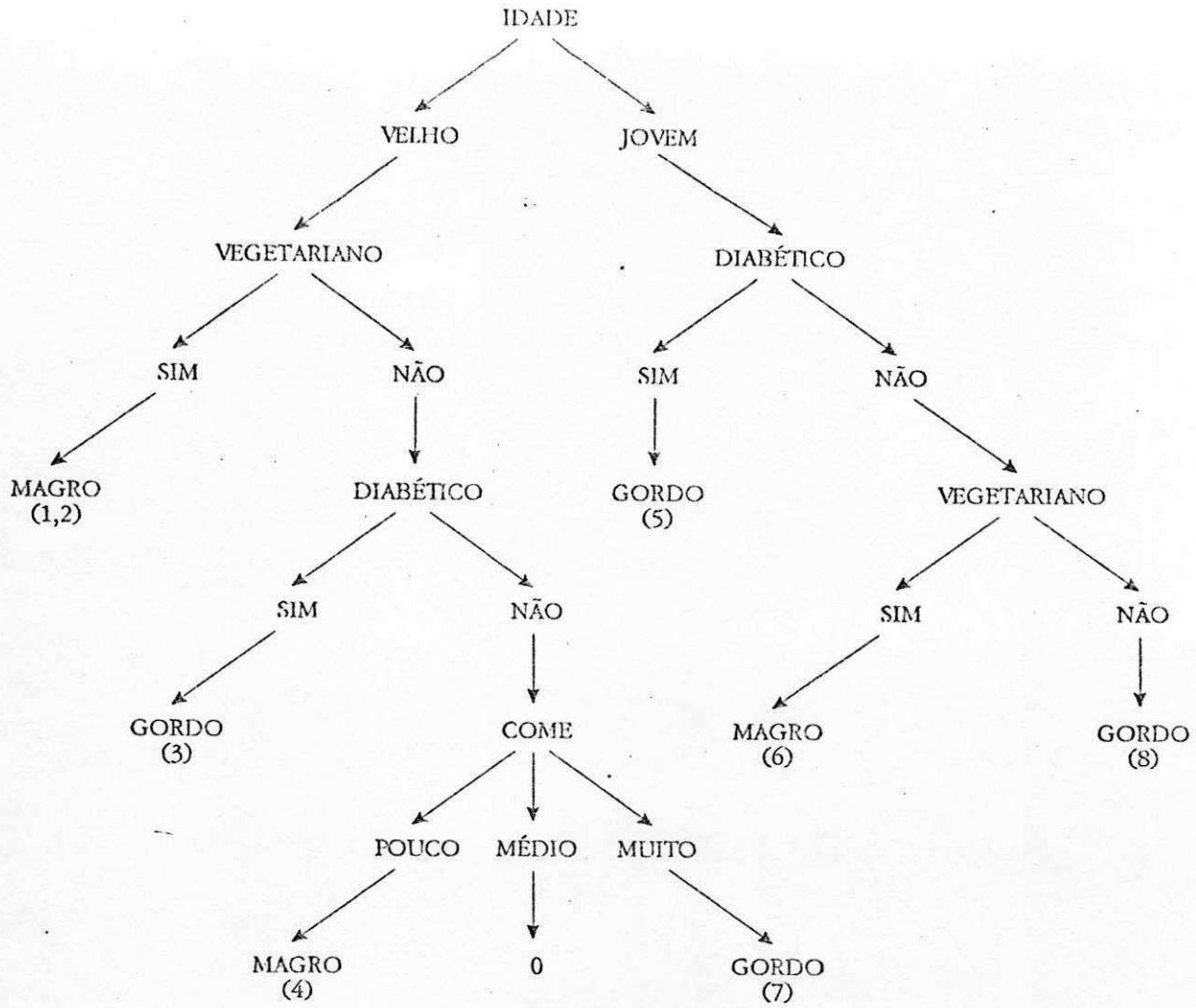

Figura 3.3 - AD gerada aleatoriamente

3.3. Análise dos Algoritmos TDIDT

De posse das informações apresentadas neste capítulo sobre os algoritmos TDIDT, já podemos levantar as principais vantagens e desvantagens desses algoritmos.

Entre as vantagens destacamos:

i) A participação do especialista e do EC na construção da BC fica minimizada.

ii) A BC é gerada a partir de exemplos ocorridos no dia-a-dia do(s) especialistas(s). Os exemplos são fáceis de serem obtidos (em relação à aquisição cognitiva) e permitem tratamento probabilístico.

iii) A BC assim obtida é livre de inconsistência [Gascuel 88], eliminando a necessidade de verificação de sua coerência [Kguyen 87].

iv) O processo não se limita a áreas específicas, isto é, pode, a princípio, ser utilizado em qualquer domínio.

Apesar do sucesso alcançado, estes métodos apresentam algumas limitações de cunho prático. As mais conhecidas são:

i) São aplicados somente em domínios onde tanto os elementos de classificação como os valores de suas características (atributos) são mutuamente exclusivos.

ii) Não permitem manipular atributos com valores contínuos.

iii) Falham na sua tentativa de construir a AD quando, entre os exemplos, existem contra-exemplos, caracterizando um domínio não determinístico.

iv) Não geram regras com incerteza.

4. O Problema Semântico e sua Solução via Árvores de Decisão

4.1. O Problema Semântico

As ADs geradas pelos algorítmos da família TDIDT são estruturas despidas de semântica. Elas apenas mapeiam, de forma reduzida, o CT usado como entrada. Devido a este fato, estes algoritmos podem gerar, em certas circunstâncias, regras sem significado que na prática são inúteis. Este fenômeno é observado principalmente na resolução de problemas reais (i. e, problemas em domínios que têm grande número de classes e condições).

Isto ocorre porque, em problemas reais, o CT geralmente só explicita uma minúscula parte de todos os exemplos possíveis dentro do domínio. Assim, podemos ter, por coincidência, que uma dada classe seja determinada por uma condição irrelevante (para a classe). Uma circunstância em que isto frequentemente acontece é quando o conjunto de exemplos contém casos raros (casos caracterizados por condições que ocorrem apenas em pouquíssimos exemplos). Vale ressalvar que o conjunto de treinamento é pouco abrangente, não por vontade do EC ou de algum elemento humano envolvido no processo de aquisiçao, mas devido ao tamanho do problema em si.

O problema semântico foi descoberto durante a validação do APREND [Gomes 88] [Gomes 89], sistema de AC baseado no ID3. O objetivo era construir um SE em ginecologia, o que representava um domínio com 18 classes e 46 atributos, cada um com, em média, 3,47 valores. O CT foi obtido de um fichário médico. Para nossa surpresa, entre outras aberrações, o APREND (leia-se, ID3) concluiu que a condição IDADE = criança era suficiente para diagnosticar VAGINITE (sic!). Isto aconteceu porque no CT, por coincidência, havia poucas crianças e todas elas estavam com vaginite (já que os dados foram obtidos de fichário médico). Portanto, o atributo IDADE era o de menor entropia. Além disso, o sistema APREND encerrou neste ponto a construção do ramo correspondente ao caso, já que todos os exemplos ficam classificados apenas com IDADE = criança. O caso raro ficou, portanto, mapeado numa regra inútil, uma vez que a premissa da regra acima ("a paciente é criança"), na prática é irrelevante para concluir se a paciente está ou não com vaginite. Vale salientar que denominamos as regras com este tipo de problema de "inúteis" porque, embora semânticamente erradas, elas de fato mapeiam o CT.

Este tipo de problema pode aparecer até mesmo em casos "de brinquedo". Observe, por exemplo, a AD da figura 3.2, que foi gerada pelo ID3. As regras r2 e r3, embora não mapeando casos raros, têm, como se pode ver, pouca utilidade prática e

são ditas inúteis. Isto porque DIABÉTICO e VEGETARIANO são fatores muito mais indicativos do que IDADE, para concluirmos se alguém é magro ou gordo. Além do mais, o atributo COME não é importante para a conclusão com o valor médio.

Na verdade, há também uma outra justificativa para esse comportamento apresentado pelos algoritmos TDIDT. Em casos reais, apenas uma parcela das condições é relevante para a determinação de uma dada classe (veja figura 4.1).

Por exemplo, a condição NACIONALIDADE = brasileira não é, de forma alguma, tão importante quanto TEMPERATURA = alta para determinar a classe FEBRE. Porém, os algoritmos TDIDT consideram todas as condições potencialmente iguais para a conclusão de uma determinada classe. É devido a este comportamento (não considerar as condições de forma diferenciada, tendo em vista o que elas significam para a classe em determinação) que o problema apresentado nesta seção foi chamado de problema semântico.

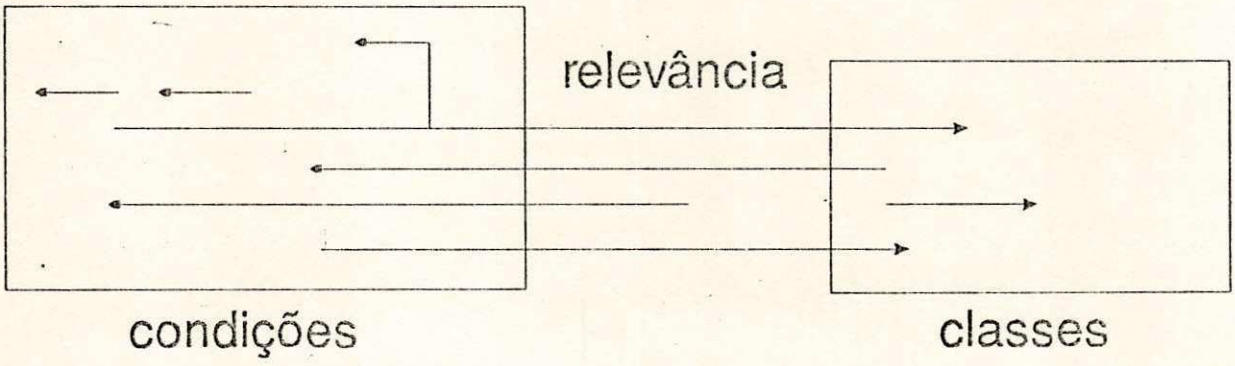

Figura 4.1 - Relação de relevância entre as condições
e as classes de um domínio

4.2. Árvores de Decisão Semânticas

Os algoritmos de indução de **ADs** da família TDIDT apresentam um problema, pois consideram os dados apenas do ponto de vista sintático. Uma solução para este problema consiste no uso de informações semânticas no processo de obtenção da **AD**. Assim produziríamos Árvores de Decisão Semânticas (**ADSs**), i. e., árvores de decisão cuja construção observou a relevância com que cada condição determina uma dada classe.

4.2.1. Matriz de Relevância

É necessário, portanto, obter de alguma forma o conhecimento a respeito do relacionamento semântico entre as condições e as classes. A fonte mais natural para obtenção deste tipo de informação é o especialista. Nós definimos uma estrutura chamada Matriz de Relevância (**MR**), que permite obter e representar este conhecimento do especialista.

Uma **MR** para um dado domínio é o relacionamento de relevância entre os elementos de classificação e os atributos deste domínio. Em uma **MR** a linha i representa o atributo a_i e a coluna j o elemento de classificação E_j. Um elemento r_{ij} de uma **MR** indica o grau de relevância do atributo a_i para a classificação de E_j. Este grau é dado pelo conjunto de valores do atributo a_i que são relevantes para conclusão de E_j. O elemento r_{ij} é, na verdade, um subconjunto do conjunto de valores do atributo a_i. Se r_{ij} for vazio, significa que o atributo a_i é irrelevante na classificação de E_j. Ao passo que, r_{ij} igual ao conjunto de valores de a_i expressa que a_i é totalmente relevante na classificação de E_j. Uma **MR** é dita completa se todos os atributos forem totalmente releventes para todas as classes. Uma **MR** é dita vazia se todos os seus elementos forem iguais ao conjunto vazio.

Uma **MR** é semelhante à uma rede de repertório (*rating grid*) dos sistemas tipo ETS. A grande diferença é que enquanto numa rede de repertório o relacionamento é total (todos os elementos de classificação se relacionam com todas as características), numa **MR** apenas os relacionamentos tidos como relevantes são considerados. Isto minimiza a participação do especialista no processo de aquisição de conhecimento.

Por exemplo, uma **MR** para o domínio do **CT** da tabela 3.1 (este domínio é caracterizado pelos atributos COME (que assume os valores pouco, médio ou muito), DIABÉTICO (valores sim ou não), IDADE (valores jovem ou velho) e VEGETARIANO (valores sim ou não), e pelas classes **MAGRO** e **GORDO**) aparece na figura abaixo.

	MAGRO	GORDO
COME	{pouco}	{muito}
DIABÉTICO	{não}	{sim}
IDADE	Ø	Ø
VEGETARIANO	{sim}	{não}

Tabela 4.2 - Exemplo de uma MR

A tabela 4.2 mostra que:

i) O atributo COME é relevante para classificar **MAGRO** no valor pouco, e, no valor muito, para classificar **GORDO**.

ii) O atributo DIABÉTICO é relevante para classificar **GORDO** no valor sim e **MAGRO** no valor não.

iii) O atributo IDADE é irrelevante para classificar **MAGRO** e **GORDO** (casos deste tipo alertam ao **EC** que o atributo **IDADE** possivelmente deve ser eliminado do **CT**).

iv) O atributo VEGETARIANO é relevante para classificar **GORDO** no valor não e **MAGRO** no valor sim.

Note que a **MR** não é uma técnica de **AC** na acepção pura da palavra, pois não há nela informação suficiente para construir uma **BC**. Entretanto, ela é muito fácil de ser construída pelo especialista. Por exemplo, um especialista gastou apenas 2 horas na construção de uma **MR** para o domínio ginecológico onde foi detectado o problema semântico, que tem 18 classes e 46 atributos (com, em média, 3,47 valores por atributo).

A idéia é repartir o trabalho: deixamos uma pequena parte com o especialista e o restante com a máquina, de forma a obter um bom resultado no menor tempo possível. O ideal seria eliminar a participação do especialista. Todavia, as soluções até hoje propostas não foram boas ou gerais o suficiente para produzir resultados que dispensem a aprovação final do especialista. Esta dificuldade de construir métodos automáticos de **AC** realmente gerais e eficazes se deve à grande complexidade do processo de aquisição.

4.2.2. Expansão Semântica de Árvores de Decisão

O primeiro algoritmo proposto para solucionar o problema semântico foi o ID3X. Ele expande a árvore gerada pelo ID3, com a ajuda da **MR**, de forma a colocar condições relevantes naqueles ramos em que todas as condições são irrelevantes. O ID3X foi posteriormente generalizado para expandir **ADs** geradas por qualquer algoritmo TDIDT. Esta generalização foi batizada de ADEX. Nesta seção apenas o ADEX será descrito, pois o ID3X nada mais é que o ADEX aplicado ao ID3.

Usando o ADEX, o resultado é a expansão da **AD** gerada por um algoritmo TDIDT. A finalidade dessa expansão é colocar a semântica obtida do especialista (contida na **MR**) na árvore. Pretendemos, desta forma, evitar que se tenha uma regra onde toda a premissa é irrelevante para a conclusão, isto é, evitar regras inúteis.

A expansão é feita de modo a colocar pelo menos uma condição relevante em todo caminho que vai da raiz a uma folha qualquer. Ou seja, toda regra terá pelo menos um componente relevante em sua premissa. Abaixo, segue o algoritmo ADEX, que expande uma **AD** gerada por um algoritmo TDIDT em uma **ADS**.

```
adex(AD)
    PARA TODA folha f (composta pelo elemento de classificação E e pelo
    conjunto de exemplos D) da AD
        tentar_expandir(f)
tentar_expandir(f)
    SE nenhuma das condições pertencentes ao caminho(raiz,f) é relevante ENTÃO
        SEJA ai um atributo não pertencente ao caminho(raiz,f)
        SEJA Vi = {Vi1,Vi2,...,Vin} o conjunto de valores que são relevantes
        para ai concluir E
        SEJA A = {a1,a2,...,ak} o conjunto de todos os ai's tais que existe
        um exemplo pertencente a E com o valorde ai pertencente a Vi
        SE A não é vazio ENTÃO
            SEJA am o atributo de melhor função de avaliação pertencente a A
            SUBSTITUA f pelo nó não-terminal am ligado pelos ramos Vm1,
            Vm2, ..., Vmn e * às folhas f1, f2, ..., fn e f*, respectivamente
            ASSOCIE a cada uma destas folhas o elemento E
            REDISTRIBUA o conjunto de exemplos D entre elas
            tenta_expandir(f*)
        SENÃO marca f como inútil
```

Figura 4.3 - Algoritmo ADEX

No algoritmo ADEX, o símbolo *, filho do atributo a_m, representa todos os valores não relevantes de a_m. A aplicação do ADEX à árvore da figura 3.2, usando a **MR** da tabela 4.2, está na figura abaixo.

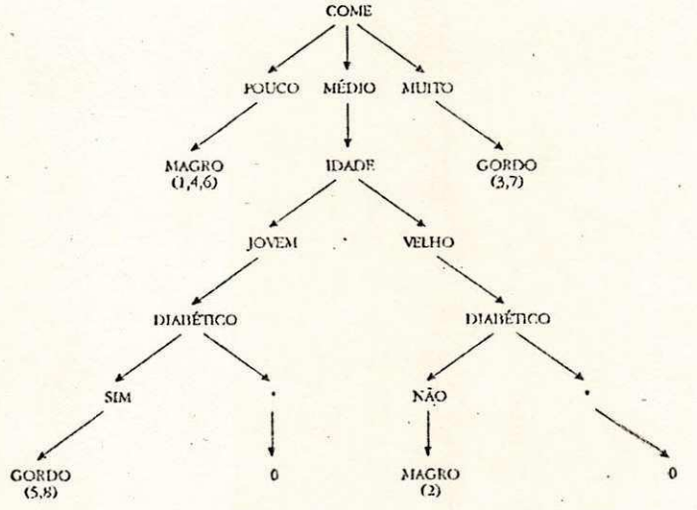

Figura 4.4 - AD expandida pelo ADEX a partir de AD ID3

As regras correspondentes à árvore acima são:

s1) se COME = pouco então **MAGRO**
s2) se COME = médio e IDADE = jovem e DIABÉTICO = sim então **GORDO**
s3) se COME = médio e IDADE = velho e DIABÉTICO = não então **MAGRO**
s4) se COME = muito então **GORDO**

Os arcos que têm como valor o * não aparecem nas regras, já que eles não acrescentam nenhuma relevância às regras originalmente obtidas. Ou seja, o símbolo * é transparente para o gerador de regras.

Da comparação das regras originais com as regras expandidas pelo ADEX, notamos que a regra r2 inicialmente tida como inútil foi substituída, com o uso do algoritmo ADEX, pela regra útil s2. Além disso, a outra regra inútil r3 foi transformada na regra útil s3. A conclusão de que as regras s2 e s3 são úteis se deve ao fato de que, na prática, o atributo DIABÉTICO pertencente a estas regras, é bastante significativo para distinguir entre **MAGRO** e **GORDO**.

Embora no exemplo apresentado todas as regras inúteis foram substituídas por regras úteis, na prática isto nem sempre acontece [Cirne 90] [Mongiovi 90]. Entretanto, isso não é devido à uma limitação do ADEX e sim à pouca informação que eventualmente possa estar contida na **MR**. Mesmo quando o ADEX não retira regras inúteis seu uso é recompensador, pois as regras inúteis ficam marcadas de forma que quando o especialista for revisar o banco de regras obtido, pode concentrar sua atenção sobre elas.

Até nos casos extremos, a **MR** completa ou vazia, o ADEX ainda produz resultados coerentes. Em ambos os casos ele não muda a árvore original. Se a **MR** for completa, concluímos que todas as regras são úteis (já que tudo é relevante). Ao passo que, se a **MR** for vazia, obtemos que todas as regras são inúteis (como nada é relevante, tudo é inútil).

4.2.3. Construção Direta de Árvores de Decisão Semânticas

Apesar da expansão semântica de uma árvore TDIDT melhorar a qualidade desta árvore, colocando condições relevantes nas premissas das regras, ainda restam alguns problemas. Como a expansão só faz acrescentar nós à árvore, esta pode assumir proporções inadequadas. Além disso, só é garantida uma condição relevante na premissa, mesmo quando esta tem um grande número de condições.

Para contornar os problemas apresentados pela expansão, propomos a construção direta da **AD** levando em consideração não só o aspecto sintático como também o aspecto semântico. Esta construção será guiada por uma Função de Avaliação Pragmática (**FAP**) que englobe ambos os aspectos. Este novo algoritmo (que utiliza a **FAP** como função de avaliação) é denominado IDRT (*Induction of Decision Relevant Trees*).

A **FAP** aqui proposta é definida de forma a ponderar a informação retirada do **CT** (aspecto sintático) com a informação obtida da **MR** (aspecto semântico). Esta ponderação será variável, permitindo ao usuário enfatizar, de acordo com a necessidade, o aspecto desejado (sintático ou semântico). O principal fator que influencia a escolha da ponderação é o grau de confiança que se tem na **MR**. A **FAP** aqui definida implica que a construção da **AD** será feita utilizando primeiro os atributos de **FAP** mais alta, i. e., o melhor atributo a expandir será aquele de maior **FAP**. Portanto, para um dado atributo a_i, o valor da **FAP m** foi definido como:

$$m(a_i) = p.N(a_i) + (1 - p).IR(a_i)$$

onde: **p** é o fator de ponderação, sendo $0 \leq p \leq 1$
N(a_i) é a função de avaliação sintática normalizada
IR(a_i) é a intensidade de relevância do atributo a_i

Na ausência de uma melhor indicação para compor uma **FAP** equilibrada, um valor razoável para **p** é 1/2.

N(a_i) é uma função de avaliação convencional (i. e., uma função de avaliação sintática) normalizada para o intervalo [0, 1]. A normalização deverá ser feita de forma que os "melhores" atributos obtenham os valores mais altos. Por exemplo, se a função de avaliação convencional considerada for a entropia **ent**, uma boa normalização seria:

$$N(a_i) = \exp(-ent(a_i))$$

onde: **exp** denota a exponenciação neperiana
ent denota a função entropia

Antes de definir **IR**(a_i), algumas outras definições se fazem necessárias. Consideremos, então:

n_V como sendo o número de valores do atributo a_i
n_E como sendo o número de elementos de classificação
$IR_V(a_i=v_j)$ como sendo a intensidade de relevância do valor v_j do atributo a_i

MR[l,c] como sendo o elemento da Matriz de Relevância que está na linha l e coluna c

Podemos agora definir a intensidade de relevância do atributo a_i como segue:

$$IR(a_i) = (1/n_V) \cdot \sum_{j=1}^{nv} IR_V(a_i=v_j)$$

$$\text{onde: } IR_V(a_i=v_j) = (1/n_E) \cdot \sum_{k=1}^{nE} f(a_i,v_j,E_k)$$

$$\text{sendo: } f(a_i,v_j,E_k) = \begin{cases} 1, \text{ se } v_j \text{ pertence a } MR[i,k] \\ 0, \text{ caso contrário} \end{cases}$$

A partir dos mesmos dados de entrada utilizados no exemplo com o ADEX (**CT** da tabela 3.1 e **MR** da tabela 4.2), o IDRT, usando **exp(-ent(a_i))** como função de avaliação sintática normalizada, gera a **AD** mostrada na figura abaixo.

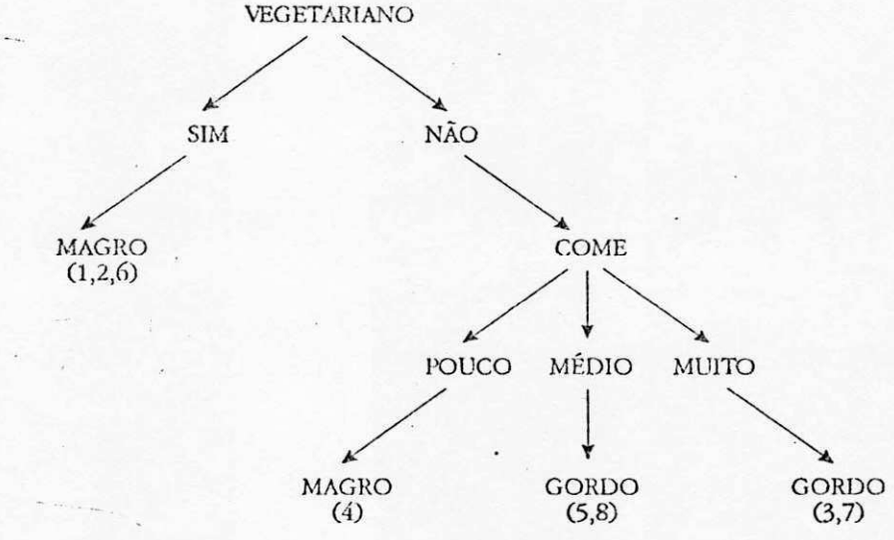

Figura 4.5 - **AD** gerada pelo IDRT usando entropia normalizada

A **AD** da figura 4.5, por sua vez, pode ser mapeada no seguinte conjunto de regras:

t_1) se VEGETARIANO = sim **então MAGRO**
t_2) se VEGETARIANO = não e COME = pouco **então MAGRO**
t_3) se VEGETARIANO = não e COME = médio **então GORDO**
t_4) se VEGETARIANO = não e COME = muito **então GORDO**

Analisando as regras fornecidas pelo IDRT (regras t) e comparando-as com as do ID3 (regras r) e do ADEX (regras s), concluímos que as primeiras são superiores às duas últimas, visto que:

i) As regras do IDRT, apesar de ligeiramente maiores que as do ID3, em sua totalidade, são corretas do ponto de vista semântico.

ii) Embora as regras do ADEX sejam todas úteis (i. e., todas têm condições relevantes na premissa) elas são bem maiores que as do IDRT.

5. O Problema Sintático e sua Solução Inicial

5.1. O Problema Sintático

Uma das características mais desejáveis nos **SBCs** é que eles sejam facilmente atualizados e modificados. Esta característica provém da modularidade da **BC**, i. e., da possibilidade de alterar o conhecimento relativo a uma parte do domínio mantendo o resto da **BC** intocado.

As **ADs** não são uma forma de representação de conhecimento que proporciona modularidade. Em outras palavras, o conhecimento de áreas específicas do domínio está pulverizado por toda a árvore. Há, entretanto, um problema maior: existe conhecimento que pode ser representado muito mais facilmente em regras que em **ADs**. Considere, por exemplo, o seguinte par de regras:

u_1) **se** VEGETARIANO = sim **então MAGRO**
u_2) **se** COME = pouco **então MAGRO**

Note que as regras são suficientes para classificar MAGRO, em relação ao **CT** da tabela 3.1. Entretanto, não há uma **AD** correspondente a estas regras, porque elas não compartilham nenhum atributo (i. e., não há atributo para ser colocado na raiz).

A **AD** mais simples que classifica as instâncias cobertas por regras que não têm atributos em comum, necessariamente adiciona pelo menos uma condição extra em uma das regras, o que a torna mais específica e frequentemente exige a adição de novos ramos para cobrir eventuais instâncias excluidas pela especialização. No nosso exemplo, tivemos a sorte de não precisar adicionar novos ramos, pois a **AD** da figura 5.1 cobre todas as instâncias classificadas por u_1 e u_2. Todavia, o que mais comumente ocorre nestes casos é que a **AD** é bastante mais complexa que as regras [Cendrowska 88].

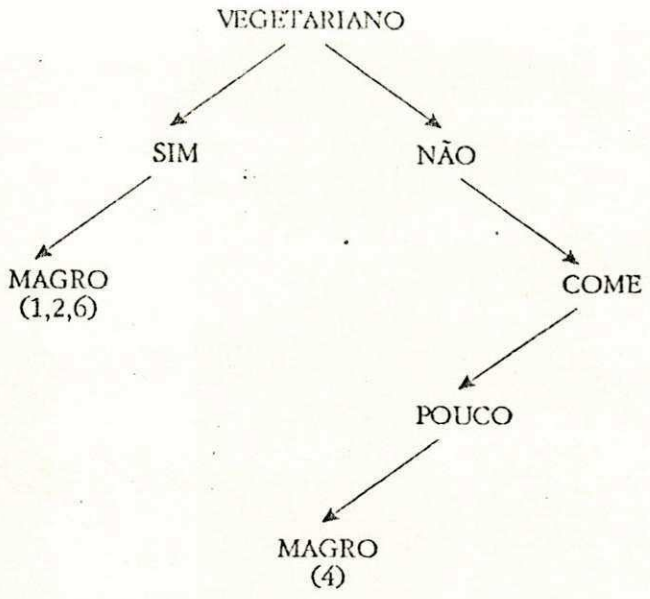

Figura 5.1 - AD correspondente as regras u_1 e u_2

Representar numa **AD** o conhecimento contido em um conjunto de regras não mapeável diretamente em uma árvore traz sérias desvantagens. Além de aumentar o número de condições necessárias para obter algumas classificações, as explanações obtidas da **AD** (ou de regras diretamente mapeadas da **AD**) são mais difíceis de entender. É que as condições adicionadas artificialmente na **AD** muitas vezes são irrelevantes para a classe que determinam. Em outras palavras, o especialista não acredita na explanação porque ela contém, na premissa, condições irrelevantes para a conclusão da classe em questão. Por exemplo, no sistema especialista ginecológico gerado pelo APREND (veja capítulo 4) todas as regras, com exceção de se IDADE = criança **então VAGINITE**, têm a condição IDADE = adulta, o que em nada contribui para clareza das explanações obtidas do sistema.

Outro ponto negativo da conversão forçada de regras em **ADs** é que o conhecimento fica menos "disponível". Considere as regras u_1 e u_2 e a árvore da figura 5.1. Se soubermos apenas a evidência COME = pouco, conseguimos chegar a conclusão através das regras. Já através da **AD**, não conseguimos.

Voltando ao acréscimo desnecessário de condições à **BC**, vale ressaltar que, em muitos domínios, isto inviabiliza o **SBC**. De fato, obter o valor de determinados atributos custa tempo e/ou dinheiro. Para se ter idéia do problema, imagine o gasto de tempo e dinheiro que traria um sistema especialista que tivesse como **BC** uma **AD** com o atributo TOMOGRAFIA-COMPUTADORIZADA na raiz. Pior ainda, a maioria das classes determinadas pela sub-árvore relativa ao valor negativo de TOMOGRAFIA-COMPUTADORIZADA poderiam ser concluídas sem a ajuda desse exame.

O problema sintático é mais grave quando há atributos com grande número de valores, pois aumentam as chances de aparecer valores que não são úteis à conclusão. Colocando de outra forma, ao calcular a função de avaliação dos atributos, os algoritmos que usam AD como saída geralmente fazem a média entre os resultados da função calculada com o atributo assumindo todos os seus valores. O problema aparece quando, apesar do resultado da função não ser "bom" para alguns valores, a média ainda é a melhor.

Observando o conjunto de regras obtidas do ID3 para CT da tabela 3.1, vemos que a regra r_3 poderia ser substituída com vantagem por u_1, pois esta última é mais representativa, tem um número menor de condições e o novo conjunto de regras ainda continua mapeando o CT. No entanto, nenhum algoritmo da família TDIDT seria capaz de fazer isto, pois após tal substituição, não há mais uma AD trivialmente equivalente a esse novo conjunto de regras.

Além do mais, regras não-modulares são pouco "robustas", pois cada exemplo é classificado apenas por uma regra. Com regras modulares, isto pode mudar. De fato, veja que os exemplos ex_1 e ex_6 ficam cobertos, após a substituição de r_3, por duas regras (r_1 e u_1).

5.2. Indução de Regras Modulares

Vários métodos já foram propostos para contornar o problema sintático. A simplificação de ADs é possível pela identificação de partes comuns de ramos distintos, mas a explosão combinatória do número de comparações que precisam ser feitas torna este método aplicável apenas para pequenas árvores. Além disso, a simplificação pode ser feita de diversas formas pois pode haver mais de uma parte comum para um dado conjunto de ramos. A questão que aparece então é qual a melhor simplificação que pode ser feita. A solução para isto envolve consulta a especialistas e/ou o uso de programas de indução que geram novas regras a partir de antigas [Cendrowska 88].

Uma alternativa extremamente interessante consiste em evitar o uso de árvores, pela indução direta de regras modulares. Contornamos, assim, os problemas que aparecem quando tentamos solucionar os problemas intrínsecos às ADs (isto é, os problemas sintáticos). O algoritmo PRISM, proposto por Cendrowska, faz exatamente isso.

5.2.1. Onde Mudar os Algoritmos TDIDT

A causa principal do problema sintático é a maneira pela qual uma AD representa o conhecimento. Quando colocamos um atributo em um nó qualquer de

uma **AD**, estamos nos obrigando a usar todos os seus valores nos ramos (isto é, colocamos na **BC** todas as condições possíveis que contêm o atributo). Ora, frequentemente ocorre que nem todas as condições são úteis para conclusão das classes. Entretanto, temos que colocar as condições inúteis na **AD** para viabilizar o uso das condições interessantes às conclusões.

Pensando nos algoritmos TDIDT o problema sintático pode ser creditado ao uso de funções de avaliação. Funções de avaliação de atributos sempre fornecem uma indicativa global da qualidade do atributo. Elas não contêm informações individualizadas sobre a qualidade das condições que podem ser compostas com o atributo.

Portanto, a mudança que deve ser feita nos algoritmos TDIDT para resolver o problema sintático consiste no uso de indicadores individualizados no lugar dos globais. Assim, enquanto os TDIDT escolhem "bons" atritutos, podemos escolher boas condições, evitando o uso forçado de condições irrelevantes.

5.2.2. O Algoritmo PRISM

O algoritmo PRISM gera as regras para cada um dos elementos de classificação separadamente. Desta forma, a inclusão artificial de condições desnecessárias nas regras é evitada.

A quantidade de informação (*information content*) de um determinado elemento de classificação **E** é calculada pela fórmula:

$$\mathbf{I(E)} = \log_2(1/p(\mathbf{E}))$$

onde: $p(\mathbf{E})$ é a probabilidade de um exemplo ser da classe **E**

Por exemplo, o cálculo da quantidade de informação da classe **MAGRO** no **CT** da tabela 3.1 é feito da seguinte forma:

$$\mathbf{I(MAGRO)} = \log_2(1/p(\mathbf{MAGRO})) = \log_2(1/0{,}5) = \log_2 2 = 1$$

As regras são construídas de forma que a informação ganha (*information gain*) pelas condições da premissa seja igual ou superior a **I(E)**. O processo termina quando todos os exemplos relativos a **E** tiverem sido utilizados para gerar regras. A informação ganha para a classificação de **E** devido à presença da condição **C** na premissa é calculada pela seguinte fórmula:

$I(E|C) = \log_2(p(E|C)/p(E))$

onde: $p(E|C)$ é a probabilidade de um exemplo, dado que contém a condição C, pertencer à classe E

Por exemplo, tomando como base o CT da tabela 3.1, a informação ganha pela condição COME = pouco para a classe **MAGRO** é dada por

$$I(\mathbf{MAGRO}|\text{COME=pouco}) = \log_2\left(\frac{p(\mathbf{MAGRO}|\text{COME=pouco})}{p(\mathbf{MAGRO})}\right) =$$

$$= \log_2(1/0{,}5) = \log_2 2 = 1$$

Já a informação ganha por IDADE = jovem para **GORDO** é

$$I(\mathbf{GORDO}|\text{IDADE=jovem}) = \log_2\left(\frac{p(\mathbf{GORDO}|\text{IDADE=jovem})}{p(\mathbf{GORDO})}\right) =$$

$$= \log_2(0{,}6667/0{,}5) = \log_2 1{,}3334 = 0{,}4151$$

Como o objetivo é ter as menores regras possíveis, o algoritmo se resume a escolher qual condição usar num dado instante. Para atingir esta meta basta escolher o maior valor de I(E|C). Porém, como p(E) é o mesmo para todo C, isso se reduz a obter o maior p(E|C). As provas que o algortimo PRISM produz regras corretas e completas em relação ao conjunto de exemplos usado como entrada, podem ser encontradas no trabalho original de Cendrowska [Cendrowska 88]. A figura 5.2 apresenta o algoritmo PRISM em pseudo-código.

```
prism(CT)
PARA TODA classe E
    ENQUANTO há exemplo em CT com E
    CONJ_PREMISSA <- o/
    INFO_GANHA <- 0
    CT_AUX <- CT
    REPITA
        ACHE, em CT_AUX, C tal que p(E|C) seja máximo
        CONJ_PREMISSA <- CONJ_PREMISSA + C
        INFO_GANHA <- INFO_GANHA + I(E|C)
        RETIRE o atributo de C de CT_AUX
    ATÉ QUE INFO_GANHA ≥ I(E)
    RETIRE de CT todos os exemplos com CONJ_PREMISSA
    IMPRIMA se CONJ_PREMISSA então E
```

Figura 5.2 - Algoritmo PRISM

5.2.3. Exemplo com o Algoritmo PRISM

Utilizando como entrada o CT da tabela 3.1, o algoritmo PRISM produz:

u_1) se VEGETARIANO = sim então **MAGRO**
u_2) se COME = pouco então **MAGRO**
u_3) se VEGETARIANO = não e IDADE = jovem então **GORDO**
u_4) se COME = muito então **GORDO**

Note que, embora tenha solucionado o problema levantado na seção anterior (a regra u_1 agora pertence a BC), o PRISM não resolveu a questão levantada no capítulo 4, pois, ainda há, na regra u_3, uma condição com IDADE, para decidir entre **MAGRO** e **GORDO**.

6. Indução Semântica de Regras Modulares

A idéia da solução global é ter um algoritmo único que reuna as potencialidades do PRISM e do IDRT e que esteja, ao mesmo tempo, livre das deficiências de ambos. Com este objetivo, propomos o RPRISM (*Relevant PRISM*). Ele induz regras modulares (como o PRISM) e usa conhecimento fornecido pelo especialista através de **MRs** (como o IDRT). A combinação destas características é o que permite solucionar simultaneamente os problemas sintático e semântico.

6.1. O Algoritmo RPRISM

O RPRISM é bastante semelhante ao PRISM. A única diferença é que o RPRISM introduz o conceito de probabilidade relevante. Ela é utilizada sempre que for feito um cálculo de probabilidade condicional. Ou seja, o RPRISM difere do PRISM somente porque enquanto o PRISM calcula probabilidades condicionais, o RPRISM computa probabilidades condicionais relevantes. A probabilidade condicional relevante $q(E|C)$ é dada pela fórmula:

$$q(E|C) = \begin{cases} p(E|C), & \text{se C é revelante para E} \\ 1/n_C, & \text{caso contrário} \end{cases}$$

onde: n_C é o número de exemplos que têm a condição C

A probabilidade relevante é, portanto, definida de forma que condições irrelevantes fiquem com os piores valores possíveis, resolvendo assim o problema semântico. De fato, a pior probabilidade que $p(E|C)$ pode ter é $1/n_C$, que define o valor da probabilidade revelante $q(E|C)$ quando C não é relevante para E.

O fato do algoritmo ser baseado no PRISM e produzir regras modulares, e não **ADs**, soluciona também o problema sintático.

Para os mesmos dados de entrada utilizados com o PRISM e com o IDRT (exemplos da tabela 3.1 e **MR** da tabela 4.2), o RPRISM gerou as seguintes regras:

v1) se COME = pouco então **MAGRO**
v2) se VEGETARIANO = sim então **MAGRO**
v3) se VEGETARIANO = não e DIABÉTICO = sim então **GORDO**
v4) se COME = muito então **GORDO**

Veja que, agora, não há mais o problema semântico da regra u_3 (substituida pela regra v_3) nem o problema sintático das regras t_2 e t_4 (substituidas pelas regras v_1 e v_4, respectivamente).

Mesmo considerando que utilizamos um exemplo "de brinquedo" para ilustrar os algoritmos apresentados, podemos ter uma idéia da variação do tamanho e da qualidade das **BCs** geradas na tabela 6.1. Nela, notamos a evolução sintática (através do número e tamanho médio das regras geradas) e semântica dos algoritmos discutidos neste trabalho.

	Regras Geradas	Tamanho Médio	Problemas Semânticos
Aleatório	7	3,00	0
ID3	4	1,50	2
ADEX	4	2,00	0
IDRT	4	1,75	1
PRISM	4	1,25	1
RPRISM	4	1,25	0

Tabela 6.1 - Comparação entre os diversos algoritmos

6.2. Uma Aplicação em Medicina

Para mostrar melhor a diferença entre os resultados obtidos pelos diferentes algoritmos discutidos neste trabalho, vamos mostrar uma pequena aplicação de diagnóstico em medicina. Embora esta aplicação seja ainda um problema de brinquedo, ela é maior que o exemplo ilustrativo que temos usando até agora e, portato, fornece uma melhor validação para as idéias aqui apresentadas.

A aplicação consiste em descobrir o quanto uma dada pessoa é propensa a ter um infarto do miocardio. Para obter este diagnóstico usamos os atributos SEXO (com os valores masculino e feminino), GLICOSE (alta ou normal), STRESS (baixo, alto ou normal), PRESSÃO SANGUINEA (alta ou normal), COLESTEROL (alto ou normal) e TABAGISTA (sim ou não). O grau de propensão ao infarto foi discretizado em quatro classe: **SEM PROPENSÃO**, **POUCO PROPENSO**, **PROPENSO** e **MUITO PROPENSO**. O conjunto de treinamento usado como entrada está na tabela 6.2.

	SEXO	GLICOSE	STRESS	PRESSÃO	COLESTER	TABAG	CLASSE
ex1	masc	normal	baixo	normal	normal	não	SEM PROPENSÃO
ex2	masc	alta	normal	alta	alto	sim	MUITO PROPENSO
ex3	masc	alta	baixo	normal	normal	não	POUCO PROPENSO
ex4	masc	normal	alto	normal	normal	não	POUCO PROPENSO
ex5	fem	normal	baixo	alta	normal	sim	POUCO PROPENSO
ex6	fem	alta	normal	alta	normal	não	POUCO PROPENSO
ex7	fem	alta	normal	normal	normal	sim	POUCO PROPENSO
ex8	fem	alta	baixo	normal	alto	sim	PROPENSO
ex9	fem	alta	normal	normal	alto	não	PROPENSO
ex10	masc	normal	baixo	normal	alto	sim	POUCO PROPENSO

Tabela 6.2 - Conjunto de Treinamento sobre Infarto

Para aqueles algoritmos que usam matriz de relevância, eliciamos de um especialista a MR exposta na tabela 6.3.

	SEM PROP.	POUCO PROP.	PROPENSO	MUITO PROP.
SEXO	Ø	Ø	Ø	Ø
GLICOSE	{normal}	Ø	{alta}	{alta}
STRESS	{baixo}	{baixo,normal}	{normal}	{alto}
PRESSÃO	{normal}	Ø	{alta}	{alta}
COLESTER	{normal}	{normal}	Ø	{alto}
TABAG	{não}	{não}	{sim}	{sim}

Tabela 6.3 - Matriz de Relevância sobre Infarto

Com esses dados de entrada, o ID3 produziu uma árvore equivalente as seguinte regras:

a1) se COLESTER = alto e SEXO = fem **então PROPENSO**
a2) se COLESTER = alto e SEXO = masc e GLICOSE = alta
então MUITA PROP.
a3) se COLESTER = alto e SEXO = masc e GLICOSE = normal
então POUCO PROP.
a4) se COLESTER = normal e SEXO = fem **então POUCO PROP.**
a5) se COLESTER = normal e SEXO = masc e GLICOSE = alta
então POUCO PROP.
a6) se COLESTER = normal e SEXO = masc e GLICOSE = normal
e STRESS = alto **então POUCO PROP.**
a7) se COLESTER = normal e SEXO = masc e GLICOSE = normal
e STRESS = baixo **então SEM PROP.**

Quando estas regras foram expandidas pelo ADEX, obtivemos:

b1) se COLESTER = alto e SEXO = fem e GLICOSE = alta
então PROPENSO
b2) se COLESTER = alto e SEXO = masc e GLICOSE = alta
então MUITA PROP.
b3) se COLESTER = alto e SEXO = masc e GLICOSE = normal
e STRESS = baixo **então POUCO PROP.**
b4) se COLESTER = normal e SEXO = fem **então POUCO PROP.**
b5) se COLESTER = normal e SEXO = masc e GLICOSE = alta
então POUCO PROP.
b6) se COLESTER = normal e SEXO = masc e GLICOSE = normal
e STRESS = alto **então POUCO PROP.**
b7) se COLESTER = normal e SEXO = masc e GLICOSE = normal
e STRESS = baixo **então SEM PROP.**

A indução da AD diretamente pelo IDRT gerou uma árvore equivalente as regras abaixo expostas.

c1) se TABAG = não e COLESTER = alto **então PROPENSO**
c2) se TABAG = não e COLESTER = normal e STRESS = alto
então POUCO PROP.
c3) se TABAG = não e COLESTER = normal e STRESS = baixo
e GLICOSE = alta **então POUCO PROP.**
c4) se TABAG = não e COLESTER = normal e STRESS = baixo
e GLICOSE = normal **então SEM PROP.**
c5) se TABAG = não e COLESTER = normal e STRESS = normal
então POUCO PROP.
c6) se TABAG = sim e STRESS = baixo e GLICOSE = alta
então PROPENSO
c7) se TABAG = sim e STRESS = baixo e GLICOSE = normal
então POUCO PROP.
c8) se TABAG = sim e STRESS = normal e PRESSÃO = alta
então MUITA PROP.
c9) se TABAG = sim e STRESS = normal e PRESSÃO = normal
então POUCO PROP.

A indução de regras modulares feitas pelo PRISM, resultou na seguinte base de conhecimento:

d1) se GLICOSE = normal e TABAG = não e STRESS = baixo
então SEM PROP.
d2) se PRESSÃO = alta e COLESTER = alto **então MUITA PROP.**
d3) se STRESS = alto **então POUCO PROP.**
d4) se COLESTER = normal e PRESSÃO = alta **então POUCO PROP.**

d5) se COLESTER = normal e STRESS = normal **então POUCO PROP.**
d6) se COLESTER = normal e GLICOSE = alta **então POUCO PROP.**
d7) se GLICOSE = normal e TABAG = sim **então POUCO PROP.**
d8) se COLESTER = alto e TABAG = não **então PROPENSO**
d9) se COLESTER = alto e SEXO = fem **então PROPENSO**

Finalmente, a indução semântica feita pelo RPRISM nos deu o seguinte conjunto de regras modulares:

e1) se GLICOSE = normal e TABAG = não e STRESS = baixo
então SEM PROP.
e2) se PRESSÃO = alta e COLESTER = alto **então MUITA PROP.**
e3) se STRESS = alto **então POUCO PROP.**
e4) se COLESTER = normal e STRESS = normal **então POUCO PROP.**
e5) se COLESTER = normal e TABAG = não e STRESS = baixo
e GLICOSE = alta **então POUCO PROP.**
e6) se COLESTER = normal e STRESS = baixo e TABAG = sim
então POUCO PROP.
e7) se STRESS = baixo e COLESTER = alto e GLICOSE = normal
então POUCO PROP.
e8) se GLICOSE = alta e STRESS = baixo e TABAG = sim
então PROPENSO
e9) se GLICOSE = alta e TABAG = não e COLESTER = alto
então PROPENSO

Analisando os resultados dos algortimos concluimos que:

i) O ID3 produziu algumas regras sem muito sentido prático, que foram expandidas pelo ADEX (a a1 e a a3).

ii) A expansão gerada pelo ADEX aumentou indesejadamente o tamanho da BC.

iii) O IDRT gerou uma saída de boa qualidade semântica, mas também grande.

iv) O PRISM gerou um resultado sintaticamente muito bom (o melhor de todos), porém com problemas semânticos (por exemplo, a regra d9).

v) A base de conhecimento gerada pelo RPRISM têm excelente qualidade, embora seja um pouco maior que a gerada pelo PRISM.

A tabela 6.4 resume os resultados obtidos pelos algoritmos para o exemplo do infarto.

	Regras Geradas	Tamanho Médio	Problemas Semânticos
ID3	7	3,00	2 (a_1 e a_3)
ADEX	7	3,29	0
IDRT	9	3,22	1 (c_1)
PRISM	9	2,00	4 (d_3, d_7, d_6 e d_9)
RPRISM	9	2,66	1 (e_3)

Tabela 6.4 - Comparação entre os algoritmos no domínio de infarto

Estas conclusões já eram esperadas. Elas estão de acordo com as idéias expostas neste trabalho. Por isto, esse exemplo contribui para validar nossos resultados sobre indução semântica.

7. Protótipo do Aquisitor Semântico

A construção de um protótipo de um sistema de aquisição de conhecimento que use os algoritmos aqui propostos tem por objetivo permitir a aplicação imediata das idéias apresentadas, mesmo que em problemas de pequeno tamanho. A aplicação prática traz o *feedback* necessário para a continuidade do processo de pesquisa científica.

O protótipo do sistema AS (Aquisitor Semântico) implementa os algoritmos ID3, ID3X (i. e., ADEX baseado no ID3), IDRT usando entropia normalizada como função de avaliação sintática (veja seção 5.2), PRISM e RPRISM. Optamos por implementar todos estes algoritmos com o intuito de fazer comparações entre eles. O protótipo do AS foi escrito em Arity Prolog 5.1 [Arity 88] e funciona em computadores que utilizem o sistema operacional MS-DOS. A linguagem Prolog foi escolhida pela rapidez de desenvolvimento que proporciona, devido ao seu alto nível de abstração. A implementação do algoritmo ID3 foi baseada naquela encontrada em [Araribóia 89].

7.1. Arquitetura Interna

O protótipo do AS tem onze módulos. São eles o módulo pricipal (**AS**), o ID3 (**ASID3**), o ADEX (**ASADEX**), o IDRT (**ASIDRT**), o de auxílio aos algoritmos que usam ADs (**ASAD**), o PRISM (**ASPRISM**), o RPRISM (**ASRPRISM**), o de auxílio aos algoritmos que induzem regras modulares (**ASIRM**), o de utilitários (**ASUTIL**), o de entrada de dados (**ASENTRA**) e o de saída de resultados (**ASSAIDA**). Esquematicamente, a relação entre os módulos está na figura 7.1.

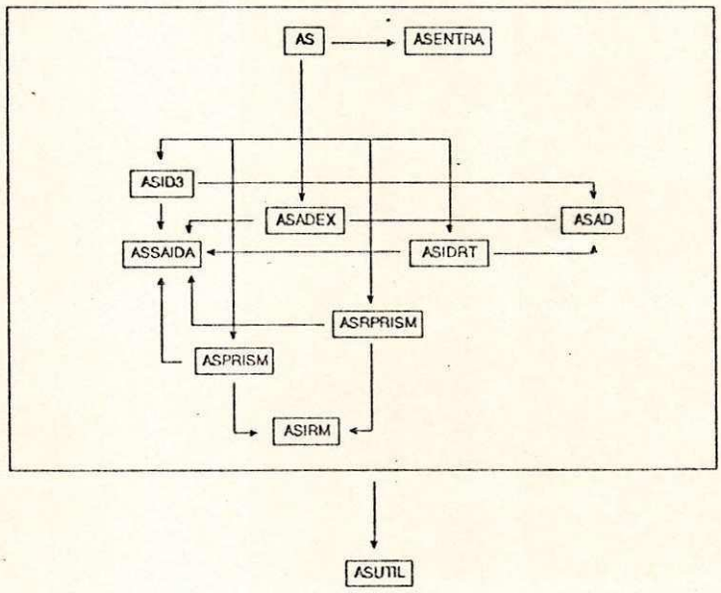

Figura 7.1 - Relação entre os módulos do protótipo do AS

7.2. Modo de Utilização

7.2.1. Definição das Entradas

As entradas do AS são dois arquivos que contêm o **CT** e a **MR**. Eles devem ter o mesmo nome-base, com as terminações **.CT** e **.MR**, respectivamente. O arquivo **.CT** deve conter apenas cláusulas **ex**, todas com aridade 2. O primeiro parâmetro denota a classe e o segundo é a lista das condições que a determinam. As condições são pares atributo = valor, escritos com o uso do operador binário infixo =. Para ilustrar, abaixo segue a listagem do arquivo **EXEMPLO.CT**, que descreve o **CT** apresentado na tabela 3.1.

```
ex(magro, [come=pouco, diabet=sim, idade=velho, veget=sim]).
ex(magro, [come=medio, diabet=nao, idade=velho, veget=sim]).
ex(gordo, [come=muito, diabet=sim, idade=velho, veget=nao]).
ex(magro, [come=pouco, diabet=nao, idade=velho, veget=nao]).
ex(gordo, [come=medio, diabet=sim, idade=jovem, veget=nao]).
ex(magro, [come=pouco, diabet=nao, idade=jovem, veget=sim]).
ex(gordo, [come=muito, diabet=nao, idade=velho, veget=nao]).
ex(gordo, [come=medio, diabet=sim, idade=jovem, veget=nao]).
```

Figura 7.2 - Arquivo EXEMPLO.CT

O arquivo **.MR** contém apenas cláusulas **mr** de aridade 3. Cada cláusula representa um elemento não vazio da **MR**. O primeiro parâmetro representa a linha (i. e., o atributo) e o segundo, a coluna (i. e., a classe). O terceiro parâmetro é o elemento propriamente dito (i. e., a lista dos valores relevantes). Como exemplo, o arquivo **EXEMPLO.MR**, que mapeia a **MR** da tabela 4.2, está listado abaixo.

```
mr(come, magro, [pouco]).
mr(come, gordo, [muito]).
mr(diabet, magro, [nao]).
mr(diabet, gordo, [sim]).
mr(veget, magro, [sim]).
mr(veget, gordo, [nao]).
```

Figura 7.3 - Arquivo EXEMPLO.MR

7.2.2. Executando o Protótipo

O sistema AS pode ser usado tanto interpretado quanto compilado. Obviamente, ele é mais rápido quando está compilado.

Para utilizar o AS compilado, devemos digitar na linha de comando do MS-DOS

AS <ENTRADA>

Onde <ENTRADA> é o nome-base dos arquivos que contêm o **CT** e a **MR** (estes arquivos devem ter as terminações **.CT** e **.MR**, veja a seção anterior). A saída será feita em arquivos com o nome-base <ENTRADA>, tendo como terminações **.ID3**, **.ADX**, **.IRT**, **.PRI** e **.RPR**, que denotam os resultados dos algoritmos ID3, ADEX, IDRT, PRISM e RPRISM, respectivamente.

A utilização do AS interpretado deve ser precedida pela entrada do programa API (*Arity Prolog Interpreter*). Quando aparecer o sinal de prontidão do interpretador, devemos digitar

```
[-as].
carrega.
as(<ENTRADA>).
```

Onde <ENTRADA> é, da mesma forma que para o AS compilado, o nome-base dos arquivos que contêm as entradas. As saídas são mostradas na tela.

No caso dos algoritmos que geram **ADs** (ID3, ADEX e IDRT), antes da saída, há a conversão automática da árvore em regras. Isto foi feito para simplificar o trabalho do módulo que imprime a saída, pois uma árvore é muito mais difícil de imprimir que seus ramos.

8. Conclusões e Trabalhos Futuros

8.1. Conclusões

Sendo a AC o ponto de estrangulamento no desenvolvimento de SBCs, todo esforço no sentido de se obter novos métodos de aquisição ou melhorar os métodos existentes é de suma importância. O processo de aquisição automática via indução a partir de exemplos é um dos mais promissores e, por isto, vem sendo alvo de numerosas investigações.

Os algoritmos de indução da família TDIDT apresentam dois problemas muito sérios: o sintático e o semântico. O problema sintático aparece devido à forma utilizada pelos algoritmos TDIDT para representar o conhecimento adquirido. Sempre são usadas ADs, embora este formalismo não seja adequado para muitos domínios. O problema semântico advém dos algoritmos da família TDIDT não levarem em consideração, durante o processo de indução, as relações de relevância existentes entre as classes e as condições de um dado domínio. Todas as condições são consideradas igualmente importantes para a conclusão de uma determinada classe.

Para solucionar o problema sintático, foi proposto, por Cendrowska, o algoritmo PRISM. O PRISM não mais utiliza ADs como saída. O conhecimento por ele induzido é representado por um conjunto de regras de produção. Os conjuntos de regras são superiores às ADs porque podem representar o conhecimento modularmente, enquanto as ADs não o podem. Entretanto, o PRISM ainda sofre do problema semântico.

Para a solução do problema semântico, fizemos a proposta de estender os algoritmos TDIDT para possibilitar o uso das informações sobre as relações de relevância do domínio. Estas informações seriam obtidas do especialista através de MRs. A MR, por ser mais simples e ter menos expressividade que uma forma convencional de adquirir conhecimento, pode ser obtida fácil e rapidamente. A primeira solução sugerida, denominada algoritmo ADEX, foi a expansão da AD gerada por um algoritmo TDIDT, de forma a incluir condições relevantes nos ramos que não as tivessem. Devido ao aumento de tamanho alcançado pela árvore expandida, esta solução não foi considerada ideal. Alternativamente, propusemos o algoritmo IDRT, que constrói diretamente a AD, mesclando informações sintáticas (encontradas através das funções de avaliação de atributos) com semânticas (representadas na MR). Embora esta solução tenha sido satisfatória para o problema semântico, ela não resolveu o problema sintático, justamente por continuar usando ADs como saída.

O algoritmo RPRISM, que induz regras modulares (como o PRISM), usando informações semânticas obtidas da MR (como o IDRT), foi proposto para resolver ambos os problemas simultaneamente. O RPRISM foi concebido para evitar os problemas sintáticos e semânticos. Vale ressaltar que, embora só tenhamos utilizado o RPRISM em problemas "de brinquedo", certamente ele trará bons resultados para casos reais, pois eles são mais suceptíveis ao problema semântico que os "de brinquedo", já que possuem uma grande quantidade de condições e só uma pequena parcela delas é relevante para cada classe. Como esta informação (qual parcela é relevante) pode ser obtida rapidamente do especialista com o uso de MRs, algoritmos que a aproveitem serão úteis. A vantagem dessa abordagem é evitar que, por coincidência, tenhamos condições que não são relevantes para sua conclusão na BC.

Note que o algoritmo RPRISM representa uma ruptura com os métodos "tradicionais" de AC a partir de exemplos. Devido à inegável complexidade da tarefa de AC, ela é difícil tanto para homens quanto para máquinas. Mas, com este novo enfoque cria-se uma abordagem híbrida entre os métodos automáticos (via indução) e os cognitivos (via MR).

8.2. Trabalhos Futuros

Obviamente, este trabalho não é o ponto final da pesquisa sobre os problemas encontrados com a indução baseada em exemplos (talvez nem exista este ponto final). Resta ainda muita coisa por fazer. Vamos apresentar alguns futuros projetos que julgamos interessantes. São eles:

i) Desenvolver uma nova metodologia de comparação entre soluções, visto que as abordadens tradicionais (acurácia, tamanho, etc) não contemplam o aspecto semântico, agora considerado pelo algortimos.

ii) Incluir outras informações (por exemplo, informações estratégicas), não só sobre relevância, no processo de indução do RPRISM.

iii) A MR também pode ser aplicada em outras situações, pois provê uma informação útil e fácil de ser obtida. Um exemplo disso pode ser encontrado no algoritmo ATM [Gomes 89]. O ATM usa estatística indutiva para construir regras com incerteza. Entretanto, devido a explosão combinatória da quantidade de regras possíveis, somente as regras com uma única condição na premissa são pesquisadas. Poderiamos usar a MR para podar o espaço de busca, só testando as regras que tivessem, na sua premissa, condições relevantes.

iv) A MR também pode ser estendida para aumentar seu poder representacional. Um caminho a seguir é colocar, no lugar dos conjuntos cantorianos,

conjuntos nebulosos nos elementos da MR. Considere que a MR estendida não precisa necessáriamente ser obtida diretamente do especialista. Ela pode ser fruto, por exemplo, da junção do conhecimento (representado em MR "normal") de vários deles.

v) Validar mais completamenet o RPRISM, através de sua aplicação na construção de SBCs de médio ou grande porte.

vi) Permitir que o RPRISM trate atributos com valores contínuos e/ou não disjuntos. As classes também devem poder ser contínuas e/ou não disjuntas.

vii) Melhorar a interface e o desempenho do sistema AS, que hoje é apenas um protótipo. A ligação com outros sistemas também deve ser provida. Para isto, talvez seja necessário reescrevê-lo em outra linguagem.

viii) Integrar o AS a um amplo sistema de aquisição e uso do conhecimento. Este sistema deve ter várias formas de adquirir e usar o conhecimento, pois a construção de SBCs não é tarefa simples nem é resolvida da mesma maneira em todos os casos. Um sistema assim facilitaria bastante a tarefa do EC, já que uma grande variedade de ferramentas estaria disponível. Este sistema pode ser o HOLOS [Oliveira 90].

Referências Bibliográficas

[Araribóia 89] Araribóia, G. *Inteligência Artificial - Um Curso Prático*. Livros Técnicos e Científicos Editora, 1989.

[Arity 88] Arity Corporation. *The Arity/Prolog Language Reference Manual*. 1988.

[Boy 86] Boy, G. *An Expert System for Fault Diagnosis in Orbital Refueling Operations*. AIAA 24th Aerospace Science Meeting. Reno, Nevada, 1986.

[Boy 87] Boy, G.; Faller, B. e Sallantin, J. *Acquisition et Ratification de Connaissances*. Relatório Técnico. CRIM, Montpellier, França, 1987.

[Boman 89] Boman, M. *The Problem of Induction*. Relatório Técnico. SYSLAB, Universidade de Estocolmo, Suécia, 1989.

[Boose 84] Boose, J. H. *Personal Construct Theory and the Transfer of Human Expertise*. Anais do 3º AAAI, 1984.

[Boose 87] Boose, J. H. *Expertise Transfer and Complex Problems: Usign AQUINAS as a Knowledge*. Aquisition Workbench for Knowledge-Based Systems, II Man-Machine Studies. Academic Press, 1987.

[Breiman 84] Breiman, L.; Friedman, J. H.; Olshen, R. A. e Stone, C. J. *Classification and Regression Trees*. Wadsworth International Group, 1984.

[Castiñeira 90] Castiñeira, M. I. e Monard, M. C. *Análise de um Método de Poda para Árvores Indutivas de Decisão*. Anais do 7º Simpósio Brasileiro de Inteligência Artificial. Campina Grande, PB, Brasil, 1990.

[Cendrowska 88] Cendrowska, J. *PRISM: An Algorithm for Inducing Modular Rules*. Knowledge-Based Systems, vol. 1, Academic Press, 1988.

[Charniak 85] Charniak, E. e McDermontt, D. *Introduction to Artificial Intelligence*. Addison-Wesley Publishing Company, 1985.

[Cirne 90] Cirne Filho, W. C. e Mongiovi, G. *Expansão Semântica para os Métodos de Aquisição de Conhecimento da Família TDIDT*. Anais do 1º Simpósio de Inteligencia Artificial y Robotica. Luján, Argentina, 1990.

[Cirne 91a] Cirne Filho, W. C. e Mongiovi, G. *Indução Semântica de Regras Modulares*. Anais do 8º Simpósio Brasileiro de Inteligência Artificial. Brasília, DF, Brasil, 1991.

[Cirne 91b] Cirne Filho, W. C. e Mongiovi, G. *Aquisição de Conhecimento através de Aprendizado Automático*. Anais do 2º Simpósio de Inteligencia Artificial y Robotica. Luján, Argentina, 1991.

[Davis 82] Davis, R.; Shrobe, H; Hamcher, W.; Wieckert, K.; Shirley, M. e Palit, S. *Diagnosis Based on Description of Structure and Function*. Anais do 1º AAAI, 1982.

[Dietterich 81] Dietterich, T. G. e Michalski, R. S. *Inductive Learning of Structural Descriptions: Evaluation Criteria and Comparative Review of Selected Methods*. Artificial Intelligence, vol. 16, 1981.

[Dietterich 86] Dietterich, T. G. e Michalski, R. S. *Learning to Predict Sequencies*. Machine Learning: An Artificial Intelligence Approach, vol. 2. Michalski, Carbonell e Mitchell (eds). Morgan Kaufmann, 1986.

[El-Khomi 88] El-Khomi, B. M. *Inductive Algorithms and Tools: A Survey*. Relatório Técnico. SYSLAB, Universidade de Estocolmo, Suécia, 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.

[Gascuel 88] Gascuel, O. *Critères pour Élaguer la Recherche Lorsque la Complétude et la Cohérence ne Sont pas Requises*. Relatório Técnico, CRIM, Montpellier, França, 1988.

[Genesereth 87] Genesereth, M. R. e Nilsson, N. J. *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann Publishers, 1987.

[Gomes 88] Gomes, F. A. C.; Mongiovi, G. e Silva, H. M. *APREND - Um Sistema de Aquisição Automática de Conhecimento a Partir de Exemplos*; 14a Conferência Latinoamericana de Informática. Buenos Aires, Argentina, 1989.

[Gomes 89] Gomes, F. A. C. *APREND: Um Sistema de Aquisição Automática de Conhecimento a Partir de Exemplos*. Tese de mestrado apresentada na UFPB. Campina Grande, Paraíba, 1989.

[Kguyen 87] Kguyen, T. A., Perkins, W. A., Laffey, T. J. e Pecora, D. *Knowledge Base Verification*. AI Magazine, vol. 8, nº 2, 1987.

[Klir 88] Klir, G. J. e Folger T. A. *Fuzzy Sets, Uncertainty and Information*. Prentice-Hall, 1988.

[Machado 90] Machado, R. J.; Duarte, V.; Denis, F. M.; da Rocha, A. F.; Ramos, M. P. e Guilherme, I. R. *Heuristic Learning Expert System*. Anais do 7º Simpósio Brasileiro de Inteligência Artificial. Campina Grande, PB, Brasil, 1990.

[Mingers 89a] Mingers, J. *An Empirical Comparison of Selection Measures for Decision-Tree Induction*. Machine Learning 3. Kluwer Academic Publishers, 1989.

[Mingers 89b] Mingers, J. *An Empirical Comparison of Pruning Methods for Decision-Tree Induction*. Machine Learning 4. Kluwer Academic Publishers, 1989.

[Mongiovi 90] Mongiovi, G. e Cirne Filho, W. C. *Um Algoritmo baseado em Conhecimento para Ampliar a Potencialidade do ID3*. Anais do 7º Simpósio Brasileiro de Inteligência Artificial. Campina Grande, PB, Brasil, 1990.

[Mongiovi 91] Mongiovi. G e Cirne Filho, W. C. *O Uso de Semântica na Construção e Expansão de Árvores de Decisão Indutivas*. Anais da XVII Conferencia Latinoamericana de Informática. Caracas, Venezuela, 1991.

[Oliveira 90] 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.

[Pingand 89] Pingand, P. e Sallantin, J. *Knowledge Acquisition by Learning: a Full-Scale Experimentation*. Relatório Técnico. CRIM, Montpellier, França, 1989.

[Quinlan 83] Quinlan, J. R. *Learning Efficient Classification Procedures and Their Application to Chess End Games*. Machine Learning: An Artificial Intelligence Approach. Michalski, Carbonell e Mitchell (eds). Morgan Kaufmann, 1983.

[Quinlan 86] Quinlan, J. R. *Induction of Decision Tree*. Machine Learning Journal I. Kluver Academic Publisher, 1986.

[Quinlan 87] Quinlan, J. R. *Generating Production Rules from Decision Trees*. Anais do IJCAI. Milão, Itália, 1987.

[Rich 83] Rich, E. *Artificial Intelligence*. McGraw-Hill, 1983.

[Schlimmer 86] Schlimmer, J. C. e Fisher, D. *A Case Study of Incremental Concept Induction*. Anais do Fifth National Conference on Artificial Intelligence. Morgan Kaufmann, 1986.

[Shapiro 87] Shapiro, A. D. *Structured Induction in Expert Systems*. Adison Wesley, 1987.

[Soucek 89] Soucek, B. e Soucek, M. *Neural and Massively Parallel Computers - The Sixth Generation*. Jonh Willey and Sons, 1989.

[Utgoff 88] Utgoff, P. E. *ID5: An Incremental ID3*. Anais da Fifth International Conference on Machine Learning. Morgan Kaufmann, 1988.

[Utgoff 89] Ufgoff, P. E. *Improved Training via Incremental Learning*. Anais do Sixth International Workshop on Machine Learning, 1989.

[Van de Vilde 89] Van de Vilde, W. *IDL, or Taming the Multiplexer*. Anais do Fourth European Working Session on Learning. Montpellier, França, 1989.

[Varejão 91] Varejão, F. M.; Machado, R. J. e Milidiu, R. L. *ALBA - Um Ambiente para Classificadores Bayesianos por Aglomeração*. Relatório Técnico do Rio Scientific Center, IBM Brasil, 1991.

[Warfield 71] Warfield, J. N. *Societal Systems: Planning, Policy and Complexity*. Jonh Wiley & Sons, 1971.

[Winston 79] Winston, P. H. *Learning by Creating and Justifying Transfer Frames*. Artificial Intelligence: An MIT Perspective. Winston e Brawn (eds). MIT Press, 1979.

Apêndice A. O Algoritmo IDL

É inerente à abordagem TDIDT que todos os elementos do **CT** estejam disponíveis para que seja possível a construção da **AD**. Depois de gerada a **AD**, a adição de novos elementos ao **CT** implica na reconstrução completa da árvore. Há situações nas quais isto é indesejável. Para contornar este problema foram propostos algoritmos que geram resultados idênticos aos obtidos pelos TDIDT, mas o fazem de maneira incremental (i. e., não necessitam de reconstruir integralmente a **AD** quando novos elementos são acrescentados ao **CT**). Por exemplo, o ID4 [Schlimmer 86], o ID5 [Utgof 88] e o ID5R [Utgoff 89] são todos versões incrementais do ID3.

Além de não serem incrementais, os algoritmos TDIDT as vezes geram árvores maiores que as ótimas. Tomemos como exemplo o problema do multiplexador 4 para 1. Um multiplexador 4:1 está representado esquematicamente na figura abaixo.

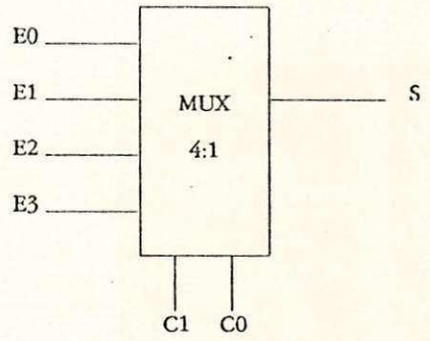

Figura A.1 - Mutiplexador 4:1

Para o multiplexador acima E_0, E_1, E_2, E_3, C_1, C_0 são entradas binárias e S é uma saída binária. S será igual a E_i, onde i é $2.C_1 + C_0$. Ou seja, C_1 e C_0 determinam qual das entradas E_i será apresentada em S.

O problema do multiplexador 4:1 resume-se a induzir uma **AD** que determine qual o valor de S dados E_0, E_1, E_2, E_3, C_1 e C_0. Temos, desta forma, um domínio com 2 classes (0 e 1, que representam os possíveis valores de S) e 6 atributos (E_0, E_1, E_2, E_3, C_1 e C_0), cada qual com 2 valores (0 e 1). Para o **CT** completo (64 elementos), o ID3 gera uma **AD** que têm 25 nós não terminais e 26 folhas. Entretanto, a solução ótima (e também a mais natural) tem apenas 7 nós não-terminais e 8 folhas (veja fig. A.2).

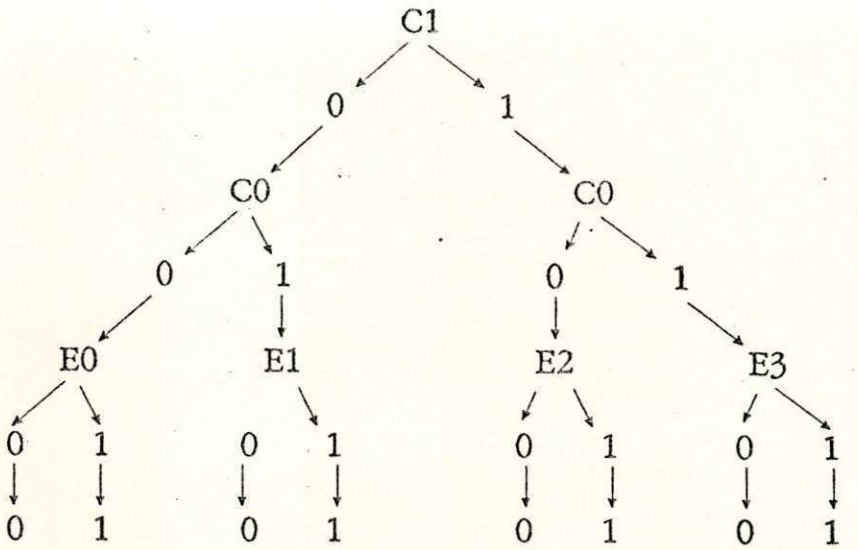

Figura A.2 - AD Ótima para o problema do multiplexador 4:1

Na tentativa de solucionar o problema da sub-otimalidade, Van de Vilde propôs o algoritmo IDL [Van de Vilde 89]. O IDL usa, além de uma função de avaliação (que é um critério estatístico), um critério topológico e é incremental. Em contraste, os algoritmos TDIDT (e também suas versões incrementais) usam apenas um critério estatístico.

A.1. Relevância Topológica

Objetivando permitir o uso de informação topológica, o IDL define o conceito de relevância topológica.

Para uma dada árvore T, um atributo a e um exemplo (i. e., um elemento do CT) e pertencente a uma classe c, a relevância topológica de a para e (denotada por $TR_T(a,e)$) é o número de ocorrências de a nos caminhos reconstruídos a partir de e.

Para uma árvore T e um exemplo e, um caminho reconstruído a partir de e é um caminho que começa numa folha associada à mesma classe de e e segue em direção à raiz, terminando quando a condição expressa em T for inconsistente com e.

Por exemplo, tomando a AD da figura 3.2, temos que os caminhos reconstruidos a partir de ex_1 são (MAGRO COME = pouco) e (MAGRO IDADE = velho). Portanto, as relevâncias topológicas dos atributos do domínio para ex_1 são:

$TR_T(COME, ex_1) = 1$
$TR_T(DIABÉTICO, ex_1) = 0$

$TR_T(IDADE, ex_1) = 1$
$TR_T(VEGETARIANO, ex_1) = 0$

A noção de relevância topológica parece fortemente ligada a de **AD** mínima. Portanto, minimizar a relevância topológica geralmente implica em minimizar a árvore.

Uma árvore **T** é dita ser topologicamente mínima em relação a um conjunto de treinamento **D** se as seguintes condições acontecem:

i) **T** é correta.

ii) **T** não pode ser podada sem perder a corretude.

iii) $TR_T(a,e) \leq 1$ para todo atributo **a** e todo **e** em **D**.

A.2. A Técnica de Transformação

O IDL usa o conceito de relevância topológica através de uma técnica de tranformação que permite minimizar a relevância topológica da árvore sem que ela perca sua corretude. Esta técnica está baseada em idéias usadas pelo ID4 e pelo ID5.

Para que a técnica funcione, é necessário manter, em cada nó, informações que permitam escolher qual é o "melhor" atributo sem ter de re-examinar o **CT**. Estas informações consistem em contadores que guardam, para cada atributo que pode ser usado no nó em questão, a quantidade de exemplos classificados para os diferentes valores do atributo, bem como a que classe pertencem tais exemplos. A idéia dos contadores foi originalmente usada pelo ID4.

O uso de contadores permite descobrir qual é o melhor atributo para ser usado em um determinado nó. Se o melhor atributo não for aquele que já está sendo usado, é possível substitui-lo sem re-examinar o **CT**. Este mecanismo de substituição é inspirado no usado pelo ID5. Ele consiste do seguinte algoritmo, que transforma o nó **N** em um nó que usa o atributo **a**.

```
SE a não é o atributo de N
    SE N é uma folha
        EXPANDA N usando a
    SENÃO
        COLOQUE recursivamente a em todos os filhos de N
        TROQUE a pelo atributo que está em N, refazendo a árvore apenas
         para estes níveis
        PODE os filhos de a quando for possível
        REMOVA os filhos de a que têm apenas um filho
```

A.3. O Algoritmo

O IDL começa com a AD vazia e processa um exemplo por vez. Os passos básicos do algoritmo estão descritos abaixo em pseudo-código. Assuma que há uma função de avaliação disponível (por exemplo, cálculo de entropia) para proporcionar um critério estatístico de avaliação de atributos.

SEJA T a árvore atual e e o novo exemplo a ser classificado
CLASSIFIQUE e, atualizando as informações dos nós do caminho de classificação
SEJA N o nó onde a classificação termina
SE e está incorretamente classificado
    EXPANDA N usando o critério estatístico
SE e não pode ser classificado
    ADICIONE um novo ramo a N e associe a este ramo uma folha com a classe de e
ATUALIZE recursivamente, para os nós que estão no caminho de classificação de e, qual é o melhor atributo. Faça isto da folha para a raiz
PARA TODO nó M no qual é necessária alguma transformação
    SEJAM a_1 ... a_n os atributos que estão abaixo de M
    PARA CADA atributo a_i
        CALCULE a relevância topológica $TR_M(a_i,e)$
        CALCULE a relevância topológica $TR_S(a_i,e)$, onde S é o filho de M que pertence ao caminho de classificação de e
        COMPUTE o ganho de relevância topológica (denotado por TRG_M) entre S e N pela formula:

$$TRG_M = \frac{TR_M(a_i,e) - TR_S(a_i,e)}{TR_M(a_i,e)}$$

        SE a_i tem o ganho de relevância topológica positivo
            USE a técnica de transformação sobre a_i
        SENÃO
            CALCULE o melhor atributo para M
                SE ele for diferente daquele que está em M
                    USE a técnica de transformação para incluir o atributo
        SE todos os filhos de M estão associados à mesma classe (i. e., se a divisão em M ficar obsoleta)
            PODE M

Apêndice B. O Algoritmo C4

O algoritmo ID3 [Quinlan 83] se aplica a problemas determinísticos e nã fornece bons resultados quando usado em domínios que apresentam incerteza. Pa abordar domínios com incerteza, Quinlan propôs o algoritmo C4 [Quinlan 87], que uma extensão do ID3 e consegue lidar com ruídos na entrada.

O C4 divide a construção da **AD** em duas fases distintas. Inicialmente, a A é gerada usando o ID3. A seguir, ela é podada, removendo-se os ramo estatisticamente pouco úteis. A poda permite controlar o tamanho da árvore, qu cresce bastante em domínios não determinísticos se é usado apenas o ID3.

O primeiro passo não será apresentado neste apêndice, pois ele consiste d ID3 e, portanto, está detalhadamente descrito no capítulo 3.

B.1. O Método de Poda do Algoritmo C4

A poda feita pelo C4 baseia-se no cálculo de taxas de erros. A taxa de erro d um nó não-terminal **t** é dada por:

$$r(t) = \frac{e(t)}{N(t)}$$

onde: **e(t)** é o número de exemplos que ficariam classificados incorretamente se **t** fosse uma folha associada à classe de maior frequência entre os exemplos mapeados em **t**
N(t) é o número de exemplos mapeados em **t**

Note que a taxa de erro assim definida é uma taxa otimista, já que os exempl usados para computá-la são os mesmos que foram usados para construir a **AD**. Pa contornar este problema, o C4 corrige a taxa de erro da mesma forma que se corrige aproximação de distribuições de probabilidade discretas feitas pela distribuiç normal. Assim, a taxa de erro corrigida é:

$$r'(t) = \frac{e(t) + 1/2}{N(t)}$$

Para uma sub-árvore que tem **t** como raiz (denotada por T_t), a taxa de erro dada por:

$$r(T_t) = \frac{\Sigma\, e(i)}{\Sigma\, N(i)}$$

Analogamente, a taxa de erro corrigida é dada por:

$$r'(T_t) = \frac{\Sigma\,[e(i) + 1/2]}{\Sigma\, N(i)} = \frac{\Sigma\, e(i) + N_{Tt}/2}{\Sigma\, N(i)}$$

onde: N_{Tt} é o número de folhas de T_t

Como $N(t) = S\ N(i)$, as taxas de erro podem ser simplificadas para o número de classificações erradas. Desta forma, temos:

$n'(t) = e(t) + 1/2$, para um nó não-terminal
$n'(N_t) = S\ e(i) + N_{Tt}/2$, para uma sub-árvore

A poda é feita levando em consideração o aumento do número de erros e a diminuição do número de folhas. Assim, a poda de uma sub-árvore será feita quando o número de erros não aumentar significativamente e levar a uma grande diminuição no número de folhas da AD.

O procedimento de poda envolve o cálculo do erro padrão (*standard error*) da sub-árvore, conforme a definição que segue:

$$SE(T_t) = \sqrt{\frac{n'(T_t)\,.\,[N(t) - n'(T_t)]}{N(t)}}$$

B.2. Um Exemplo do Algoritmo C4

Considere o nó IDADE da AD da figura 3.2. Esta árvore foi gerada pelo ID3 e é, portanto, o resultado do primeiro passo do algoritmo C4 para o CT da tabela 3.1. Para este nó, temos que:

$n'(t) = 1 + 1/2 = 1{,}5$
$n'(T_t) = (0 + 0) + 2/2 = 1$

$$SE(T_t) = \sqrt{\frac{1 \cdot (3 - 1)}{3}} = 0{,}82$$

Portanto, uma vez que 1,82 > 1,5 => 1 + 0,82 > 1,5 => $n'(T_t) + SE(T_t) > n'(t)$, a sub-árvore é podada pelo C4.

B.3. Análise Crítica do Algoritmo C4

A poda do C4 não é um procedimento com forte fundamentação teórica, embora a geração da árvore o seja (está baseada no cálculo de entropia). Por exemplo, o uso da correção de continuidade é totalmente empírico.

Entretanto, alguns resultados interessantes vêm sido obtidos com o método. Suas principais vantagens são:

i) Sua rapidez, se comparado a outros métodos que podam **AD**.

ii) Não precisar de um conjunto de exemplos diferente daquele usado para gerar a **AD**.

Como ponto negativo podemos citar o fato do C4 não gerar várias árvores com diferentes graus de poda, o que é bastante útil quando se tem um especialista para compará-las. Além do mais, Mingers mostrou que há melhores alternativas à poda que a proposta pelo C4 [Mingers 89b].

B.4. Outros Métodos de Poda

O método de poda do C4 é conhecido como Poda por Erro Pessimista (*Pessimistic Error Pruning*). Entre outros métodos de poda, podemos citar a Poda por Redução de Erro (*Reduced-Error Pruning*), a Poda por Erro Mínimo (*Minimum-Error Pruning*), a Poda por Valor Crítico (*Critical Value Pruning*) [Mingers 89b].

Mingers mostrou que há diferenças entre os métodos e que a Poda por Erro Mínimo e a Poda por Erro Pessimista (a usada pelo C4) podem apresentar problemas. Devido a isto, é sugerido o uso de um dos outros três métodos quando se precisa podar **AD**s.

Outro resultado interessante do trabalho do Mingers é que não há evidência de que os métodos de poda e os de geração de **ADs** interagem. Ou seja, os métodos de poda são independentes dos de geração de **ADs** e podem ser intercambiados entre os diversos algoritmos.

```
for (i=0;i<r_trig;i++) {        /* Canais do multiplexador usados como *
                                * canais de gatilho                  */
    fscanf(in,"%d\n",&tempd);   /* Numero do canal                    */
    if (tempd<0) tempd=0; if (tempd>15) tempd=15;
    r_chns[i]=tempd; trig[tempd]=i;
    fscanf(in,"%f\n",&tempf);   /* Limiar superior em percentagem     */
    if (tempf<-100.0) tempf=-100.0; if (tempf>100.0) tempf=100.0;
    r_lims[i]=tempf;
    fscanf(in,"%f\n",&tempf);   /* Limiar inferior em percentagem     */
    if (tempf<-100.0) tempf=-100.0; if (tempf>100.0) tempf=100.0;
    r_limi[i]=tempf;
    if (r_limi[i]>r_lims[i]) { r_limi[i]=tempf; r_limi[i]=r_lims[i]; r_lims[i]=tempf; }
    }
            /* Parametros para a visualizacao na tela             */
for (i=0;i<16;i++) {'
    fscanf(in,"%s %s",dat,string);  /* Nome da variavel do canal */
    strcpy(var_name[i],string);
    fscanf(in,"%s %s",dat,string);  /* Unidade do canal */
    strcpy(unit_name[i],string);
    fscanf(in,"%s %f",dat,&tempf); /* Valor maximo do canal */
    vmax[i]=tempf;
    fscanf(in,"%s %f",dat,&tempf); /* Valor minimo do canal */
    vmin[i]=tempf;
    if (vmin[i]>vmax[i]) { tempf=vmax[i]; vmax[i]=vmin[i]; vmin[i]=tempf; }
    }
fclose (in);
/* Checa espaço livre em disco corrente. Se não há espaço, desliga registro. */
getdfree(0,&dt); fmem=(long) (dt.df_avail*dt.df_bsec*dt.df_sclus);
```

```
if (fmem<8000) { fputc(7,stdout); r_modo=0; }
/* Formação parcial do nome do arquivo do registro */
if (r_modo==1) ext[0]='#';
if (r_modo==2) ext[0]='$';
r_ativa=0; r_flag=0;
/* ***************** INICIALIZE A PIACA DE AQUISIÇÃO DE DADOS ********************** */
/*    REGISTRO DE CONTROLE <BASE+9>:  b7  b6 b5 b4 b3   b2     b1  b0        */
/*                                                [INTE I2  I1   I0  x   DMAE ST1 ST0]     */
outportb(base+9,creg);
if (inportb(base+9) != creg) {
   gotoxy(5,23); fputc(7,stdout);
   cprintf("FALHA NO SISTEMA DE AQUISIÇÃO DE DADOS!");
   return -3;
   }
outportb(base+8,1); /* reseta o bit de solicitacao de interrupcao */
/* ***************** LE O REGISTRO DO STATUS DO CONVERSOR A/D ******************** */
/*   REGISTRO DE STATUS A/D <BASE+8>: b7   b6   b5   b4   b3   b2   b1   b0     */
/*                                                [EOC UNI MUX INT CN3 CN2 CN1 CN0]   */
ad_bias=ch_type=0;
st = inportb(base+8);
if ((st & 0x40) == 0x40) ad_bias=-1;  /* Conversao unipolar       */
else ad_bias=1;                  /* Conversao bipolar        */
if ((st & 0x20)==0x20)   ch_type=1;   /* Canais A/D "single-ended" */
else ch_type=-1;                 /* Canais A/D diferenciais   */
mux_config();  /* Configura registro do multiplexador */

/* ****************** CONFIGURA O REGISTRO DE SINAIS ********************** */
if (r_modo) {
```

```
    pre_N=(unsigned) (pre_data/sample); pos_N=(unsigned) (pos_data/sample);
    r_buf=(unsigned huge *) farcalloc((pre_N+pos_N),sizeof(unsigned));
    r_chn=(unsigned huge *) farcalloc((pre_N+pos_N),sizeof(unsigned));
    if (r_buf==NULL || r_chn==NULL) {
       gotoxy(5,23); fputc(7,stdout);
       cprintf("NAO HA MEMORIA PARA O REGISTRO DE SINAIS...");
       gotoxy(5,24);
       cprintf("REGISTRO DE SINAIS DESLIGADO!");
       r_modo=0;
       fprintf(r_out,"MEMORIA ESGOTADA!!\n\n"); fclose(r_out);
       }
    else r_ativa=1; /* ativa registro */
    for (i=0;i<r_trig;i++) {
       lims[i]=(unsigned) 2048+20.475*r_lims[i];
       limi[i]=(unsigned) 2048+20.475*r_limi[i];
       if (lims[i]>4095) lims[i]=4095; if (limi[i]>4095) limi[i]=4095;
       }
    }

pre_ptr=0; pos_ptr=pre_N;

/* ********* INICIALIZACAO DO MONITOR DE VIDEO NO MODO GRAFICO ********** */
if (graph_init()) {
    gotoxy(5,23); fputc(7,stdout);
    cprintf("FALHA NA INICIALIZACAO DO MODO GRAFICO !");
    return -5;
}
denorm();
```

```
graphcolor[0]=EGA_BLACK; graphcolor[1]=EGA_GREEN;
graphcolor[2]=EGA_BLUE;  graphcolor[3]=EGA_RED;
graphcolor[4]=EGA_LIGHTGRAY; graphcolor[5]=EGA_LIGHTGREEN;
graphcolor[6]=EGA_LIGHTBLUE; graphcolor[7]=EGA_LIGHTRED;

/* ******* DETERMINACAO DA JANELA DE EXIBICAO DOS SINAIS ***************** */
xmn=Graphs[0]; xmx=Graphs[2]; dx=xmx-xmn;
ymn=Graphs[3]; ymx=Graphs[1]; nzy=(ymx+ymn)/2;

/* ******* CÁLCULO INCREMENTO DAS COORDENADAS-Y DOS SINAIS EM PIXELS ****** */
dyt=(float ) -(ymx-ymn)/4096.;
/*  PREPARACAO DE VARIAVEIS AUXILIARES PARA A CONVERSAO A/D EM TEMPO REAL
*/
kbch=0; if (loop<0) { loop=-1; ptr=-2; } else ptr=0;
final=REFRESH; ad_modify(&final);

/* ********** INSTALACAO DA ROTINA DE INTERRUPCAO DO TECLADO ********** */
install_interrupts();

/* ***** LOOP DE CONVERSAO A/D E EXIBICAO DOS SINAIS EM TEMPO REAL ****** */
do {
   final=ad_loop(); /* Malha de monitoração com registro de perturbações */
   ad_modify(&final);
}
while (final!=END);

/* * TERMINO DO LOOP DE CONVERSAO A/D E EXIBICAO DOS SINAIS EM TEMPO REAL */
deinstall_interrupts(); /* Restaura antiga tabela de vetores de interrupção */
```

```
cleardevice();
closegraph(); /* Encerra Modo gráfico */
if (r_modo) { fclose(r_out); farfree(r_buf); }
free(xy_map);
return 0;
}

/* **** ROTINA PARA A INSTALACAO DE ROTINAS DE DEFINIDAS PELO USUARIO NA
TABELA DE INTERRUPCAO **** */
void  install_interrupts (void )
{
disable(); /* Impede todas as interrupções por software*/
imr=inportb(0x021);
do { outportb(0x021,0xFF); } while (inportb(0x021) != 0xFF); /* Impede todas os sinais de
interrupção na ehtrada do controlador */
oldint09=getvect(0x09);
oldintTs=getvect(0x0F);
setvect(0x09,newint09); /* Vetor de interrupção do teclado */
setvect(0x0F,newintTs); /* Vetor de interrupção da rotina que controla o período de amostragem
*/
do { outportb(0x021,0x00); } while (inportb(0x021) != 0x00); /* Libere todas os sinais de
interrupção na entrada do controlador */
enable(); /* Libere as interrupções por software*/
}

/* ********* ROTINA PARA A RESTAURACAO DA TABELA DE INTERRUPCAO ********* */
void  deinstall_interrupts (void )
{
```

```
do { outportb(0x021,0xFF); } while (inportb(0x021) != 0xFF); /* Impede todas os sinais de
interrupção na entrada do controlador */
setvect(0x09,oldint09);
setvect(0x0F,oldintTs);
do { outportb(0x021,imr); } while (inportb(0x021) != imr); /* Libere aqueles sinais de interrupção
na entrada do controlador anteriormente selecionados */
enable(); /* Libere as interrupções por software*/
}

/* ***************** ROTINA DE INTERRUPCAO PARA TECLADO ***************** */
void interrupt newint09 (...)
{
static unsigned char al=0, ah=0;
enable();
kbch=inportb(0x060);
al=inportb(0x061); ah=al; al=al|0x80;
outportb(0x061,al); outportb(0x061,ah);
outportb(0x020,0x20);
}

/* ***************** ROTINA DE INTERRUPCAO PARA CONTROLAR O PERIODO DE
AMOSTRAGEM ***************** */
void interrupt newintTs (...)
{
enable();
ad_flag=1;
outportb(0x020,0x20);
```

```
}

/* ********** ROTINA DA TELA DE EXIBICAO DOS SINAIS ********** */
void graph_frame (void)
{
char  string[85];
struct tm  tt;
extern int  v_chns[16],dxx[16];
extern unsigned offset;
extern float  vmax[16],vmin[16];
extern char  unit_name[16][10],var_name[16][10];
int   h;

gettime(&t); /* Obtem a data e a hora via DOS */
/* Preparar as variáveis com a data e a hora para sua exibição */
tt.tm_sec =t.ti_sec; tt.tm_min=t.ti_min; tt.tm_hour=t.ti_hour;
tt.tm_mday=d.da_day; tt.tm_mon=d.da_mon; tt.tm_year=d.da_year;
/* Prepara a tela gráficopadrão */
setlinestyle(SOLID_LINE,1,THICK_WIDTH);
setcolor(EGA_RED);
rectangle(W_title [0],W_title [1],W_title [2],W_title [3]);
rectangle(W_graph [0],W_graph [1],W_graph [2],W_graph [3]);
rectangle(W_legend[0],W_legend[1],W_legend[2],W_legend[3]);
setfillstyle(SOLID_FILL,EGA_LIGHTGRAY);
bar(W_title [0]+3,W_title [1]+3,W_title [2]-3,W_title [3]-3);
setcolor(EGA_WHITE);

settextstyle(8,HORIZ_DIR,0);   /* Estilo de texto          */
```

```
strcpy(string,"MONITORAÇÃO COM REGISTRO DE SINAIS");
setusercharsize(5,10,6,8); h=textheight(string);
outtextxy(W_title[0]+50,W_title[1],string);
setcolor(EGA_WHITE);
sprintf(string,"Campina Grande-PB,%02d de %s de %04d às %02d:%02d%.2s",
tt.tm_mday,mes[tt.tm_mon-1],tt.tm_year,tt.tm_hour,tt.tm_min,pm(&tt));
setusercharsize(1,2,3,8);
outtextxy(W_title[0]+50,W_title[1]+h,string);
setfillstyle(SOLID_FILL,EGA_WHITE);
bar(W_graph [0]+3,W_graph [1]+3,W_graph [2]-3,W_graph [3]-3);
setfillstyle(SOLID_FILL,EGA_WHITE);
bar(Graphs[0]-3,Graphs[1],Graphs[2]+3,Graphs[3]);
setfillstyle(SOLID_FILL,EGA_LIGHTGRAY);
bar(W_legend[0]+3,W_legend[1]+3,W_legend[2]-3,W_legend[3]-3);
settextstyle(8,HORIZ_DIR,0);   /* Estilo de texto          */
setusercharsize(1,2,1,3);
sprintf(string,"canal %02d: %-.5s %+.2f %+.2f %c",v_chns[offset+0],
        var_name[v_chns[offset+0]],vmin[v_chns[offset+0]],vmax[v_chns[offset+0]],
        unit_name[v_chns[offset+0]][0]);
h=textheight(string);
if (mux_lo<=offset+0 && mux_hi>=offset+0) {
    setcolor(graphcolor[(offset+0)%8]);
    moveto(10,W_legend[1]+1*h+4); linerel(25,0);
}
if (mux_lo<=offset+1 && mux_hi>=offset+1) {
    setcolor(graphcolor[(offset+1)%8]);
    moveto(10,W_legend[1]+2*h+4); linerel(25,0);
}
```

```
if (mux_lo<=offset+2 && mux_hi>=offset+2) {
    setcolor(graphcolor[(offset+2)%8]);
    moveto(10,W_legend[1]+3*h+4); linerel(25,0);
}
if (mux_lo<=offset+3 && mux_hi>=offset+3) {
    setcolor(graphcolor[(offset+3)%8]);
    moveto(10,W_legend[1]+4*h+4); linerel(25,0);
}
setcolor(EGA_BLACK);
if (mux_lo<=offset+0 && mux_hi>=offset+0) { outtextxy(40,W_legend[1]+1*h-5,string); }
sprintf(string,"canal %02d: %-.5s %+.2f %+.2f %c",v_chns[offset+1],
        var_name[v_chns[offset+1]],vmin[v_chns[offset+1]],vmax[v_chns[offset+1]],
        unit_name[v_chns[offset+1]][0]);
if (mux_lo<=offset+1 && mux_hi>=offset+1) { outtextxy(40,W_legend[1]+2*h-5,string); }
sprintf(string,"canal %02d: %-.5s %+.2f %+.2f %c",v_chns[offset+2],
        var_name[v_chns[offset+2]],vmin[v_chns[offset+2]],vmax[v_chns[offset+2]],
        unit_name[v_chns[offset+2]][0]);
if (mux_lo<=offset+2 && mux_hi>=offset+2) { outtextxy(40,W_legend[1]+3*h-5,string); }
sprintf(string,"canal %02d: %-.5s %+.2f %+.2f %c",v_chns[offset+3],
        var_name[v_chns[offset+3]],vmin[v_chns[offset+3]],vmax[v_chns[offset+3]],
        unit_name[v_chns[offset+3]][0]);
if (mux_lo<=offset+3 && mux_hi>=offset+3) { outtextxy(40,W_legend[1]+4*h-5,string); }

setcolor(EGA_BLACK);
settextstyle(3,HORIZ_DIR,0);
setusercharsize(1,3,1,3);
if (r_modo==0) outtextxy(400,W_legend[1]+1*h-5,">> REGISTRO DESLIGADO");
if (r_modo==1) outtextxy(400,W_legend[1]+1*h-5,">> GATILHAMENTO MANUAL");
```

```
if (r_modo==2) outtextxy(400,W_legend[1]+1*h-5,">> GATILHAMENTO POR SOFTWARE");

if (r_modo==1) {
    strcpy(string,"LOGICA: TECLA <SPACE>");
    h=textheight(string);
    outtextxy(400,W_legend[1]+2*h,string);
}

if (r_modo==2) {
    sprintf(string,"LOGICA: C#<Ymin ou C#>Ymax");
    h=textheight(string);
    outtextxy(400,W_legend[1]+2*h,string);
    sprintf(string,"# CANAIS: %d",r_trig);
    outtextxy(400,W_legend[1]+3*h,string);
}

sprintf(string,"MUX:%d-%d",mux_lo,mux_hi);
outtextxy(400,W_legend[1]+4*h,string);

switch (offset) {
    case 0: /* Mostra os canais 0 a 3 */
        setfillstyle(SOLID_FILL,EGA_RED);
        bar(W_title[2]-50,W_title[3]-30,W_title[2]-40,W_title[3]-20);
    break;
    case 4: /* Mostra os canais 4 a 7 */
        setfillstyle(SOLID_FILL,EGA_YELLOW);
        bar(W_title[2]-50,W_title[3]-30,W_title[2]-40,W_title[3]-20);
    break;
```

```
    case 8: /* Mostra os canais 8 a 11 */
        setfillstyle(SOLID_FILL,EGA_GREEN);
        bar(W_title[2]-50,W_title[3]-30,W_title[2]-40,W_title[3]-20);
    break;
    case 12: /* Mostra os canais 12 a 15 */
        setfillstyle(SOLID_FILL,EGA_CYAN);
        bar(W_title[2]-50,W_title[3]-30,W_title[2]-40,W_title[3]-20);
    break;
}

sprintf(string,"%d-%d",offset,offset+3);
h=textheight(string);
outtextxy(W_title[2]-55,W_title[3]-2*h,string);
}

/* *************** ROTINA QUE EXIBE 'AM' OU 'PM' PARA AS HORAS ************** */
char *pm(struct tm *t)
{
if  (t->tm_hour<12) return "AM";
else return "PM";
}

/* ******* ROTINA DE MONITORACAO GRAFICA E REGISTRO DE PERTURBACOES ****** */
int ad_loop (void)
{
do {
     if (ad_flag==1) { /* Checa se o sinal para nova amostragem já chegou */
         if (freeze==0) {    /* A TELA NAO ESTA' CONGELADA */
```

```
outportb(base,0); /* COMANDO DE SOFTWARE PARA CONVERSÃO A/D  */
        /* LOOP DE ESPERA DA CONVERSAO A/D */
while ((inportb(base+8)&0x80) == 0x80);
        /* TRANSFERENCIA DOS DADOS A/D PARA MEMORIA   */
adl=inportb(base); adh=inportb(base+1);
adt=(unsigned) ((adh<<8)|adl)>>4; chn=adl&15;
if (trig[chn]>=0) {  /* É CANAL DE GATILHO DO REGISTRO */
   if (pre_ptr<=(mux_hi-mux_lo)) { old_adt[chn]=adt; cnst=0; }
   else { cnst=adt-old_adt[chn]; old_adt[chn]=adt; }
      /* HOUVE UMA PERTURBACAO ENTAO ACIONA A MEMORIA POS
        DATA DO REGISTRO */
   if (cnst<=limi[chn] || cnst>=lims[chn]) {
      if (r_modo==2 && r_flag==0) {
         r_flag=1;
         r_count++;
         gettime(&t); }
         getdfree(0,&dt); fmem=(long) (dt.df_avail*dt.df_bsec*dt.df_sclus);
         if (fmem<8000) { fputc(7,stdout); r_modo=0; }
         }
    }
if (r_ativa==1) {   /* REGISTRO DE SINAIS E' ATIVADO?      */
   if (r_flag==1) { /* ESTA' GRAVANDO NA MEMORIA POS-DATA? */
     *(r_buf+pos_ptr)=(unsigned) adt;
     *(r_chn+pos_ptr)=(unsigned) chn; pos_ptr++;
     if (pos_ptr>(pre_N+pos_N)) r_ativa=0;
     }
   else {          /* REGISTRO ESTA' GRAVANDO NA MEMORIA PRE-DATA */
     *(r_buf+pre_ptr)=(unsigned) adt;
```

```
        *(r_chn+pre_ptr)=(unsigned) chn; pre_ptr++;
        if (pre_ptr>pre_N) pre_ptr=0;
        }
      }
        /* CHECAGEM DO SINAL CONVERTIDO PARA SUA EXIBICAO
           NA TELA */
    if (chn>=offset && chn<offset+4) {
      x=(int) xmn+xx[chn]+1; y=(int) ymx+adt*dyt;
      if (xx[chn]<=0) ya[chn]=y;
        /* PLOTA O RESULTADO DA CONVERSAO A/D NA TELA */
      setcolor(graphcolor[chn%8]; line(xa[chn],ya[chn],x,y);
      xx[chn]++; xa[chn]=x; ya[chn]=y;
       /* CHECAGEM SE TRACADO DO SINAL CHEGOU NO FINAL DA
          TELA */
      if (xx[chn]>=dx-5) {
        setviewport(xmn+1,ymx,xmx,ymn,1);
        vpage=apage; apage=vpage; vpage=tpage;
        setvisualpage(vpage); setactivepage(apage); /* Troca de página do video */
       /* PREPARACAO DA TELA PARA PROXIMA EXIBICAO */
        clearviewport();
        if (one_shot==1) freeze=1;
        setviewport(0,0,scrx,scry,1);
        setfillstyle(SOLID_FILL,EGA_WHITE);
        bar(Graphs[0]-3,Graphs[1],Graphs[2]+3,Graphs[3]);
        xx[offset+0]=xx[offset+1]=xx[offset+2]=xx[offset+3]=0;
        xa[offset+0]=xa[offset+1]=xa[offset+2]=xa[offset+3]=xmn+1;
       /* CHECAGEM SE GRAVACAO NA MEMORIA POS-DATA CHEGOU
          NO FINAL. GRAVA RESULTADO EM DISQUETE E PREPARA O
```

```
                    REGISTRO PARA PROXIMA PERTURBACAO          */
                  if (r_ativa==0 && r_flag==1) {
                    disable();
                    sprintf(file,"%s%02d%02d%c",local,t.ti_hour,t.ti_min,t.ti_sec+128);
                  _makepath(r_file,drive,dir,file,ext);
                  if ((r_out=fopen(r_file,"wt+"))!=NULL) {
                    fprintf(r_out,"  %04X\t%04X\n",pre_ptr,pre_N);
                    for (i=0;i<pre_N+pos_N;i++)
                    fprintf(r_out,"%01X %01X\t%04X\n",i,*(r_chn+i),*(r_buf+i));
                    fclose(r_out);
                    }
                r_ativa=1; r_flag=0; pre_ptr=0; pos_ptr=pre_N;
                enable();
                  }
                  }'
              }
            }
          }

                  /* FOI PRESSIONADA A TECLA <ESC>? SE FOR FINALIZA A
                    CONVERSAO E EXIBICAO GRAFICA */
if (kbch==1) { kbch=0; fputc(0x07,stdout); return END; }

                  /* FOI PRESSIONADA A TECLA <SPACE>? SE FOR ACIONA O
                    REGISTRO DE SINAIS */
if (kbch==57) {
    kbch=0;
    if (r_modo==1) {
```

```
        r_flag=1; gettime(&t);
        getdfree(0,&dt); fmem=(long) (dt.df_avail*dt.df_bsec*dt.df_sclus);
        if (fmem<8000) { fputc(7,stdout); r_modo=0; }
        fputc(0x07,stdout);
        }
    }

                /* FOI PRESSIONADA A TECLA <D>? SE FOR, CONGELA OU
                   DESCONGELA TELA */
if (kbch==25) {
    kbch=0;
    if (freeze==0) freeze=1;
    else { freeze=one_shot=0; return REFRESH; }
    }

                /* FOI PRESSIONADA A TECLA <S>? SE FOR, ENQUANDO A
                   TELA ESTA' CONGELADA, LIMPA A TELA, REALIZA 565
                   AMOSTRAS (UMA LARGURA DE  TELA) E CONGELA NOVAMENTE */
if (kbch==31) {
    kbch=0;
    if (freeze==1) { one_shot=1; freeze=0; return ONE_SHOT; }
    }

                /* FOI PRESSIONADA A TECLA <L>? SE FOR, LIMPA A
                   TELA */
if (kbch==19) { kbch=0; return REFRESH; }

                /* FOI PRESSIONADA A TECLA <SETA PARA DIREITA>?
```

```
                SE FOR, MOSTRA O CONJUNTO DE 4 SINAIS POSTERIOR*/
if (kbch==77) {
    kbch=0;
    switch (offset) {
        case 0:  offset=4;  break;
        case 4:  offset=8;  break;
        case 8:  offset=12; break;
        case 12: offset=0;  break;
        }
    return REFRESH;
    }

            /* FOI PRESSIONADA A TECLA <SETA PARA ESQUERDA>?
               SE FOR, MOSTRA O CONJUNTO DE 4 SINAIS ANTERIOR */
if (kbch==75) {
    kbch=0;
    switch (offset) {
        case 0:  offset=12; break;
        case 4:  offset=0;  break;
        case 8:  offset=4;  break;
        case 12: offset=8;  break;
        }
    return REFRESH;
    }

            /* FOI PRESSIONADA A TECLA <SETA PARA CIMA> OU <SETA
               PARA BAIXO>? SE FOR, SELECIONA OS N CANAIS PROXIMOS
               ou ANTERIORES CANAIS DO MULIPLEXADOR, RESPECTIVAMEN
```

```
                    TE */
if (kbch==72) { kbch=0; mux_hi++; mux_lo++; return MUX_UPDATE; }
if (kbch==80) { kbch=0; mux_hi--; mux_lo--; return MUX_UPDATE; }
                /* FOI PRESSIONADA A TECLA <-> OU <+>? SE FOR, AUMENTA
                   O NUMERO DE CANAIS DO MULIPLEXADOR SELECIONADO,
                   RESPECTIVAMENTE */
if (kbch==74) {
    kbch=0; mux_hi--;
    if (mux_hi<mux_lo) mux_hi=mux_lo;
    return MUX_UPDATE;
    }
if (kbch==78) {
    kbch=0; mux_hi++;
    if ((mux_hi-mux_lo)>15) mux_hi--;
    return MUX_UPDATE;
    }

if (loop<0) { loop=-1; ptr=-2; } else ptr++;
}
while (ptr<loop);
return END;
}

/* ROTINA PARA ATUALIZAR PARAMETROS PARA MONITORAÇÃO E REGISTRO DE
PERTURBAÇÕES */
void ad_modify(int *flag)
{
switch (*flag) {
```

```
case END    : return; /*Finaliza monitoração */
case MUX_UPDATE: /* Atualiza o registro do multiplexador da placa */
        mux_config();
case REFRESH : /*  Atualiza tela gráfica */
        cleardevice();
        setactivepage(0);
        directvideo=1; graph_frame();
        graph_grid2(xmn-4,xmx+4,ymn+2,ymx-2,5,EGA_BLACK);
  graph_numb(xmn,xmx,ymn-10,ymx-10,5,-100.0,100.0,sample,ad_bias,EGA_BLACK);
        outtextxy(xmn+250,ymn+10,"TEMPO EM SEGUNDOS");
        settextstyle(8,VERT_DIR,0); setusercharsize(1,3,1,4);
        outtextxy(W_graph[0]+7,W_graph[1]+30,"UNIDADES EM PERCENTAGEM");
        setactivepage(1);
        directvideo=1; graph_frame();
        graph_grid2(xmn-4,xmx+4,ymn+2,ymx-2,5,EGA_BLACK);
  graph_numb(xmn,xmx,ymn-10,ymx-10,5,-100.0,100.0,sample,ad_bias,EGA_BLACK);
        outtextxy(xmn+250,ymn+10,"TEMPO EM SEGUNDOS");
        settextstyle(8,VERT_DIR,0); setusercharsize(1,3,1,4);
        outtextxy(W_graph[0]+7,W_graph[1]+30,"UNIDADES EM PERCENTAGEM");
        vpage=0; apage=1; /* Página visual é a página 0  Página ativa é a página 1 */
        setvisualpage(vpage); setactivepage(apage);
        xx[offset+3]=xx[offset+2]=xx[offset+1]=xx[offset+0]=0;
        xa[offset+3]=xa[offset+2]=xa[offset+1]=xa[offset+0]=xmn+1;
        ya[offset+3]=ya[offset+2]=ya[offset+1]=ya[offset+0]=nzy;
        break;
case ONE_SHOT: /* Aquisição de 565 amostras por canal */
        setviewport(xmn+1,ymx,xmx,ymn,1);
        tpage=apage; apage=vpage; vpage=tpage;
```

```
            setvisualpage(vpage); setactivepage(apage);
            clearviewport();
            setviewport(0,0,scrx,scry,1);
            setfillstyle(SOLID_FILL,EGA_WHITE);
            bar(Graphs[0]-3,Graphs[1],Graphs[2]+3,Graphs[3]);
            xx[offset+0]=xx[offset+1]=xx[offset+2]=xx[offset+3]=0;
            xa[offset+0]=xa[offset+1]=xa[offset+2]=xa[offset+3]=xmn+1;
            break;
            }
}

/* ROTINA PARA CONFIGURAR/RECONFIGURAR O REGISTRO DO MULTIPLEXADOR DA
PLACA DE AQUISIÇÃO DE DADOS */
void mux_config(void)
{
/* ********** COLOCA A FAIXA DE VARREDURA DO MULTIPLEXADOR ********** */
/*  REGISTRO DO MULTIPLEXADOR <BASE+2>: b7  b6   b5  b4   b3  b2   b1   b0  */
/*                                      [CH3 CH2 CH1 CH0 CL3 CL2 CL1 CL0] */
int val;

if ((ch_type==1)  && (mux_hi>15)) mux_hi= 0;
if ((ch_type==1)  && (mux_hi< 0)) mux_hi=15;
if ((ch_type==1)  && (mux_lo< 0)) mux_lo=15;
if ((ch_type==1)  && (mux_lo>15)) mux_lo= 0;
if ((ch_type==-1) && (mux_hi> 7)) mux_hi= 0;
if ((ch_type==-1) && (mux_hi< 0)) mux_hi= 7;
if ((ch_type==-1) && (mux_lo< 0)) mux_hi= 7;
if ((ch_type==-1) && (mux_lo> 7)) mux_hi= 0;
```

```
if (mux_lo>mux_hi) { val=mux_hi; mux_hi=mux_lo; mux_lo=val; }
val=mux_hi*16+mux_lo;
outportb(base+2,val);
if (inportb(base+2) != val) {
      gotoxy(5,23); fputc(7,stdout);
      cprintf("FALHA NO MULTIPLEXADOR !");
      }
}

/*****************************************************************************************************
*    PROGRAMA  : RT.H  CABEÇALHO DO PROGRAMA RT_354.CPP                                              *
*    VERSAO    : 3.00        VERSAO FINAL                                                            *
*    DATA      : 18.11.1993   AUTOR: MARLON WILFRED GEMERTS                                          *
*    LINGUAGEM : TURBO C++ V3.01                                                                     *
*****************************************************************************************************
*  PROGRAMA COM DEFINICOES E DECLARACOES DAS VARIAVEIS E FUNCOES                                     *
*  UTILIZADAS NO PROGRAMA RT_354.CPP                                                                 *
*****************************************************************************************************/

#include <time.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <graphics.h>
#include <dos.h>
#include <dir.h>
#include <conio.h>
```

```
#include <alloc.h>

#define KEYBD        0x09

/* ** PROTOTIPOS DAS FUNCOES DEFINIDAS NO PROGRAMA RT_GRAPH.CPP ******** */
extern void denorm(void);
extern void graph_grid2(unsigned xmn,unsigned xmx,unsigned ymn,unsigned ymx,
     unsigned udiv,unsigned cor);
extern void graph_numb(unsigned xmn,unsigned xmx,unsigned ymn,unsigned ymx,
     unsigned udiv,float  min,float  max,float  sample,unsigned bias,unsigned cor);
extern int  graph_init(void);

/* ************ PROTOTIPOS DAS FUNCOES DEFINIDAS LOCALMENTE ************* */
void  install_interrupts   (void);
void  deinstall_interrupts (void);
void  interrupt newint09    (...);
void  interrupt (*oldint09) (...);
void  interrupt newintTs    (...);
void  interrupt (*oldintTs) (...);
void  graph_frame          (void);
char *pm(struct tm *t);
int   ad_loop (void);
void  ad_modify(int *flag);
void  update_params(void);

/* ********** DECLARACAO DAS VARIAVEIS GLOBAIS DEFINIDAS NO  ************************
  ******************** PROGRAMA RT_GRAPH.CPP  ************************************************
*/
```

```
extern unsigned scrx,scry;

extern unsigned W_title[4],W_graph[4],W_legend[4],Graphs[4];


/* ************ DECLARACAO/DEFINICAO DAS VARIAVEIS GLOBAIS ************** */

int dxx[16],exib[16],graphcolor[8],*xy_map,v_chns[16],r_chns[16],trig[16],xa[16],xx[16],ya[16];

int  ad_bias,adh,adl,ad_flag=0,apage,adt,base,chn,dx,freeze=0,i,loop,nzy,one_shot=0,ptr,

     r_ativa,r_flag,tpage,r_trig,r_modo,v_exib,vpage,x,y;

unsigned  limi[16],lims[16],huge *r_buf,huge *r_chn;

unsigned  count=0,offset=0,pos_N,pos_ptr,pre_N,pre_ptr,xmn,xmx,ymn,ymx;

float     sample,vmax[16],vmin[16],r_limi[16],r_lims[16];

unsigned  char kbch,imr;

char      chstr[16][10],strad[10],unit_name[16][10],var_name[16][10];

FILE     *in,*r_out;


char *mes[12] = {

"JANEIRO",

"FEVEREIRO",

"MARÇO",

"ABRIL",

"MAIO",

"JUNHO",

"JULHO",

"AGOSTO",

"SETEMBRO",

"OUTUBRO",

"NOVEMBRO",

"DEZEMBRO"

};
```

```
/*****************************************************************************
 * Programa : RT_GRAPH.CPP                                                   *
 * Revisao  : 3.00          VERSAO FINAL                                     *
 * Data     : 16/11/1993    Marlon W. Gemerts                                *
 *===========================================================*
 * Esse programa possui algumas funcoes graficas para serem utilizadas no   *
 * programa RT_354.CPP  (Programas de monitoracao em tempo real)             *
 *****************************************************************************/

#include <string.h>
#include <stdio.h>
#include <graphics.h>
#include <ctype.h>
#include <alloc.h>

void denorm(void);
void graph_grid2(unsigned xmn,unsigned xmx,unsigned ymn,unsigned ymx,
        unsigned udiv,unsigned cor);
void graph_numb(unsigned xmn,unsigned xmx,unsigned ymn,unsigned ymx,
        unsigned udiv,float  min,float  max,float  sample,unsigned bias,unsigned cor);
int   graph_init(void );

float     rx,ry;
int       gdriver,gmode;
unsigned scrx,scry;

unsigned W_title[4] = {  0,  0,639, 43 };
```

```
unsigned W_graph[4] = {  0, 43,639,255 };

unsigned W_legend[4] = {  0,255,639,319 };

unsigned Graphs[4] = { 55, 50,625,230 };


/* Funcao para denormalizacao de coordenadas especiais na tela */

void  denorm(void )

{

unsigned char i;

for (i=0; i<4; i=i+2) {

      W_title [i]=W_title [i]*rx; W_title [i+1]=W_title [i+1]*ry;

      W_graph [i]=W_graph [i]*rx; W_graph [i+1]=W_graph [i+1]*ry;

      W_legend[i]=W_legend[i]*rx; W_legend[i+1]=W_legend[i+1]*ry;

      Graphs  [i]=Graphs  [i]*rx; Graphs  [i+1]=Graphs  [i+1]*ry;

      }

}


/* Funcao para desenhar grade de grafico */

void  graph_grid2(unsigned xmn,unsigned xmx,unsigned ymn,unsigned ymx,

        unsigned udiv,unsigned cor)

{

unsigned i;

float    dxp,dyp;


setcolor(cor);

setlinestyle(SOLID_LINE,1,1);

dxp=(float ) (xmx-xmn)/udiv; dyp=(float ) (ymn-ymx)/udiv;

line(xmn,ymx,xmx,ymx); line(xmn,ymn,xmx,ymn);

line(xmn,ymx,xmn,ymn); line(xmx,ymx,xmx,ymn);
```

```
for (i=1; i<udiv; i++) {
    line(xmn-4,ymx+dyp*i,xmn,ymx+dyp*i);
    if (xmx+4<W_graph[2]-1) line(xmx+4,ymx+dyp*i,xmx,ymx+dyp*i);
    line(xmn+dxp*i,ymx-4,xmn+dxp*i,ymx);
    line(xmn+dxp*i,ymn+4,xmn+dxp*i,ymn);
    }
}

/* Funcao para colocar o valor real da variavel no eixo-y e no eixo-x do grafico */
void  graph_numb(unsigned xmn,unsigned xmx,unsigned ymn,unsigned ymx,
      unsigned udiv,float  min,float  max,float  sample,unsigned bias,unsigned cor)
{
unsigned i,h,w,zero,xdiv,ydiv;
float    dxp,dyp,dxf,dyf,temp;
char     numstr[10];

setcolor(cor);
settextstyle(SMALL_FONT,HORIZ_DIR,4);
if (min>max) { temp=min; min=max; max=temp; }
ydiv=2*udiv; xdiv=udiv;
if (udiv<1) udiv=1;
dxp=(float ) (xmx-xmn)/xdiv; dyp=(float ) (ymn-ymx)/ydiv;
dyf=(float ) (max-min)/ydiv; dxf=(float ) dxp*sample;
if (bias) {
    zero=(ymn+ymx)/2;
    if (min*max<0) {
        sprintf(numstr,"%3.0f",0.0); h=textheight(numstr);
        outtextxy(W_graph[0]+27,zero+h/2,numstr);
```

```
        }
    }
for (i=0; i<=ydiv; i++) {
    sprintf(numstr,"%3.0f",min+dyf*i); h=textheight(numstr);
    outtextxy(W_graph[0]+27,ymn-dyp*i+h/2,numstr);
    }
for (i=0; i<=xdiv; i++) {
    sprintf(numstr,"%.2g",dxf*i);
    h=textheight(numstr); w=textwidth(numstr);
    outtextxy(xmn+dxp*i-w/2+2,W_graph[3]-h-17,numstr);
    }
}

/* Rotina de inicializacao grafica */
int  graph_init(void )
{
detectgraph(&gdriver,&gmode);  if (graphresult()<0)   return -3;
initgraph(&gdriver,&gmode,""); if (graphresult()<0)   return -4;

setgraphmode(1);              /* modo grafico do video e' 1 */
scrx=(unsigned) getmaxx();
scry=(unsigned) getmaxy();    /* Dimensao da tela grafica   */
rx=(float ) (scrx+1)/640.;    /* Relacao entre tela real e  */
ry=(float ) (scry+1)/320.;    /* tela normalizada (CGA-LO)  */

setviewport(0,0,scrx,scry,1);
return 0;
}
```

BIBLIOGRAFIA

ASHMOLE,P.H. & Di CAPRIO,U. *Recorders for longer term system disturbances on the CEGB and ENEL systems.* IEEE Transactions on Power Systems, feb. 1988, p.137-141.

BRONZEADO,H.S. *Conceito, Avialação, Análise e controle dos distúrbios causados pela operação de consumidores especiais*. I ciclo de palestras de estudos e planejamento da operação de sistemas", Salvador-BA, 1986.

CHAUDHURI,N.;GHOSH,S.:GHOSH,A.M. *A technique for simultaneous measurement with a microcomputer,* IEEE Transactions on industrial electronics. maio 1985, p.144-119.

CHOI,S.S. & CARISON,P.J. *Development and Application of a power- system digital transient recorder,* IEEE proceeding, september 1987. p. 368-376.

CLEWES,T.W. et alli. *A digital transient recorder for power system monitoring and analysis,* IEEE Transactions on power apparatus and systems, jul. 1983, p.2186-2193.

COSTA,E.A.C. et alli. *Desenvolvimento de um registrador digital de perturbação nos sistemas elétricos.* 8 Congresso Brasileiro de Automatica, set. 1990. p.902-907.

COSTA,E.A.C. *Desenvolvimento de um registrador de perturbações nos sistemas elétricos,* Dissertação de mestrado, CCT/DEE, Campina Grande. jun. 1991.

D'AJUZ A. et alli. *Transitórios elétricos e coordenação de isometro - Aplicação em sistemas de potência de alta tensão*, Furnas, centrais elétrico S.A. 1987.

DALLY,J.W.;RILEY,W.F.;McCONNEL.K.G. *Instrumentation for engeneering measurements*. John Wiley & Sons, New York, 1984. 576 p.

DJOKIC,B.;BOSNJAKOVIC,P.;BEGOVIC,M. *New method for reactive power and energy measurement* IEEE Transactions on instrumentation and measurement, apr. 1992. p.280-285.

FERRARO,R.F. *Guia do programador para as placas EGA e VGA*. Ciência Moderna, Rio de Janeiro, 1991, 696 p.

KELLY-BOOTLE,S. *Dominando o turbo C*. 2 ed. Ciência Moderna, Rio de Janeiro, 1989, 610 p.

KRUTZ,R.L. *Interfacing techniques in digital design with emphasis on microprocessors*. John Wiley & Sons, New York,1988. 380 p.

KURE-JENSEN,J. & HANISCH,R. *Integration of steam turbine controls into power plant systems* IEEE Transactions on energy conversion, mar. 1991, p.177-187.

LEE,D.C. *System Disturbance Monitoring in Ontario Hydro*, IEEE Transactions on Power Systems, feb. 1988, pp. 144-148.

LEE,R.E. & OSBORN,R.H. *A micrcomputerbased data acquisition system for high impedance fault analysis.* IEEE Transactions on Power Apparatus and Systems, oct. 1985, pp. 2748-2753.

ODENBERG,R. & BRASKICH,B.J. *Measurement of voltage and current surges on the AC power line computer and industrial environments* IEEE Transactions on Power Apparatus and Systems, oct. 1985,pp. 2681-2691.

PINSON,J.L. *Designing screen interfaces in C*. Yourdon press, Englewood Cliffs, 1991, 267p.

ROSALES MONTERO,L.R. *Projeto, implementação e simulação digital do sistema de excitação e regulador de tensão para um microgerador síncrono de 3KVA*. Dissertação de Mestrado. DEE/CCT, Campina Grande, abr. 1991.

ROSALES MONTERO,L.R. *Desenvolvimento de controladores digitais para um modelo reduzido de sistemas elétricos de potência*. Tese de Doutorado. DEE/CCT, Campina Grande, em publicação.

SAY,M.G. *Eletricidade geral* Hemus, São Paulo, s.a.

SAWICKI,T.R. & REINPRECHT,R. *Aplicação de microprocessadores na industria*, Ed. Campus, Rio de Janeiro. 1985.

SCHILDT,H. *Linguagem C: guia prático e interativo.* Mc Graw-Hill, Rio de Janeiro, 1989, 363 p.

SHEINGOLD,D.H. *Analog-digital conversion handbook.* 2.ed. Analog Devices.Norwood,1976.

SIEMENS. *Model for demonstrating problems in transmission systems and power stations,* Siemens review, ago. 1969, p. 315- 321.

The IEEE Task Force on Instrumentation for system dynamic performance. *Instrumentation for monitoring power system dynamic performance.* IEEE Transactions on Power Systems, feb. 1987, pp. 145-152.

VENIKOV,V. *Transient processes in electrical power systems.* Mir.Moscow. 1977.