VALIDAÇÃO DE BASES DE CONHECIMENTO EM SISTEMAS ESPECIALISTAS BASEADOS EM REGRAS

Hercules Antonio do Prado

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

Prof. Antônio de Almeida Pinho,D.Sc.
(Presidente)

Profa. Cláudia Guerreiro Ribeiro do Vale,D.Sc.

Prof. Nelson Maculan Filho,D.Sc.

RIO DE JANEIRO, RJ - BRASIL
SETEMBRO DE 1989

PRADO, HERCULES ANTONIO DO
Validação de Bases de Conhecimento em Sistemas Especialistas Baseados em Regras [Rio de Janeiro] 1989
IX, 97 p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 1989)
Tese - Universidade Federal do Rio de Janeiro, COPPE
1. Sistemas Especialistas 2. Bases de Conhecimento 3. Validação I. COPPE/UFRJ II. Título (série)

A Alberto e Onézia.

AGRADECIMENTOS

Agradeço ...

... ao Prof. Antônio de Almeida Pinho pela orientação firme e dedicada na escolha e desenvolvimento do tema da tese e na elaboração desta dissertação.

... aos amigos da COPPE e da EMBRAPA, particularmente, Wamberto, Cláudia Sales, Adelina, Maurício, Genaro, Marcos Mota e Francisco Simplício, pelo apoio na condução de diversos assuntos relacionados com o curso.

... à EMBRAPA e à CAPES, cujo suporte financeiro e organizacional tornou possível a execução deste trabalho.

... à Sônia, da Seção de Registro da COPPE, pela maneira gentil e atenciosa com que sempre fui atendido.

... à Maria Zilda, pelos inúmeros "galhos quebrados" durante a minha estada no Rio.

... à Isa e ao Eugênio, pelo apoio oferecido nas minhas rápidas idas a Brasília.

... em especial, ao meu filho Daniel, por compreender que podemos estar juntos mesmo distantes, e à sua mãe, Bernadete, por acumular com exemplar eficiência as funções de pai durante a minha ausência.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.).

VALIDAÇÃO DE BASES DE CONHECIMENTO EM SISTEMAS ESPECIALISTAS BASEADOS EM REGRAS

Hercules Antonio do Prado

Setembro de 1989

Orientador: Prof. Antônio de Almeida Pinho

Programa: Engenharia de Sistemas e Computação

Neste trabalho abordamos o problema da validação de bases de conhecimento em sistemas especialistas baseados em regras, propondo algoritmos para detecção dos problemas mais comuns identificados nessas bases. No enfoque adotado, uma base de conhecimento submetida à validação é representada através de um grafo E/OU. Sobre este grafo, inicialmente, é aplicado um algoritmo existente na literatura para detecção de encadeamentos circulares. A partir do grafo que representa a base sem encadeamentos circulares é processada uma transformação com a geração do grafo de uma base equivalente. Em seguida são aplicados os algoritmos para detecção de condições desnecessárias, de regras incompatíveis e de regras inativáveis. Para os problemas apontados são sugeridas algumas formas de tratamento.

Abstract of Thesis presented to COPPE/UFRJ as partial fulfillment of the requirements for the degree of Master of Science (M.Sc.).

KNOWLEDGE BASES VALIDATION IN

RULE-BASED EXPERT SYSTEMS

Hercules Antonio do Prado

September, 1989

Thesis Supervisor: Prof. Antônio de Almeida Pinho

Department: Systems Engineering and Computing

In this work we face the problem of knowledge base validation in rule-based expert systems, purposing algorithms to detect the most common problems in these bases. In our approach, a knowledge base submitted to validation is represented by an AND/OR graph. To this graph, firstly, is applied an algorithm, already existing in the literature, for circular chaining detection. To the graph that represents the base without circular chaining a transformation is processed producing the graph of an equivalent base. Then algorithms are applied to detect unnecessary conditions, and incompatible and non-triggerable rules. Finally, we suggest some ways to treat the appointed problems.

Índice

I - Introdução 1

II - Inteligência Artificial e Sistemas Especialistas (SEs) 4

II.1 - O Contexto dos SEs 4

II.2 - Uma Definição de SE 6

II.3 - Características Gerais de um SE 8

II.4 - SEs no Processo de Elaboração do Conhecimento 10

II.5 - A Situação Atual 11

III - O Problema da Validação de Bases de Conhecimento (BCs) 14

III.1 - Descrição do Problema 14

III.2 - Revisão Bibliográfica 17

III.3 - Crítica dos Modelos Disponíveis 20

IV - Especificação do Ambiente - O BACO 21

IV.1 - Objetivos 21

IV.2 - Estrutura do Sistema 22

IV.3 - Uma Base de Conhecimento no BACO 24

IV.4 - O Grafo do Conhecimento 24

IV.5 - O Processo de Inferência 26

IV.6 - O Encadeamento Progressivo 27

IV.7 - O Encadeamento Regressivo 29

IV.8 - O Encadeamento Misto 29

IV.9 - Incompatibilidade de Condições 30

IV.10 - Consistência e Completeza no BACO 30

V - O Conceito de Base Reduzida 32

V.1 - Definições 32

V.2 - Construção da Base Equivalente Reduzida (BER) 37

V.3 - Complexidade do Algoritmo BER 42

VI - Validação das Bases Construídas sob o BACO 45

VI.1 - Obtenção de Bases sem Encadeamento Circular 45

VI.2 - Complexidade do Algoritmo NÍVEL 49

VI.3 - Obtenção de Bases Concisas 49

VI.4 - Complexidade do Algoritmo CONC 53

VI.5 - Obtenção de Bases sem Regras Incompatíveis 53

VI.6 - Complexidade do Algoritmo CONFL 59

VI.7 - Descrição das Regras Inativáveis 59

VI.8 - Relações entre uma BC e sua BER 61

VI.9 - Tratamento dos Problemas Detectados 65

VII - Uma Aplicação 68

VII.1 - Especificação da Base de Conhecimento 68

VII.2 - Detecção de Encadeamentos Circulares 71

VII.3 - Geração da Base Equivalente Reduzida 73

VII.4 - Detecção das Condições Desnecessárias 81

VII.5 - Detecção das Regras Incompatíveis 89

VIII - Conclusões 92

VIII.1 - Vantagens 92

VIII.2 - Desvantagens 93

VIII.3 - Sugestões de Linhas de Pesquisa 93

Referências Bibliográficas 95

Capítulo I

Introdução

O desenvolvimento do sistema DENDRAL, para análise de compostos orgânicos, no final dos anos 60, pela equipe do Dr. Feigenbaum em Stanford, pode ser considerado o passo mais significativo na criação da área de Sistemas Especialistas (SEs). A idéia de se criar um sistema baseado em conhecimento para efetuar inferências sobre um domínio específico apresentava-se como uma alternativa aos esforços frustrados anteriormente para criação de sistemas inteligentes de aplicação genérica.

Em meados dos anos 70, o desenvolvimento do sistema MYCIN para diagnóstico e recomendações de tratamento de meningite e doenças infecciosas do sangue, pela equipe do Dr. Shortliffe, também de Stanford, confirmou o potencial dos SEs e estabeleceu um outro marco de grande importância na gênese deste tipo de sistema: a partir da experiência com o desenvolvimento do MYCIN foi criado o primeiro *shell* (o EMYCIN), software que facilita o desenvolvimento de aplicações na área de SEs.

Desde então, diversos trabalhos foram desenvolvidos no sentido de se obter *shells* cada vez mais completos, com interfaces mais amigáveis, maior capacidade para armazenamento do conhecimento, tratamento de incertezas e outras características. Entretanto, existem ainda poucos instrumentos para que os responsáveis pela criação e manutenção de bases de conhecimento o possam fazê-lo de forma dirigida. Na ausência de tais instrumentos, a criação de bases de conhecimento ocorre sem que se possa garantir certos requisitos mínimos de qualidade do seu conteúdo.

Com o objetivo de criar facilidades para obtenção de bases de conhecimento de melhor qualidade propomos um modelo consolidado de rotinas para

validação destas bases, capaz de detectar:

- Encadeamentos circulares;
- Condições desnecessárias em regras ou cadeias de regras (subordinação de regras);
- Regras ou cadeias de regras incompatíveis; e
- Regras inativáveis por dependerem de premissas incompatíveis.

A presente proposta foi elaborada a partir de estudos sobre as bases de conhecimento criadas sob o **BACO** - Gerenciador de **BA**ses de **CO**nhecimento para o Desenvolvimento de Sistemas Especialistas do Tipo Diagnóstico, desenvolvido pelo Grupo de Inteligência Artificial da UFRJ. Sendo de natureza aplicativa, este trabalho aborda os aspectos teóricos estritamente necessários à sua fundamentação.

Inicialmente, no capítulo II - Inteligência Artificial e Sistemas Especialistas (SEs) - situamos os SEs no contexto da Inteligência Artificial (IA) e discorremos sobre alguns aspectos de caráter geral concernentes à área.

O capítulo III - O Problema da Validação de Bases de Conhecimento (BCs) - descreve o problema e discute os estudos existentes sobre o assunto.

O capítulo IV - Especificação do Ambiente - O BACO - detalha as características do BACO, incluindo a definição do grafo do conhecimento, estrutura utilizada para relacionar as condições e as regras do sistema e sobre a qual foram realizados os estudos aqui apresentados.

O capítulo V - O Conceito de Base Reduzida - detalha uma transformação do grafo do conhecimento em outro equivalente, com a identificação de condições desnecessárias e a criação de uma estrutura mais adequada aos demais estudos de validação.

No capítulo VI - Validação das Bases Construídas sob o BACO - são detalhados os procedimentos para tratamento do grafo do conhecimento.

O capítulo VII - Uma Aplicação - descreve uma base de conhecimento fictícia cuja estrutura apresenta os problemas a serem detectados e que é submetida aos procedimentos descritos, com o objetivo de avaliar a aplicabilidade dos mesmos.

Finalmente, no capítulo VIII - Conclusões - são discutidos os resultados obtidos, são apresentadas algumas vantagens e desvantagens da utilização das bases reduzidas, são analisados alguns aspectos da utilização de heurísticas na busca sobre o grafo do conhecimento, e são apresentadas sugestões para o desenvolvimento de pesquisas nesta área.

Capítulo II

Inteligência Artificial e Sistemas Especialistas (SEs)

"... e o saber se multiplicará." *Bíblia Sagrada, Daniel, Cap. 12, Ve. 4d*

Neste capítulo situamos os SEs na área de Inteligência Artificial, apresentamos uma definição, discutimos as suas características gerais e descrevemos a situação atual.

II.1 - O Contexto dos SEs

"Inteligência Artificial é a parte da Ciência da Computação interessada na criação de sistemas computacionais inteligentes, i.é, sistemas que exibem características associadas ao comportamento inteligente humano - compreensão de linguagem, aprendizado, raciocínio, resolução de problemas, e assim por diante."

Esta definição de IA, por BARR *et alia* (1981), mostra o ambiente no qual se insere a pesquisa em SEs. Entretanto, para se obter uma definição clara desta área de investigação é útil um breve histórico das tentativas de se inserir inteligência no computador, desde a criação deste na década de 50 até o momento em que se consolidam os conceitos básicos da área. Tal histórico vamos encontrar em FORSYTH (1984), onde o autor estabelece marcos que discutimos a seguir.

Inicialmente, muitos pesquisadores orientaram seus trabalhos inspirados na idéia, baseada nos estudos de Norbert Wiener e Warren McCulloch sobre Cibernética, de que seria possível simular no computador um sistema de neurônios sobre o qual se poderia executar um programa de treinamento, através de estímulos e de punições, e torná-lo inteligente de fato. As expectativas não foram satisfeitas devido ao elevado grau de sofisticação do que se queria simular -

o cérebro humano possui cerca de 10^{12} neurônios - e pelas enormes restrições de software e hardware. Contudo, apesar do insucesso destas pesquisas em termos práticos na época, a idéia de se criar um sistema neuronal para representação do conhecimento foi retomada no início dos anos 80 com o trabalho de J. Hopfield.

Na década de 60 Allen Newell e Herbert Simon, de Carnegie-Mellon, estabeleceram um importante marco com a criação do *GPS - General Problem Solver*. Considerando o fato de que as funções de manipulação simbólica, às quais supostamente o raciocínio humano poderia ser reduzido, são tarefas possíveis de serem realizadas por um computador, eles abordaram a resolução de um problema pela subdivisão deste em subproblemas e assim por diante até se chegar ao nível de problemas triviais que, resolvidos, levariam à solução do problema inicial. Desta forma e, admitindo que existam diversas formas de se decompor um problema, a solução era procurada exaustivamente entre todos os estados possíveis do problema* . O modelo criado, apesar do seu poder de resolução, se restringia a classes de problemas muito triviais, sem grande interesse prático.

O próximo passo em direção à criação de um paradigma de SE seria dado no final dos anos 60 em Stanford pela equipe do Prof. Feigenbaum. Contrariando a tendência anterior de se procurar desenvolver mecanismos genéricos no sentido de serem aplicados a qualquer área, a equipe se fixou na idéia de estreitar o domínio do conhecimento, enfocando áreas bem específicas. O modelo criado foi o DENDRAL, sistema utilizado até hoje na área de Química para análise de compostos orgânicos.

Dentro da filosofia do DENDRAL, foi desenvolvido em seguida, também em Stanford por SHORTLIFFE *et alia* (1984), o sistema MYCIN para diagnóstico e recomendações para tratamento de meningite e doenças infecciosas do sangue. Com uma arquitetura diferente do seu antecessor, o MYCIN foi o primeiro sistema com uma separação clara entre a base de conhecimento e o mecanismo de inferência. Esta característica deu grande flexibilidade ao sistema e permitiu sua extensão a outras especialidades, levando à criação do primeiro *shell*, o EMYCIN, cuja principal característica é possuir uma estrutura padronizada para

* O conjunto destes estados é chamado "espaço de estados" do problema.

gerenciamento de bases de conhecimento distintas, constituindo-se numa valiosa ferramenta para o desenvolvimento de SEs. A independência entre o conhecimento e o mecanismo de inferência existente em tal estrutura permitiu também uma grande flexibilidade para atualização deste conhecimento.

II.2 - Uma Definição de SE

A expressão Sistemas Especialistas tem sido utilizada para designar softwares que têm por objetivo atuar como consultores em áreas do conhecimento bastante restritas no mesmo nível, ou próximo, do especialista humano. O uso do têrmo *Especialistas* desta forma é contestado por DREYFUS *et alia* (1986) que, para justificar sua argumentação definem cinco níveis de habilidade nas várias áreas em que o ser humano pode atuar:

1. *Principiante*, no qual a pessoa aprende os fatos mais importantes sobre a área, além de regras de ação sobre estes fatos.

2. *Principiante Avançado*, que são os principiantes com muita experiência em situações reais, mas que continuam baseando suas ações estritamente nas regras aprendidas.

3. *Competente*, cuja principal diferença para os níveis anteriores é o grau de envolvimento. Enquanto os primeiros sentem-se pouco responsáveis por estarem simplesmente aplicando regras, o competente pode fazer opções contrárias a estas regras, assumindo a responsabilidade pela sua escolha.

4. *Proficiente*, no qual a maneira de encarar as situações não se dá, na maioria das vezes, por opção consciente de se adotar um ou outro caminho. A solução para um dado problema é tentada primeiro pela busca na memória de experiências similares para as quais ja deu solução, sem se deter em analisar os componentes do problema atual. A esse comportamento os autores chamam *intuição* ou *know how*.

5. *Especialista*, que sabe conscientemente o que fazer, devido à sua maturidade e capacidade de entendimento. A habilidade para se solucionar problemas no

seu campo de trabalho é como se fosse uma extensão do seu próprio corpo. Para ilustrar esta habilidade é citado o exemplo do piloto muito experiente que não se considera pilotando um avião mas sim voando.

Assim, de acordo com os autores, a denominação correta seria Sistemas *Competentes* pelo fato dos sistemas criados até hoje se aproximarem mais do nível de habilidade de competentes do que de especialistas. Naturalmente, o que pesa nesta classificação é o fato de estes sistemas poderem fazer opções muitas vezes contrárias às regras básicas através de heurísticas* , não havendo a característica de responsabilidade sobre estas opções.

Não é discutida a utilidade destes sistemas que, como vimos, são aplicados a vários setores da atividade humana, mas é feito um alerta sobre o risco de "empregados principiantes passarem a ver o conhecimento ao nível do especialista como função de grandes bases de conhecimento e massas de regras nas quais o programa confia. Tais empregados falharão em avançar além do nível de competentes, e a empresa poderá finalmente descobrir que a sua fonte de conhecimento especializado e sabedoria terá secado."

A despeito desta argumentação, notamos que a denominação Sistemas Especialistas já é consagrada mundialmente e, portanto, será a expressão utilizada por nós. Adotamos, por ser normalmente aceita por pesquisadores da área, a definição de Feigenbaum transcrita em HARMON *et alia* (1985):

"Sistemas Especialistas são programas inteligentes de computador que utilizam conhecimento e procedimentos de inferência para resolver problemas difíceis o bastante para requerer significativa experiência humana para solucioná-los. O conhecimento necessário para executar tarefas a este nível mais os procedimentos de inferência usados podem ser considerados como um modelo de perícia dos melhores praticantes da área. O conhecimento de um Sistema Especialista consiste de fatos e heurísticas. Os fatos constituem o corpo da informação que é amplamente compartilhada e publicamente disponível e, geralmente, são estabelecidos por especialistas da área. As heurísticas são regras de bom julgamento pouco discutidas

*"Heurísticas", aqui, referem-se a funções matemáticas baseadas na experiência do especialista no domínio considerado com o fim de orientar a busca de soluções no espaço de estados do problema.

e, na maioria das vezes, pessoais, que caracterizam o nível da perícia considerada para a tomada de decisão em uma área. O nível de desempenho de um Sistema Especialista é , principalmente, função da quantidade e qualidade do conhecimento que ele possue."

II.3 - Características Gerais de um SE

Numa primeira abordagem discutimos as características de um SE por contraste com os sistemas convencionais, explorando os principais pontos que diferenciam a primeira da segunda classe de sistemas.

Um SE manipula conhecimento, não só dados, e utiliza formas simbólicas de representação e processamento, enquanto que um sistema convencional efetua apenas manipulações numéricas. A solução de um problema submetido a um SE é buscada pelo sistema, por algoritmos que percorrem o espaço de estados do problema através de processos heurísticos. Desta forma um SE não tem soluções prontas para os problemas, mas sim elementos que podem levar a alguma solução; num sistema convencional o programa já é a própria solução definida através de um algoritmo. O conhecimento num SE é legível, fácilmente compreensível e alterável, devido ao fato de que ele se aproxima da forma como os especialistas humanos procedem. Na busca de soluções um SE pode utilizar-se de conhecimento sobre si mesmo (meta conhecimento) para dirigir os processos de busca heurística por caminhos mais promissores. Estas últimas duas características inexistem num sistema convencional.

Sob outra ótica os SEs caracterizam-se por sua arquitetura básica, em geral aceita como padrão, composta de uma base de conhecimento, uma base de dados, um mecanismo de inferência e uma interface com o usuário. A interrelação entre os componentes de um SE e o usuário é ilustrada pela figura 1.

Construído sobre tal estrutura básica um SE deve ainda ser capaz de :

- Lidar com problemas com nível de complexidade que normalmente requerem

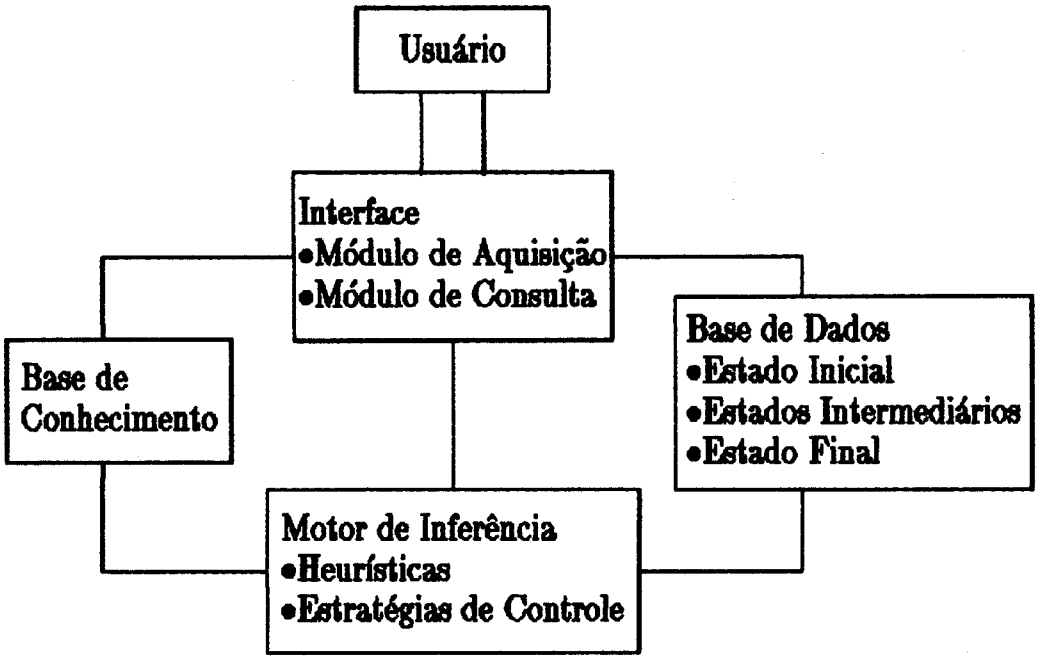

Figura 1: Arquitetura Básica de um SE

um conhecimento associado à experência humana em áreas muito especializadas.

- Apresentar alta performance em têrmos de velocidade e confiabilidade de modo a ser considerado pelo usuário como uma ferramenta útil.
- Explanar e justificar soluções e recomendações de forma a convencer o usuário de que o raciocínio utilizado é correto.

A seguir passamos a discutir os componentes da estrutura apresentada.

Base de Conhecimento (BC) - na base de conhecimento está contida a experiência que se captou de especialista(s), ou de outras fontes, no domínio de conhecimento ao qual se aplica o SE. Nesta base o conhecimento é representado, mais popularmente, através de regras de produção (**SE** *condição* **ENTÃO** *ação*). Extremamente flexível, esta forma de representação modela o raciocínio do especialista numa cadeia de regras que o sistema percorre segundo o tipo de

encadeamento adotado no motor de inferência. Outras formas de representação existem e são amplamente discutidas nas publicações sobre SE. O exemplo abaixo mostra um trecho do que poderia ser uma BC, escrita em regras de produção, de um SE usado por um órgão de desenvolvimento de sistemas de um grupo empresarial para a definição de prioridades para atendimento:

SE *empresa-não-coligada*
ENTÃO *só-sistema-administrativo*

SE *usuário-administrativo*
ENTÃO *sistema-administrativo*

SE *usuário-gerente*
ENTÃO *sist-apoio-a-decisão*

SE *sist-apoio-a-decisão* **E** *só-sistema-administrativo*
ENTÃO *negar-pedido*

Base de Dados - aqui ficam armazenados os dados iniciais de um determinado problema, os fatos concluídos através de inferências sobre os dados iniciais e o estado atual do problema. Através destes registros pode-se fazer o rastreamento do raciocínio desde as premissas até as conclusões, permitindo assim a justificação das conclusões.

Motor de Inferência - é o conjunto de rotinas que realiza as operações de busca e comparação. Durante o processo de busca, o motor de inferência utiliza, normalmente, as técnicas de encadeamento progressivo que vai dos dados iniciais às conclusões e/ou encadeamento regressivo, que toma as conclusões como hipóteses e vai voltando aos dados iniciais a fim de prová-las.

Interface com o Usuário - neste módulo é realizada toda a comunicação com o usuário para obtenção dos dados necessários ao processo de inferência e para informação das ocorrências durante este processo.

II.4 - SEs no Processo de Elaboração do Conhecimento

Além do seu poder multiplicador do conhecimento especializado, raro e de alto custo, tornando-o disponível no momento e lugar necessários, os SEs oferecem

vantagens adicionais através do estímulo ao processo de elaboração deste conhecimento. WEISS *et alia* (1988) identificam tal ganho como sendo a formalização do conhecimento especializado e o questionamento decorrente da integração de diversas fontes de conhecimento.

Muito do conhecimento de um especialista extrapola os limites do que existe formalizado na sua respectiva área de atuação, definindo um tipo de saber associado à sua experiência ou ao seu sentimento, algo que não é facilmente formulado em têrmos computacionais. Ao explicitar o conhecimento sobre como ele resolve problemas no dia-a-dia, o especialista é levado a refletir sobre a formalização deste conhecimento, criando alternativas factíveis de serem usadas na solução de problemas sob as mesmas circunstâncias. Ainda que esta formalização não seja alcançada de imediato, o SE capta o conhecimento e forma uma base sobre a qual isto poderá ser tentado de novo, futuramente.

Quanto ao questionamento que surge com a integração de diversas fontes de conhecimento, sejam especialistas ou publicações, isto decorre do confronto de proposições como idéias, princípios ou técnicas ainda não aceitos cientificamente, retratando antes a experiência acumulada pelos autores das proposições. Assim, o cotejo entre diversas opiniões sobre um determinado problema possibilita a comparação e o julgamento de alternativas, aprofundando o conhecimento no domínio considerado.

II.5 - A Situação Atual

Um importante impulso recente dado à área de SE deve-se ao lançamento, em outubro de 1981, do chamado "projeto japonês de 5a. geração" que tem como principal característica o fato de ser dominado pelo software e, segundo LUCENA (1987), pelo "uso não convencional do software." Entende-se por "uso não convencional" a produção de software por software e o uso de sistemas inteligentes para obtenção de formas de comunicação cada vez mais próximas do usuário.

Neste contexto os SEs ocupam lugar tanto como apoio ao desenvolvimento do projeto, através de sistemas para atuar em áreas altamente es-

pecializadas como projeto de circuitos, quanto como beneficiário dos resultados alcançados com a tecnologia de VLSI* e de paralelismo. Particularmente, com o paralelismo, espera-se obter um grande impulso na construção de interfaces em linguagem natural, com ampla aplicação em SEs.

Além disto, o reconhecimento por parte dos usuários da aplicabilidade cada vez maior dos SEs, aliado à crescente demanda por conhecimento especializado, tem justificado um incremento significativo na produção de aplicações de SEs. Na tabela 1 são relacionados alguns SEs conhecidos e suas respectivas áreas de aplicação.

Exemplos de Sistemas Especialistas	
Nome	Aplicação
MYCIN	Diagnóstico e recomendações de tratamento de meningite e doenças infecciosas do sangue
PUFF	Diagnóstico e recomendações de tratamento de doenças pulmonares
DENDRAL	Identificação da estrutura de compostos orgânicos
XCON e XSEL	Configuração de computadores VAX e seleção de componentes eletrônicos
PROSPECTOR	Localização de jazidas minerais
EL	Analisador de circuitos eletrônicos
FOLIO	Orientação de investimentos

Tabela 1: Alguns SEs Conhecidos

A relação apresentada é representativa do leque de aplicações desenvolvidas em SEs e dela depreende-se que tais aplicações, em sua maioria, atendem à categoria de problemas conhecida como de classificação. A solução de problemas deste tipo consiste em, mediante a descrição de objetos, enquadrá-los em classes de interesse. Diagnóstico médico é um caso típico de problema de classificação.

A par do crescimento da área de SEs, com a experiência acumulada no desenvolvimento de aplicações, tem se formado um corpo de princípios, ferramentas e técnicas para desenvolvimento de novas aplicações. Neste sentido WEISS *et alia* (1988) sugerem algumas etapas, resumidas a seguir, que podem ser

* VLSI do inglês, significa integração de circuitos em altíssima escala.

utilizadas como linhas gerais no processo de criação de SEs.

1. **Etapa Inicial do Projeto da BC**: Consiste na definição do problema, com sua descrição pormenorizada, identificação das hipóteses, dados, conclusões, linhas de raciocínio e estratégias para evolução nestas linhas, e considerações sobre a representação do problema, incluindo a organização dos componentes do conhecimento na base.

2. **Etapa de Desenvolvimento e Teste do Protótipo**: Aqui os autores reconhecem que a chave do sucesso na criação de um SE é começar de maneira simples e avançar até um sistema mais complexo. Assim, recomendam a seleção de uma amostra representativa de todo modelo com raciocínios que sejam suficientemente simples de se testar através de um protótipo. Uma vez que o protótipo produza um raciocínio aceitável, pode ser expandido para incluir outras variantes do problema que deve interpretar.

3. **Refinamento e Generalização da BC**: Nesta etapa, que pode tomar um tempo considerável, ocorre a consolidação do sistema como uma estrutura funcional e confiável. Um ganho subjacente a esta etapa é o que foi tratado na seção II.4: com a interação entre o especialista e o sistema, o primeiro é levado a uma reflexão permanente sobre os seus conhecimentos podendo, conseqüentemente, reformulá-los, pela inserção de novos conceitos e correção ou mesmo refutação de outros. Desta forma o sistema funciona como um instrumento de aprendizado para o especialista.

Capítulo III

O Problema da Validação de Bases de Conhecimento (BCs)

Neste capítulo identificamos as principais deficiências encontradas em BCs, descrevemos a nossa abordagem na verificação destas e relatamos os trabalhos encontrados na bibliografia sobre o assunto.

III.1 - Descrição do Problema

Durante a evolução dos SEs muito do esforço empregado tem sido no sentido de se *aprender* a desenvolver sistemas, procurando identificar estruturas básicas e caminhos para se criar um SE. Poucos resultados, entretanto, têm sido alcançados na obtenção de ambientes para aquisição de conhecimento dentro de níveis de qualidade aceitáveis. As formas como tem ocorrido a transferência da capacidade de solucionar problemas de especialistas humanos para programas de computador dão margem ao surgimento de características indesejáveis, que prejudicam a aceitação destes programas como instrumentos úteis na solução de problemas. Neste trabalho enfocamos as mais comuns destas características, propondo, todavia, uma abordagem mais abrangente do que a normalmente utilizada para sua detecção.

Embora os problemas típicos de BCs sejam conceitualmente semelhantes na maioria das bases, observamos que a definição de um modelo para detecção destas falhas é dependente da representação do conhecimento utilizada; daí decorre uma maior ou menor facilidade na manipulação destas falhas. Devido a este fator, desenvolvemos nossos estudos especificamente para BCs escritas através de regras de produção que utilizem raciocínio preciso. A idéia é definir um modelo com estas características mínimas, criando condições para extensões futuras, com características mais complexas. Utilizou-se o BACO como ambiente operacional pelo fato deste software, além de criar BCs nas condições já especificadas, traz a facilidade de generalização das estruturas de grafos, forma utilizada para organizar

as regras de produção internamente.

As características que invalidam uma BC por nós abordadas são: encadeamento circular, condições desnecessárias, regras incompatíveis e regras inativáveis. A seguir descrevemos cada uma destas características. Note-se que só apresentamos uma definição mais formal das mesmas no capítulo VI, pois antes precisamos especificar o modelo para o qual queremos defini-las, o que é feito no próximo capítulo. As regras que ilustram a exposição são de uma BC fictícia para tomada de decisão no caso de ocorrer algum problema com um automóvel.

1. **Encadeamento circular** - ocorre quando numa cadeia de regras as premissas (ou parte delas) da primeira coincidem com a conclusão da última. Desta forma, dependendo do grau de coincidência das condições que fecham a cadeia, o motor de inferência pode entrar num ciclo sem fim quando tentar acessar alguma condição da cadeia. Se a coincidência for total o encadeamento circular é inevitável; se a coincidência for apenas parcial, tal poderá não ocorrer. Exemplo:

COINCIDÊNCIA PARCIAL	COINCIDÊNCIA TOTAL
SE *carro-com-defeito* **E** *não-há-guincho* **ENTÃO** *colocar-triângulo*	**SE** *carro-com-defeito* **ENTÃO** *colocar-triângulo*
SE *colocar-triângulo* **ENTÃO** *chamar-mecânico*	**SE** *colocar-triângulo* **ENTÃO** *chamar-mecânico*
SE *chamar-mecânico* **ENTÃO** *carro-com-defeito*	**SE** *chamar-mecânico* **ENTÃO** *carro-com-defeito*

Suponhamos que o carro esteja com defeito; na coincidência parcial não haverá problema, caso haja guincho, enquanto que na total o motor de inferência entrará num ciclo sem fim.

2. **Condições desnecessárias** - podem ocorrer pela simples repetição de regras ou, ainda, pela subordinação de regras, causando uso desnecessário de recursos de armazenamento, além de uma queda de performance na busca. Exemplo de subordinação de regras:

REGRA 1

SE *pneu-furado*
  E *não-chove*
  E *roupa-não-é-clara*
ENTÃO *trocar-pneu*

REGRA 2

SE *pneu-furado*
ENTÃO *trocar-pneu*

Sempre que a regra 1 for disparada a 2 também será. Note que a regra 1 é mais restritiva que a 2. A decisão sobre qual regra deve ser mantida depende da necessidade de se restringir mais ou menos as condições necessárias para a ativação da regra.

3. **Regras incompatíveis** - ocorrem quando conclusões mutuamente exclusivas podem ser obtidas simultâneamente. É necessário diferenciar dois tipos de incompatibilidade: aquele com conclusões contraditórias explícitas (p.ex. *trocar-o-pneu* e *não-trocar-o-pneu*) e aquele com conclusões diferentes (*trocar-o-pneu* e *chamar-o-borracheiro*). A incompatibilidade no primeiro caso é evidente e no segundo pode ocorrer ou não, dependendo se as conclusões são mútuamente exclusivas ou não.

4. **Regras inativáveis** - são regras que dependem de premissas incompatíveis e que, portanto, não podem ser disparadas. Esta característica, embora semelhante, não se encaixa no caso definido no item (3). Neste caso podemos dizer que existe uma incompatibilidade entre a conclusão de uma regra e as suas premissas. Exemplo:

SE *pneu-furado*
  E *chove*
ENTÃO *chamar-borracheiro*

SE *pneu-furado*
  E *tem-pressa*
ENTÃO *trocar-pneu*

SE *trocar-pneu*
  E *chamar-borracheiro*
ENTÃO *aguardar-consertar-pneu*

Estas regras, aparentemente, não apresentam problemas. Entretanto, se as conclusões das duas primeiras forem mutuamente exclusivas e isto estiver especificado, a regra que conclui *aguardar-consertar-pneu* nunca será disparada.

Qualquer decisão sobre em que ponto interromper um encadeamento circular, a supressão de condições em regras, a reestruturação de regras ou a resolução de conflitos cabe ao engenheiro do conhecimento* , juntamente com o especialista.

III.2 - Revisão Bibliográfica

O primeiro esforço no sentido de se validar BCs deve-se a DAVIS (1979), que desenvolveu o programa TEIRESIAS, módulo de aquisição do conhecimento implementado para o MYCIN e depois extendido para o EMYCIN. Foi implementado um mecanismo para verificação das regras que, a partir de uma base montada, cria uma tabela para mostrar quais atributos são usados para concluir outros. Com base nesta tabela, as novas regras são verificadas para se identificar atributos referenciados em suas premissas que não tenham aparecido como conclusão de alguma regra ou como condição inicial. O programa permite também efetuar o rastreamento do raciocínio, caso surja alguma conclusão suspeita, e a identificação de encadeamento circular. Neste ponto é interessante notar o tratamento dado a este problema pelo MYCIN: nele são diferenciadas auto-referência, que ocorre quando uma regra utiliza a mesma condição como premissa e como conclusão, e cadeias onde existem mais de uma regra. As primeiras são intencionais e fazem parte do tipo de raciocínio particular do sistema, enquanto que as segundas são indesejáveis. O tratamento dado é ilustrado através da seguinte cadeia de raciocínio:

$$\cdots D \rightarrow A \rightarrow B \rightarrow C \rightarrow D \cdots$$

"O MYCIN resolve este problema marcando todos os parâmetros que estão sendo verificados pelo seu mecanismo de busca. O sistema simplesmente

* A expressão *engenheiro do conhecimento* designa o profissional que realiza a comunicação entre as fontes do conhecimento (especialistas, livros, etc) e um sistema especialista.

ignora a regra se um dos parâmetros checados na sua premissa já foi marcado." Supondo que esta meta fosse D, o resultado do processo seria:

$$A \rightarrow B \rightarrow C \rightarrow D$$

O maior mérito do TEIRESIAS em termos de validação está no fato de ser o primeiro resultado significativo neste assunto. Na verdade ele não é um programa específico para validação, sendo esta tarefa apenas um item dentre outros abrangidos pelo programa, que inclui módulos de correção, justificação e revisão das regras.

SUWA *et alii* (1984) apresentaram um programa para verificação de completeza e consistência da BC do ONCOCIN, um sistema baseado em regras da Clínica de Oncologia do Centro Médico de Stanford. Neste trabalho são abordados os seguintes problemas, tratados pelos autores como inconsistências:

- **Conflito**: duas regras com as mesmas premissas e conclusões diferentes sobre o mesmo atributo;
- **Redundância**: duas regras com as mesmas premissas e mesmas conclusões; e
- **Subordinação**: duas regras tendo os mesmos resultados, sendo que uma contém mais restrições do que a outra. Sempre que a regra mais restritiva ocorre a outra também ocorre, gerando um tipo de redundância.

Quanto à completeza, o trabalho aborda os casos de regras cujas premissas não aparecem nem como condições iniciais nem como conclusões de outras regras, além de relacionar as combinações de atributos não utilizadas.

Para cumprir seu objetivo, o programa separa a BC em dois conjuntos disjuntos de regras com base nos atributos referenciados nas premissas e nas conclusões. Em seguida monta uma tabela contendo todas as possíveis combinações de valores dos atributos. Com base nesta tabela é que são identificados os problemas descritos. Devido a esta abordagem muitas combinações sem sentido

podem ser confundidas com problemas de completeza. Um exemplo é dado no próprio trabalho: "não existem machos grávidos (pela própria natureza) e nem crianças alcoólatras (no domínio considerado)". Como o programa não dispõe de um razoável conhecimento do domínio do problema, o resultado é que ele poderá apresentar uma grande quantidade de erros potenciais, dos quais apenas alguns são reais. Este programa verifica as regras dinamicamente, i.é, à medida que estas são carregadas, e seu principal problema é o escopo da verificação, restrito a pares de regras, não considerando problemas ao nível de cadeias de regras.

NGUYEN *et alii* (1987) apresentaram o CHECK, uma extensão do verificador do ONCOCIN para o LES (*Lockheed Expert System).* Este software analisa a BC estaticamente, i.é, após todos os elementos terem sido carregados. Nesta análise são detectados, além dos problemas abordados no seu antecessor, condições desnecessárias, conclusões inacessíveis, atributos ilegais e raciocínio circular. Apesar das melhorias introduzidas pelo CHECK, persistem as limitações em têrmos do escopo da verificação.

Outro software existente é o TIMM (*The Intelligent Machine Model)* citado por Nguyen. Nele a definição de inconsistência inclui, além das regras com premissas iguais e conclusões dos mesmos atributos com valores diferentes, aquelas com premissas iguais e conclusões de atributos diferentes. Além destas inconsistências, os problemas de completeza são verificados por um processo randômico baseado no exame de similaridade entre combinações de premissas não utilizadas em regras e aquelas para as quais existem regras definidas.

Citado no mesmo trabalho, há o INSPECTOR, ferramenta desenvolvida para o KES (*Knowledge Engineering System*), que permite a identificação de recursão, o mesmo que auto-referência no MYCIN, e atributos referenciados como premissa sem aparecer como conclusão em alguma regra.

Em DAMSKY *et alia* (1987) é apresentado um sistema baseado na lógica do cálculo de predicados, que utiliza o conceito de *escopo* de uma variável, predicado ou regra, que é aplicado na detecção de ciclos. É apresentado também um algoritmo que detecta inconsistências através da dedução do predicado e da

sua negação.

PINHO *et alia* (1988) discute o encadeamento circular e, para sua detecção, utiliza o conceito de *nível* de uma condição. Além desta aplicação, o nível pode ser utilizado como heurística, para estabelecer prioridades entre regras, quando houver mais de uma em condições de ser disparada. São introduzidas as classes de incompatibilidade, como um conceito mais abrangente que a simples negação de uma condição, para identificação de conflitos e de conclusões inacessíveis.

Como contribuição da IBM, o ESE (*Expert Systems Environment*), lançado recentemente no Brasil, faz a verificação de valores de atributos referenciados em confronto com o universo de valores possíveis, através do seu módulo de aquisição do conhecimento.

III.3 - Crítica dos Modelos Disponíveis

Uma crítica geral às soluções apresentadas é que não existe um modelo consolidado de validação que inclua todos os tipos de problemas levantados. Além disto, com exceção de DAMSKY *et alia* (1987) e PINHO *et alia* (1988), não há uma preocupação em se definir precisamente os problemas que uma BC pode apresentar. Exemplo disto é a definição de redundância como inconsistência.

Capítulo IV

Especificação do Ambiente - O BACO

Neste capítulo fazemos uma descrição sucinta do BACO - Gerenciador de Bases de Conhecimento para o Desenvolvimento de Sistemas Especialistas do Tipo Diagnóstico, incluindo seus objetivos, estrutura básica e mecanismo de inferência. O material para este capítulo foi retirado de PINHO (1988).

IV.1 - Objetivos

Concebido pelo Grupo de Inteligência Artificial - GIA da UFRJ, o projeto BACO visa prover o usuário de SEs de uma estrutura básica para desenvolvimento de suas aplicações, com as seguintes características:

- Portabilidade, ou seja, a propriedade de uma aplicação ser implementada em equipamentos diferentes daquele no qual ela foi desenvolvida;
- Flexibilidade inferencial, proporcionando ao usuário opções para (a) o tipo de raciocínio empregado, preciso ou impreciso, e no segundo caso do método de tratamento de incerteza, e (b) o encadeamento, regressivo, progressivo ou misto, utilizado na busca de soluções;
- Interface amigável com os usuários, com suporte para (a) criação das BCs, (b) projeto de diálogos e (c) explanação e justificação das conclusões; e
- Facilidade para interfaceamento com bancos de dados e com outros sistemas.

Abordado, inicialmente, como uma ferramenta acadêmica, o BACO, através de experiências como o uso de heurísticas, análise de métodos de tratamento de incerteza e outras, levadas a têrmo pelos pesquisadores do GIA, caminha para sua consolidação como um software útil à comunidade de informática.

IV.2 - Estrutura do Sistema

O BACO possui na sua estrutura as seguintes entidades:

- Arquivo de Fatos;
- Arquivo de Regras;
- Conjunto de Rotinas Auxiliares; e
- Conjunto de Motores de Inferência.

Cada entidade tem seu papel definido da seguinte forma:

- **ARQUIVO DE FATOS** - contém as representações de todos os fatos possíveis de ocorrer no sistema na forma de triplas O-A-V (Objeto-Atributo-Valor) com a indicação da sua ocorrência ou não. Para isso, existe um campo adicional para o fator de certeza (FC), que expressa a medida do conhecimento que temos da veracidade do fato. Este arquivo recebe um tratamento para identificação de OAVs não iniciais para as quais não existem regras que as concluam.
- **ARQUIVO DE REGRAS** - aqui o conhecimento referente a cada aplicação está representado na forma de regras de produção, segundo o seguinte esquema:

SE $(OAV)_1$
E $(OAV)_2$
...
E $(OAV)_n$
ENTÃO $(OAV)_{n+1}$

$(OAV)_i$, $i = 1, 2, \ldots, n$, são os antecedentes da regra e $(OAV)_{n+1}$ o conseqüente. Quando todos os antecedentes de uma regra têm status V, o motor de inferência pode concluir seu conseqüente.

- **ROTINAS AUXILIARES** - são rotinas utilizadas para: (a) construção dos arquivos de **FATOS** e **REGRAS**, (b) estruturação do diálogo com o usuário, (c) análise das condições do arquivo de **REGRAS**, (d) adequação do gerenciador a cada aplicação, (e) consulta, e (f) edição do conhecimento em forma gráfica.

- **MOTORES DE INFERÊNCIA** - os motores de inferência diferem quanto à forma de encadeamento utilizado: regressivo *(backward chaining)*,progressivo *(forward chaining)* ou misto, à utilização de heurísticas e ao tipo de raciocínio empregado: preciso ou impreciso. No caso de raciocínio impreciso é oferecida a opção para tratamento de incertezas pela Teoria da Confirmação (forma utilizada no MYCIN), Teoria da Evidência (Dempster-Schaffer) ou Teoria da Possibilidade (*fuzzy sets*). Atualmente, a análise, adequação e implementação destes métodos no BACO constitui o objeto da tese de mestrado de SESCONETTO BORGES (1989).

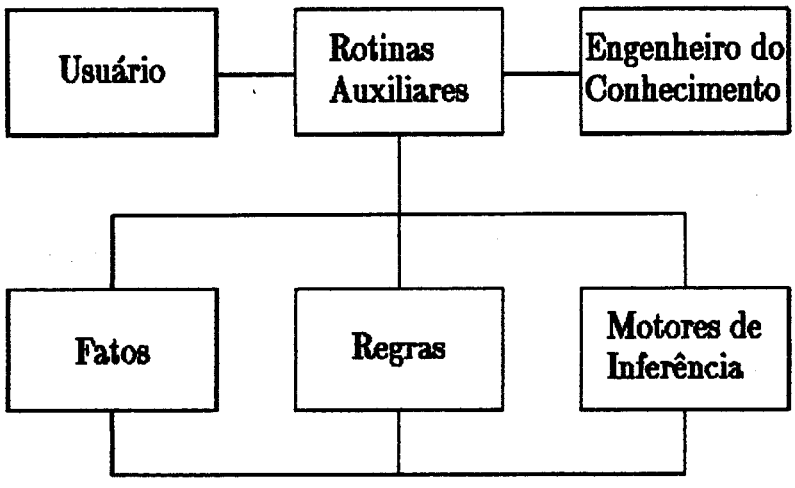

Figura 2: Estrutura do BACO

Existem dois modos de operação do sistema: o modo engenheiro do conhecimento e o modo usuário. Através do primeiro o engenheiro do conhecimento realiza as seguintes atividades:

1. Estruturação do conhecimento captado das fontes disponíveis e montagem da BC no sistema através das rotinas auxiliares;

2. Depuração e testes da BC visando garantir a sua qualidade quanto aos aspectos abordados neste trabalho; e

3. Adequação do gerenciador às características da aplicação, definindo o tipo de motor de inferência e de raciocínio a ser utilizado e estruturando o diálogo com o usuário.

No modo usuário são realizadas as consultas às BCs existentes pelos respectivos usuários. As relações entre o engenheiro do conhecimento, o usuário e as entidades componentes do BACO são ilustradas pela figura 2.

IV.3 - Uma Base de Conhecimento no BACO

Uma BC no BACO é um sistema na forma BC={C,R}, onde C é o conjunto de todas as triplas OAV do universo da aplicação e R é o conjunto das regras do sistema. As regras na BC são apresentadas na forma

$$R(a_1, a_2, ..., a_n) = b$$

onde $a_1, a_2, ..., a_n$ são os antecedentes e b o conseqüente da regra. As regras assim construídas permitem somente o conectivo "e" nos antecedentes e a especificação de um único conseqüente. Apesar dessas limitações tornarem o modelo restrito em têrmos de poder de síntese, há um ganho compensador em generalidade para tratamento de regras, no tocante aos critérios de validação aqui definidos, e flexibilidade para atualização das mesmas.

IV.4 - O Grafo do Conhecimento

Para organização interna do conhecimento o BACO utiliza a estrutura de grafo *AND/OR*, aqui denominado *grafo do conhecimento* (GC). Este grafo consiste num conjunto de vértices, cada um representando uma condição (*OAV*) do arquivo de **REGRAS**. Estas condições se interrelacionam formando as regras da BC.

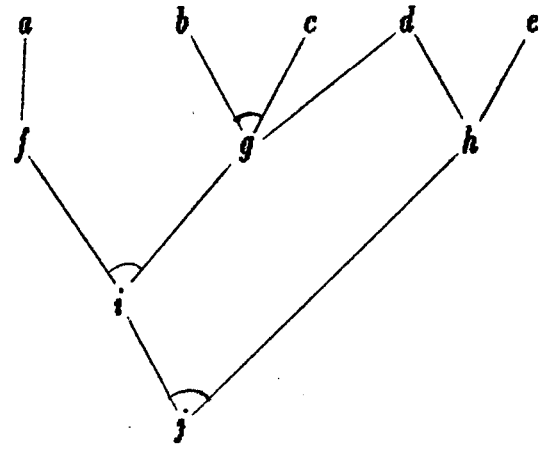

Figura 3: Exemplo de GC

São definidos três tipos de condições:

1. Iniciais - só aparecem como antecedentes;
2. Finais - só aparecem como conseqüentes; e
3. Intermediárias - aparecem como antecedentes em umas regras e como conseqüentes em outras.

Cada aresta (k-conector) do GC representa uma regra onde k é o número de antecedentes da mesma. Se o antecedente é único temos uma aresta OR, caso contrário, uma aresta AND. O GC* correspondente às seguintes regras é mostrado na figura 3:

1. SE *a* ENTÃO *f*

2. SE *b* E *c* ENTÃO *g*

3. SE *d* ENTÃO *g*

* Apesar dos GCs serem grafos direcionados, nos exemplos apresentados neste trabalho omitimos o sentido da maioria dos arcos. Nestes casos, os vértices de origem de cada arco encontram-se, fisicamente, acima dos vértices de destino e, portanto, o sentido deve ser assumido como de cima para baixo. Nas exceções o sentido é explicitado.

4. **SE** *d* **ENTÃO** *h*

5. **SE** *e* **ENTÃO** *h*

6. **SE** *f* **E** *g* **ENTÃO** *i*

7. **SE** *h* **E** *i* **ENTÃO** *j*

Sobre o GC é definido o conceito de nível da seguinte forma:

Definição 1 *Nível de k - Para cada vértice k num GC definimos:*

$$nível\ (k) = \begin{cases} 0, & \textit{se k é inicial} \\ \max\ (nível(antecedente(k))) + 1, & c.c. \end{cases}$$

Como veremos na seção VI.1, este conceito é útil na detecção de encadeamento circular.

IV.5 - O Processo de Inferência

Dentre as facilidades de inferência previstas no BACO foi implementado inicialmente o motor de inferência para raciocínio preciso com encadeamento regressivo. Está prevista também a implementação do encadeamento progressivo e do misto.

Na forma de raciocínio implementada, os FCs (aqui denominados *status de veracidade*) assumem sómente valores extremos V (verdadeiro) ou F (falso). Para se obter os FCs das condições finais são definidos três conceitos:

1. *Grupo de Antecedentes de x* - é o conjunto de condições antecedentes de uma regra que tem x como conseqüente;

2. *Status de Veracidade de uma Condição* - é o valor V ou F definido para condições não iniciais em função de seus antecedentes. Uma condição não inicial x tem seu status S(x)=V quando todos os elementos de pelo menos um grupo de antecedentes de x tiver status V; tem status S(x)=F se todos

os grupos de antecedentes tiverem pelo menos um elemento com status F. O status de veracidade das condições iniciais é determinado pelos dados fornecidos pelo usuário.

3. *Condições de Partida* - São os argumentos do processo de inferência, ou seja, as condições iniciais, no caso de encadeamento progressivo, ou não iniciais, no caso de encadeamento regressivo. No encadeamento misto podem ser iniciais e/ou não iniciais.

Durante o processo de inferência adotado no BACO, o sistema deve ter condições de responder a três questões:

1. Como escolher o próximo objetivo dentre as condições de partida?
2. Uma vez estabelecido o status de uma condição não inicial, qual conseqüente passa a ser o novo objetivo?
3. Quando, ao se determinar o status de uma condição x, constata-se que ela possui um grupo de antecedentes com status V ou desconhecido, qual antecedente passa a ser objetivo?

Para responder a estas questões o sistema pode se utilizar de conhecimento sobre o grafo para estabelecer estratégias de escolha com o objetivo de minimizar o número de inferências. Algumas destas estratégias, baseadas no conceito de nível, são:

1. Dar preferência a condições iniciais com conseqüentes em níveis mais baixos.
2. Dar preferência a conseqüentes mais próximos de uma condição final.
3. Dar preferência a antecedentes com nível mais próximo a zero.

IV.6 - O Encadeamento Progressivo

O encadeamento progressivo pode ser melhor compreendido através do algoritmo EP que apresentamos nesta seção. Considere este algoritmo apenas como uma ilustração do processo, pois não se tem definida como será feita a implementação deste

tipo de encadeamento no BACO. O processamento utiliza uma lista de "condições de partida" que neste caso são iniciais.

Algoritmo EP

Dados: GC e o conjunto de condições de partida.

1. Enquanto houver condições no conjunto de condições de partida retire uma, chame-a x e faça:
   (a) Enquanto x tiver conseqüente não marcado tome um deles, chame-o y, marque-o, marque o caminho corrente (definido pela aresta que liga x e y) e faça:
      i. Se y tem status definido desmarque o caminho corrente. Caso contrário faça:
         A. Se existe um grupo de antecedentes de y em que todas as condições têm status V, então S(y)=V. Caso contrário faça:
            I. Se em todos os grupos de antecedentes de y tem pelo menos uma condição com status F então S(y)=F.
         B. Se y é final ou não tem status desmarque o caminho corrente. Caso contrário faça $x = y$.
         C. Enquanto x é não inicial e tem todos os conseqüentes marcados desmarque-os, faça x=antecedente de x no caminho corrente e desmarque o caminho corrente.
   (b) Desmarque os conseqüentes de x.

A partir do conjunto de condições de partida o EP procura obter o status de veracidade do maior número de condições existentes no GC. Para isso ele toma cada condição inicial e percorre o GC em profundidade, atribuindo status às condições até chegar a uma que seja final ou que tenha um conseqüente para o qual não conseguiu obter o status (passo 1.a.i.B). No momento seguinte (1.a.i.C) é verificada a existência de conseqüentes da condição ainda não avaliados. Caso não exista algum o processo é dirigido para o ancestral existente no caminho já percorrido que ainda tenha conseqüentes não avaliados, reiniciando-se em 1.a ou

1, caso se necessite de uma nova condição inicial. Se a condição tiver algum conseqüente ainda não avaliado o processo é reiniciado em 1.a.

IV.7 - O Encadeamento Regressivo

Neste caso a lista de condições de partida é composta de condições finais (conclusões). A cada momento o sistema possui uma condição objetivo para a qual procura estabelecer status V ou F. Para se obter o status de uma condição x o sistema procede da seguinte forma:

1. Se x é inicial e seu status não figura no arquivo de **FATOS** solicita informação ao usuário;
2. Se x é não inicial e seu status não figura no arquivo de **FATOS**, examina o grupo de antecedentes de x. Pode ocorrer que:
   - exista um grupo de antecedentes de x em que todas as condições têm status V, neste caso S(x)=V;
   - em todos os grupos de antecedentes de x exista pelo menos uma condição com status F, neste caso S(x)=F;
   - exista um grupo de antecedentes de x em que as condições têm status V ou desconhecido, neste caso muda a condição objetivo para um dos antecedentes de x com status desconhecido.

IV.8 - O Encadeamento Misto

Este tipo de encadeamento funciona de forma análoga ao encadeamento regressivo com uma diferença: ao se estabelecer o status de uma condição, caso ela não seja condição final, é acionado o encadeamento progressivo a partir dela, antes de se dirigir o processo para outro objetivo.

IV.9 - Incompatibilidade de Condições

O BACO utiliza o conceito de *incompatibilidade de condições* que pode ser considerado uma extensão da negação de um têrmo na lógica proposicional e é definido da seguinte forma:

Definição 2 *Incompatibilidade - Uma condição y é incompatível com uma condição x se x não puder ser verdadeira quando y o for. Observe que esta definição não exige reflexividade. Não é exigido que x seja incompatível com y quando y for incompatível com x.*

O exemplo da página 16 ilustra essa definição: neste caso, *chamar-o-borracheiro* é incompatível com *trocar-o-pneu*.

A partir dessa definição é caracterizado um conjunto de condições que mantém uma relação de incompatibilidade com uma determinada condição. Esse conjunto é definido da seguinte forma:

Definição 3 *Classe de Incompatibilidade da Condição x CI(x) - A classe de incompatibilidade de uma condição x é constituída por todas as condições incompatíveis com x. O fato de y pertencer à classe de incompatibilidade de x será notado por* $y \in \mathrm{CI}(x)$.

Cabe ao engenheiro do conhecimento definir, para cada condição, a sua classe de incompatibilidade. No exemplo da definição 2 *chamar-o-borracheiro* $\in$ CI(*trocar-o-pneu*).

IV.10 - Consistência e Completeza no BACO

Ao abordar o tema da verificação de BCs os conceitos de consistência e completeza surgem imediatamente, trazendo uma discussão existente na literatura sobre a sua existência num sistema lógico. Apresentamos, informalmente, um resumo dessa

discussão a partir de elementos tirados de HOFSTADTER (1979) e ENDERTON (1972).

Segundo a definição da lógica matemática um sistema lógico na forma [L,A,R], caracterizado por uma linguagem L, um conjunto A de axiomas e um conjunto R de regras de inferência, é dito completo, se toda proposição φ que seja implicação lógica* de qualquer conjunto Γ de fórmulas deste sistema é teorema** do mesmo; mais formalmente, SE $\Gamma \models \varphi$ ENTÃO $\Gamma \vdash \varphi$.

O conceito de consistência é definido pelo fato de que toda proposição φ que seja teorema de Γ é implicação lógica de Γ, em notação lógica, SE $\Gamma \vdash \varphi$ ENTÃO $\Gamma \models \varphi$.

Note que os conceitos de consistência e completeza dizem respeito a sistemas lógicos, conforme caracterizado acima. O foco do nosso trabalho são as teorias*** (BCs) construídas sob o BACO, para as quais queremos definir critérios de validação. Desta forma, a questão da consistência e completeza no BACO foge ao escopo do nosso estudo, sendo que para o mesmo interessa sómente que o BACO é uma extensão da lógica proposicional, definido pela estrutura [L,R,A,I], onde L é uma linguagem para representação simbólica, R um conjunto de regras de inferência, A um conjunto de axiomas e I um conjunto de classes de incompatibilidade. A introdução do conjunto I deve-se ao uso do conceito de incompatibilidade de condições definido para as teorias construídas sob o BACO para as quais os conflitos são definidos de forma peculiar (ver seção IV.8).

Nos próximos capítulos, para que não ocorram ambigüidades, evitamos o uso dos termos consistência e completeza aqui referenciados.

---

* Dizemos que um conjunto de fórmulas Γ implica logicamente φ se toda atribuição de valores verdade às sentenças de Γ e a φ que satisfaça todos os elementos de Γ também satisfaz φ.

** Os teoremas de Γ são sentenças que podem ser obtidas pela aplicação de regras de inferência (um número finito de vezes) a partir dos axiomas de Γ.

*** Uma teoria T é definida como sendo um conjunto de sentenças tal que, para qualquer sentença φ da linguagem se $T \models \varphi$ então $\varphi \in T$

Capítulo V

O Conceito de Base Reduzida

Devido à nossa abordagem para validação de BCs considerar cadeias de regras e não simplesmente regras de forma isolada, torna-se necessário identificar estas cadeias, partindo das suas condições iniciais até os objetivos, assim consideradas as condições não iniciais (finais e intermediárias). Além disto, conforme será visto na seção VI.5, é preciso que tenhamos o controle das relações de incompatibilidade entre estas cadeias de regras. Para atender a estes requisitos definimos neste capítulo: Árvore de Decorrência, Árvore Reduzida, Bases Equivalentes, Base Reduzida e Classes Extendidas de Incompatibilidade (CEI).

V.1 - Definições

Definição 4 *Árvore de Decorrência de k -Para cada vértice k num GC, definimos:*

$$\textit{árvore de decorrência de } k = \begin{cases} k, & \textit{se } k \textit{ é inicial} \\ k \cup \textit{cada combinação das} & \\ \textit{árvores de decorrência} & \\ \textit{dos antecedentes de } k & \textit{c.c.} \end{cases}$$

Na figura 4 mostramos um GC com as suas respectivas árvores de decorrência.

Definição 5 *Árvore Reduzida de k - É a árvore que se obtém, a partir de uma árvore de decorrência de k, através do fechamento transitivo e da eliminação dos vértices intermediários. Considere a árvore de decorrência da figura 5; se a e b forem verdadeiros, e também será. A árvore reduzida da figura 6 representa esse fato.*

Na figura 7 mostramos as árvores reduzidas correspondentes ao exemplo da definição 4.

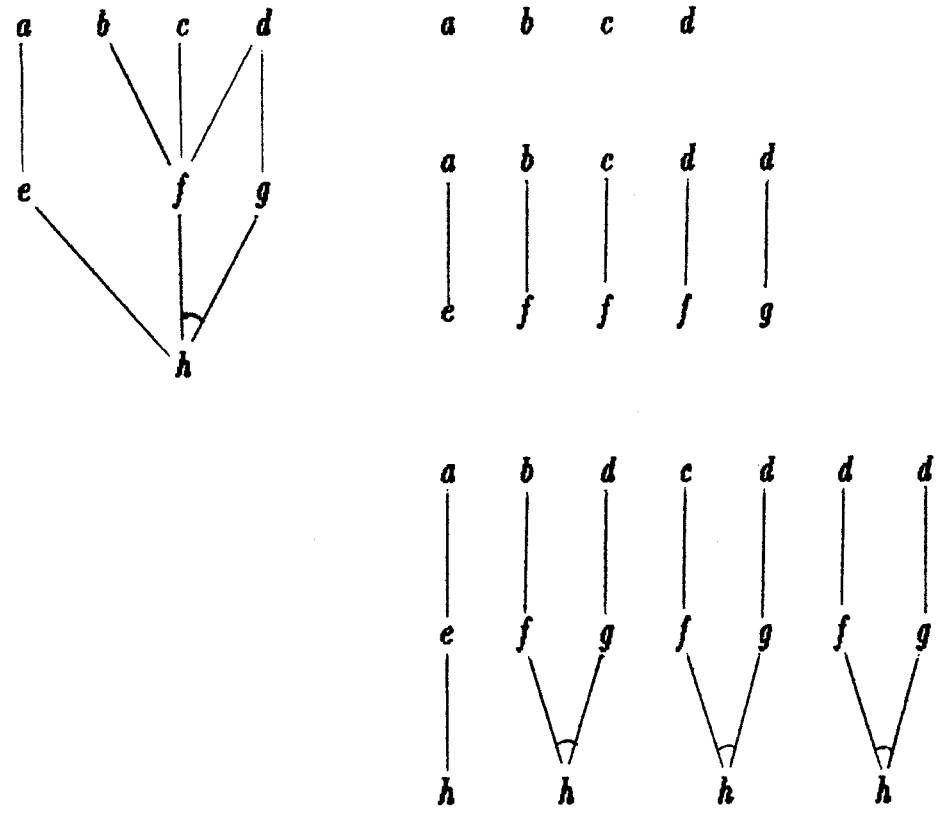

Figura 4: Exemplo de GC com as Respectivas Árvores de Decorrência

Definição 6 *Aplicação sobre uma BC - Em uma BC Z chamamos "uma aplicação sobre Z^v à função $f(a_1, \ldots, a_n) = \{b_1, \ldots, b_m\}$ que, a um conjunto qualquer $A = \{a_1, \ldots, a_n\}$ de condições iniciais, associa o conjunto $B = \{b_1, \ldots, b_m\}$ que contém todas as condições não iniciais que podem ser inferidas quando todos os elementos de A tiverem status V, com $A \in Z$ e $B \in Z$.*

Definição 7 *Bases de Conhecimento Equivalentes - Duas BCs C_1 e C_2 são equivalentes se:*

1. *Possuem conjuntos isomorfos de vértices iniciais;*

2. *Possuem conjuntos isomorfos de vértices não iniciais; e*

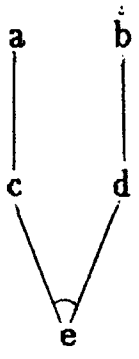

Figura 5: Exemplo de Árvore de Decorrência

Figura 6: Exemplo de Árvore Reduzida

3. *Para todo $A_1 \subseteq$ conjunto de vértices iniciais de C_1 e $A_2 \subseteq$ conjunto de vértices iniciais de C_2 $f(A_1)$ é isomorfo a $f(A_2)$.*

Um exemplo de BCs equivalentes pode ser obtido a partir de uma pequena modificação do exemplo da figura 4, conforme mostra a figura 8.

Definição 8 *Base de Conhecimento Reduzida - Uma BC é reduzida quando o grafo que a representa não possui vértices intermediários. Sua representação é um grafo bipartite.*

Nosso objetivo, ao definir base reduzida, é obter, a partir de uma dada BC, uma base reduzida equivalente a essa. Os processos de validação que descreveremos ficam mais simples se trabalharmos com bases reduzidas; no restante

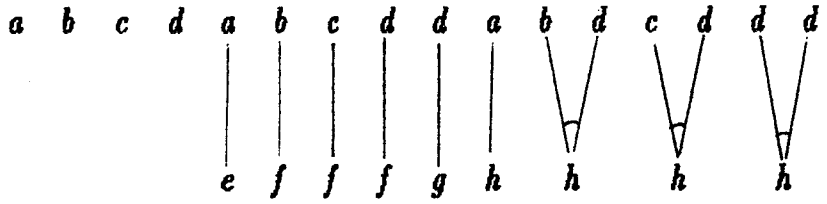

Figura 7: Exemplos de Árvores Reduzidas

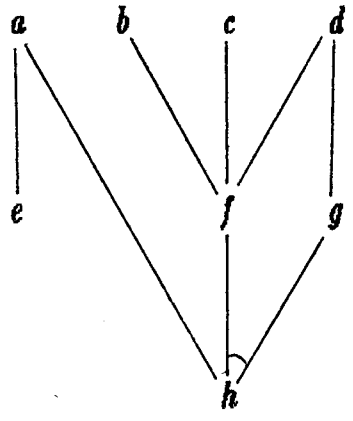

Figura 8: GC de uma BC Equivalente à do Exemplo da Definição 4

do nosso trabalho designaremos base equivalente reduzida por BER. Na figura 9 mostramos um GC de uma BC reduzida.

Definição 9 *Classes Extendidas de Incompatibilidade (CEI) - Para o conjunto* $\{a_1, \ldots, a_n\}$ *de condições iniciais de cada árvore de decorrência, chamamos classe extendida de incompatibilidade ao conjunto*

$$CEI(a_1, a_2, \ldots, a_n) = CI(a_1) \cup CI(a_2) \cup \cdots \cup CI(a_n) \cup CI(b_1) \cup CI(b_2) \cup \cdots \cup CI(b_m)$$

onde $\{b_1, b_2, \cdots, b_m\}$ *é o conjunto dos vértices não iniciais das árvores de decorrência com condições iniciais* $\subseteq \{a_1, a_2, \cdots, a_n\}$. *Este conjunto especifica as condições que não podem ser inferidas quando as condições* $\{a_1, a_2, \ldots, a_n\}$ *e to-*

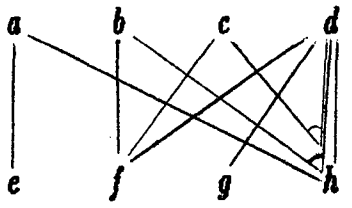

Figura 9: Exemplo de uma BC Reduzida

das as condições que podem ser inferidas a partir deste conjunto tiverem status V.

Para ilustrar essa definição, considere a árvore de decorrência da figura 10. Admitindo CI(f)=$\{a, b\}$ e CI(h)=$\{j, l\}$, temos CEI(c, d)=$\{a, b, j, l\}$.

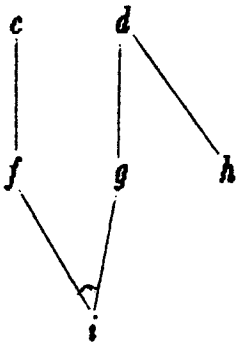

Figura 10: Árvore de Decorrência para Ilustrar Cálculo de CEI

Neste trabalho consideramos que o disparo de uma regra inclui, além da inferência do seu conseqüente, a atribuição de status F às condições existentes nas suas respectivas CIs, no caso de uma BC original, ou na sua CEI, no caso de uma BER.

V.2 - Construção da Base Equivalente Reduzida (BER)

Para geração da BER a partir de um GC dado a que chamamos G, é preciso que, no nosso processo, G esteja representado na forma de listas, conforme o esquema:

$$(v_n, (P_1), (P_2), \ldots, (P_i), \ldots, (P_m))$$

onde v é o vértice não inicial considerado, n é o seu nível, P_i, $i = 1, \ldots, m$ é o conjunto de vértices de origem da i-ésima aresta incidente em v_n e m é a quantidade destas arestas. O conjunto de listas do GC da figura 4 (página 33) é o seguinte:

1. (a_0)
2. (b_0)
3. (c_0)
4. (d_0)
5. $(e_1, (a))$
6. $(f_1, (b), (c), (d))$
7. $(g_1, (d))$
8. $(h_2, (e), (f, g))$

Além disto é necessária uma estrutura a que chamamos *lista de caminho* e é representada da seguinte forma:

$$(META, (C_1, C_2, \ldots, C_p))$$

Esta lista expressa um caminho pelo qual se pode inferir determinado vértice não inicial *(META)* a partir de alguma combinação das condições iniciais suficientes para se inferir cada um dos seus antecedentes em cada aresta. Nesta estrutura cada C_j, $j = 1, \cdots, p$, é um conjunto de condições iniciais suficientes para se alcançar o j-ésimo antecedente de *META* na aresta considerada e p é a quantidade de antecedentes na mesma. Note que, para o conjunto P_i de vértices de origem em cada aresta do GC, existem tantas listas de caminho quantas forem as formas

de se combinar os caminhos até cada um dos seus elementos. Na lista de caminho a posição das condições iniciais de cada antecedente é fixa, ou seja, o primeiro antecedente tem suas condições iniciais em C_1, o segundo em C_2, e assim por diante. Há um caso particular de lista de caminho que só contém *META* quando esta tem *nível* = 0 (é condição inicial). Note que cada uma das listas representa uma árvore de decorrência reduzida de META. O conjunto de listas de caminho do GC da figura 4 é o seguinte:

1. (a)
2. (b)
3. (c)
4. (d)
5. $(e,(a))$
6. $(f,(b))$
7. $(f,(c))$
8. $(f,(d))$
9. $(g,(d))$
10. $(h,(a))$
11. $(h,(b,d))$
12. $(h,(c,d))$
13. $(h,(d))$

O algoritmo para obtenção da BER é escrito da seguinte forma:

Algoritmo BER

Dados: G representado na forma de listas e com níveis atribuídos conforme seção IV.4.

1. Crie listas de caminho na BER para todos os vértices com nível = 0.
2. Para todos os vértices em cada nível de G, a partir do 1 até o último, faça:
   (a) Tome um vértice no nível n, chame-o v_n e coloque-o em *META*.
   (b) Se $n = 1$, para cada caminho definido por v_n e as condições iniciais de cada aresta incidente em v_n, coloque as condições iniciais na lista de caminho e grave-a na BER. Crie a CEI correspondente a esta árvore reduzida pela união das CIs de v_n e das condições iniciais do caminho considerado e chame-a CEICORR. Grave CEICORR. Inclua os elementos de CEICORR em cada CEI já gravada cujo conjunto de argumentos $\supset$ conjunto de argumentos de CEICORR, eliminando as duplicidades.
   (c) Caso contrário, para cada aresta incidente em v_n, faça
      i. Crie uma CEI para as árvores reduzidas que serão geradas, contendo inicialmente CI(v_n). Chame-a CEICORR.
      ii. $j = 1$.
      iii. Até que $j = 0$ faça
         A. Tome o j-ésimo antecedente de v_n e chame-o antecedente corrente.
         B. Se nível do antecedente corrente $\neq 0$, considere suas árvores reduzidas na BER, as respectivas CEIs e faça:
            I. Se ele tem algum arco incidente não marcado, mova os vértices de origem do primeiro arco não marcado para a posição correspondente ao antecedente corrente na lista de caminho de v_n, marque o arco e faça $j = j + 1$. Caso haja alguma CEI para o antecedente corrente em CEICORR, elimine-a. Coloque em CEICORR a CEI referente à aresta considerada do antecedente corrente.
            II. Caso contrário, desmarque os arcos incidentes ao j-ésimo antecedente de v_n e faça $j = j - 1$.
         C. Caso contrário, se o antecedente corrente está marcado, desmarque-o e faça $j = j - 1$. Se ele não está marcado, mova-o

para a sua posição na lista de caminho de v_n, coloque sua CI em CEICORR, marque-o e faça $j = j + 1$.

D. Se $j >$ quantidade de antecedentes de v_n grave a lista de caminho e faça $j =$ quantidade de antecedentes de v_n. Elimine as condições duplicadas em CEICORR e grave-a. Inclua os elementos de CEICORR em cada CEI já gravada cujo conjunto de argumentos $\supset$ conjunto de argumentos de CEICORR, eliminando as duplicidades.

3. Rearranje a BER de forma a que só exista uma lista para cada vértice. Para tanto cada vértice não inicial deverá ter relacionados os seus arcos incidentes e respectivos vértices de origem. Elimine também as listas em que *META* tenha nível = 0.

O BER possui dois passos básicos: o **2** que gera todas as árvores reduzidas de cada vértice não inicial, garantindo que todos os caminhos entre os vértices iniciais e os não iniciais são explicitados, e o **3** que rearranja estas árvores, eliminando a duplicação de vértices e produzindo a BER.

No passo **2** temos duas formas de tratamento de vértices, de acordo com o seu nível. Se o nível é 1 os caminhos possíveis até o vértice são obtidos imediatamente através dos seus antecedentes. Se o nível é maior do que 1 estes caminhos são obtidos pelas combinações dos caminhos que chegam aos seus antecedentes em cada aresta. Como exemplo, considere o GC da figura 3 (página 26) que é mostrado com os níveis atribuídos, de acordo com a definição apresentada na seção IV.4, na figura 11.

A representação do grafo em forma de lista fica:

1. $(f_1, (a))$

2. $(g_1, (b,c), (d))$

3. $(h_1, (d), (e))$

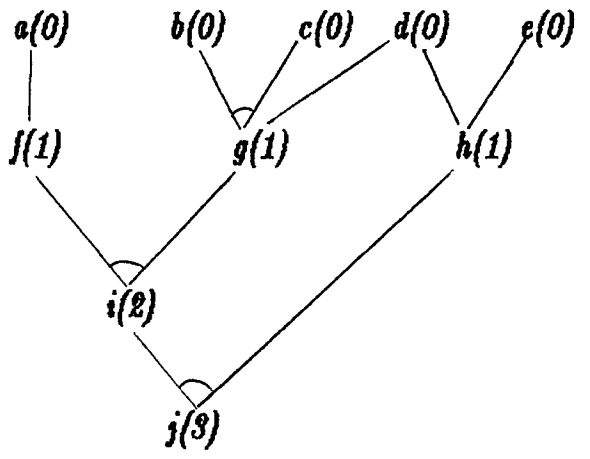

Figura 11: GC da Figura 3 com Níveis Atribuídos

4. $(i_2, (f, g))$

5. $(j_3, (i, h))$

A execução do algoritmo BER sobre este grafo produz, no passo 2 as árvores reduzidas da figura 12.

No terceiro passo as árvores acima são rearranjadas, formando a BER da figura 13.

A representação da BER em forma de lista, depois de executado o passo 3, fica:

1. $(f, (a))$

2. $(g, (b,c), (d))$

3. $(h, (d), (e))$

4. $(i, (a, b,c), (a, d))$

5. $(j, (a, b,c, d), (a, b,c, e), (a, d), (a, d, e))$

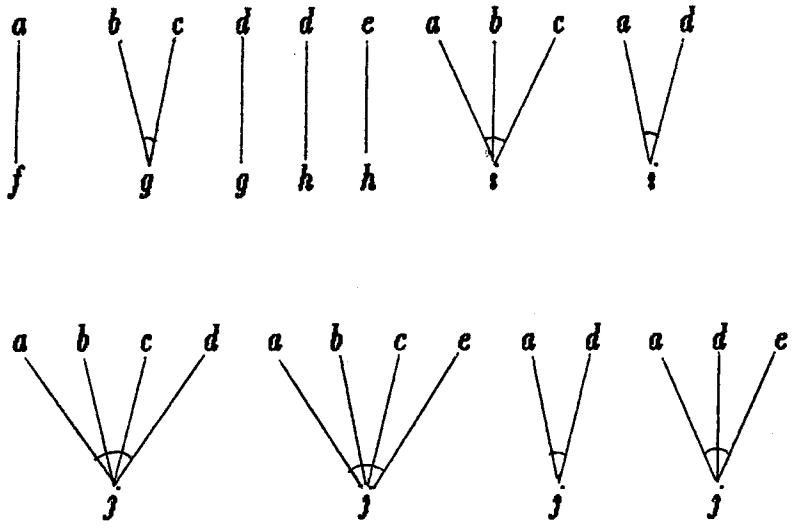

Figura 12: Árvores Reduzidas Geradas pelo BER

V.3 - Complexidade do Algoritmo BER

Nesta análise de complexidade procuramos determinar um limite superior para a complexidade de pior caso; vamos utilizar para isso um grafo de conhecimento com entrada máxima. Considere: z = total de vértices com nível = 0, e = total de vértices com nível > 0, a = número máximo de arestas por vértice, v = número máximo de vértices por aresta e n = nível máximo do grafo.

Nos passos **1** e **2.a** são imediatas as complexidades $O(z)$ e $O(e)$, respectivamente. No **2.b**, supondo que só existam vértices com nível 0 ou 1, temos $O(e \times a)$.

Para cálculo da complexidade em **2.c** utilizamos o *grafo máximo*, definido aqui como aquele que possui n níveis, a arestas por vértice e v vértices por aresta. A ordem de grandeza, $O(T_n)$, da complexidade neste passo é expressa pelo número de árvores geradas para um vértice no nível n do grafo máximo. Na figura 14 exemplificamos um grafo máximo com $a = 3$, $v = 3$ e $n = 2$. Na ilustração só está representado um dos três arcos incidentes ao vértice no nível 2.

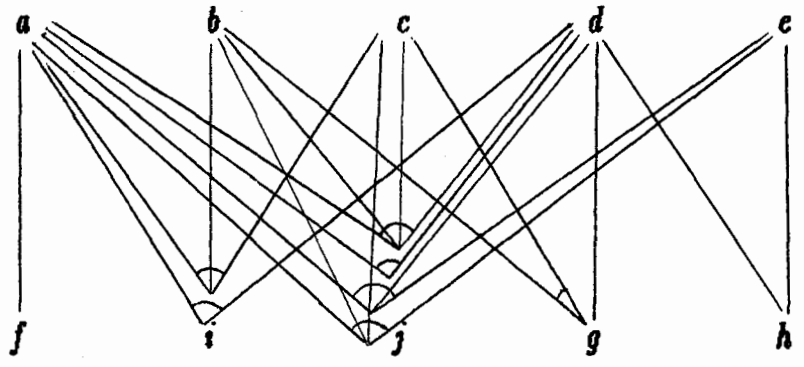

Figura 13: Exemplo de BER

Dedução da fórmula geral de T_n:

$T_1 = a$

$T_2 = \underbrace{T_1 \times T_1 \times \cdots \times T_1}_{v \text{ vezes}} \times a = a^v \times a = a^{v+1}$

$T_3 = \underbrace{T_2 \times T_2 \times \cdots \times T_2}_{v \text{ vezes}} \times a = (a^{v+1})^v \times a = a^{(v^2+v+1)}$

$\vdots$

$T_n = a^{(v^{n-1}+v^{n-2}+\cdots+1)} = a^{(\frac{1-v^n}{1-v})}$

Prova por indução:

Para $n = 1$ verificamos que $T_1 = a^{(\frac{1-v^1}{1-v})} = a$

Provamos que para $n+1$ a fórmula é válida, ou seja,

$T_{n+1} = a^{(\frac{1-v^{n+1}}{1-v})}$

$T_{n+1} = a \times (a^{\frac{1-v^n}{1-v}})^v = a \times a^{(v \times \frac{1-v^n}{1-v})} = a^{(v \times \frac{1-v^n}{1-v}+1)} = a^{(\frac{v-v^{n+1}+1-v}{1-v})} =$

$= a^{(\frac{1-v^{n+1}}{1-v})}$ ■

Obviamente um algoritmo com tal complexidade não é útil sob as

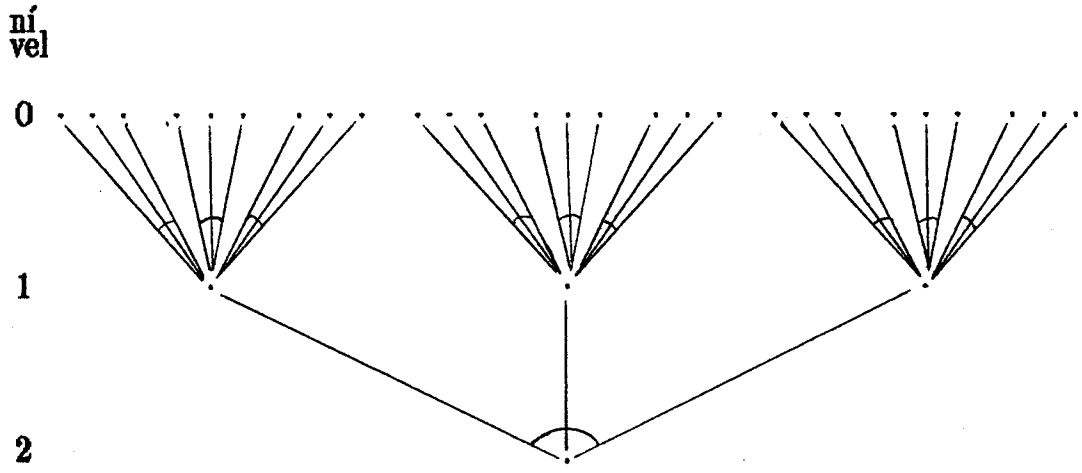

Figura 14: Exemplo de Grafo Máximo

condições de cálculo da mesma. Entretanto, isto não inviabiliza a sua aplicação em diversos casos reais, cujas condições são bem diferentes daquelas.

No passo **3** é feita a comparação das listas referentes às árvores reduzidas geradas em **2.c** para juntar em uma única lista os vértices que tiverem mais de uma árvore. Fazendo t = número total de árvores reduzidas temos a complexidade $O(t^2)$. Vale aqui o mesmo comentário feito para a complexidade no passo **2.c**.

Capítulo VI

Validação das Bases Construídas sob o BACO

Aqui descrevemos os procedimentos propostos para verificação de encadeamentos circulares, condições desnecessárias, regras incompatíveis e regras inativáveis nas BCs do BACO. O conceito de encadeamento circular é definido sobre a BC original e os demais sobre a BER. Neste capítulo poderá ser observado que os procedimentos aqui apresentados constituem um modelo mais poderoso para verificação do que os métodos levantados na bibliografia, principalmente pelo fato de detectar mais problemas do que aqueles. Nas análises de complexidade dos algoritmos apresentados devem ser considerados dois pontos: (a) com exceção do algoritmo NÍVEL, as complexidades dizem respeito ao tempo de processamento, pois é feita, simplesmente, a verificação de estruturas já existentes; (b) quanto às elevadas complexidades obtidas, vale a mesma observação feita para o algoritmo BER na seção V.3.

VI.1 - Obtenção de Bases sem Encadeamento Circular

O material para esta seção encontra-se em PINHO *et alia* (1988) onde é discutido o encadeamento circular e o tratamento dado às BCs do BACO para sua detecção. Para se definir o que é encadeamento circular no BACO é necessário o conceito de ancestral, definido sobre um GC.

Definição 10 *Ancestral de um vértice b - Dada uma regra* $R(a_1, a_2, \ldots, a_n) = b$ *dizemos que o vértice k é ancestral de b se k é igual a algum* a_i *ou é ancestral de algum* $a_i, i = 1, 2, \ldots, n$.

Definição 11 *Encadeamento circular - Dizemos que uma BC possui encadeamento circular se algum vértice do seu GC é ancestral de si mesmo.*

Na detecção de encadeamento circular utiliza-se o conceito de nível definido na seção IV.4. O teorema 1 apresenta as condições para que a base não tenha encadeamento circular:

Teorema 1 *Para que uma BC não tenha encadeamento circular é necessário e suficiente que todos os vértices do seu GC possam ter seu nível calculado.*

As provas, informais, deste e dos próximos teoremas serão feitas em dois passos: no primeiro mostramos que as condições do teorema implicam na característica esperada para a BC e no segundo que a existência desta característica implica nas condições do teorema. Desta forma é verificada a equivalência entre as condições e a característica.

Prova:

P: Todos os vértices do GC podem ter seu nível calculado.

Q: A BC não tem encadeamento circular (nenhuma condição é ancestral de si mesma).

P *implica* Q por indução:

- Se todos os vértices têm nível = 0 então nenhuma condição é ancestral de si mesma:
  Pela própria definição de nível, quando um vértice tem nível = 0 ele é uma condição inicial que, portanto, não tem ancestral.
- Admita que, para todos os vértices com nível até n, nenhuma condição da BC é ancestral de si mesma; então, nenhum vértice do nível $(n+1)$ é ancestral de si próprio. Considere um vértice k no nível $(n+1)$. Como o nível de um vértice é sempre uma unidade maior que o do seu antecedente de maior nível e os os níveis dos ancestrais deste vértice são calculados a partir dos níveis dos seus antecedentes, então todos os ancestrais de k têm nível $\leq n$. Assim, k não pode ser ancestral de um vértice de mesmo nível e, portanto, não pode ser ancestral de si mesmo.

Q *implica* P por refutação:

1. Negue P:

   ¬P: Algum vértice não pode ter seu nível calculado.

2. Chame esse vértice de x e admita que o conjunto de vértices de um GC é finito.

3. Se x não pode ter seu nível calculado é porque algum antecedente seu também não pode. Este não pode porque também tem algum antecedente que não pode. E assim por diante. Como o conjunto de vértices é finito e sempre há algum antecedente que não pode ter seu nível calculado, algum vértice já examinado torna a ser antecedente sem nível calculado, e portanto ancestral de si mesmo.

O algoritmo para detectar encadeamento circular consiste na obtenção do nível de cada vértice no GC e é escrito da seguinte forma:

Algoritmo NÍVEL

Dados: BC na forma $C = (k_1, k_2, \ldots, k_n)$ e R.

1. Tome um vértice k_i que ainda não tenha seu nível determinado.
2. Se k_i é inicial, faça nível $(k_i) \leftarrow 0$.
3. Se k_i é não inicial e todos os seus pais já tiveram seus níveis determinados, faça nível $k_i \leftarrow max($ nível$(pai(k_i))) + 1$.
4. Repita os passos acima até que ocorra uma das seguintes condições:
   (a) Todos os vértices tiverem seus níveis definidos, neste caso não há encadeamento circular.
   (b) Restou um conjunto de vértices para os quais as condições 2 e 3 falharam e, portanto, não é possível determinar seus níveis. Neste caso os vértices envolvidos estão em pelo menos um encadeamento circular ou dependem de vértices nesta situação.

Exemplificando, considere a aplicação do algoritmo NÍVEL sobre o GC da figura 15, cuja representação em forma de listas é a seguinte:

1. $(f, (a), (b))$
2. $(g, (c), (d))$
3. $(e, (f))$
4. $(h, (f, g))$
5. $(a, (e))$

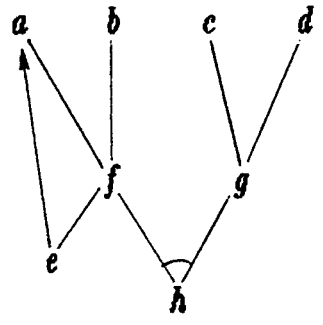

Figura 15: GC com Encadeamento Circular para Atribuição de Níveis

Na atribuição de níveis teremos b, c e d com nível 0, g com nível 1, e a, e, f e h sem condições de se determinar. Na figura 16 é mostrado o GC com seus níveis atribuídos, representado pelas listas:

1. $(f_?, (a_?), (b_0))$
2. $(g_1, (c_0), (d_0))$
3. $(e_?, (f_?))$
4. $(h_?, (f_?, g_1))$
5. $(a_?, (e_?))$

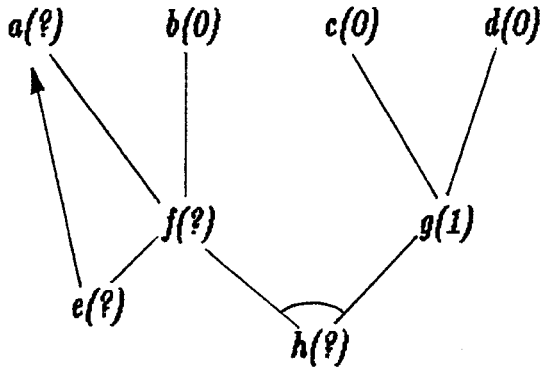

Figura 16: GC após a Atribuição dos Níveis

VI.2 - Complexidade do Algoritmo NÍVEL

O algoritmo é executado sobre o GC na forma de listas, como especificado em V.2.

Supomos, no pior caso, que o próximo vértice para o qual é possível atribuir nível seja o último disponível. Considerando-se n vértices, teríamos n ciclos do algoritmo para o primeiro, $(n-1)$ para o segundo, assim por diante, até o último, com 1 ciclo. Assim, ao todo, teríamos

$$(n + (n-1) + (n-2) + \cdots + 1) = \frac{n^2 + n}{2}$$

o que dá uma complexidade de pior caso $O(n^2)$. Note-se que esta complexidade diz respeito ao tempo de processamento. Como é gerado um dado novo, o nível, existe uma complexidade de espaço que é linear ao número de vértices do GC.

VI.3 - Obtenção de Bases Concisas

A verificação de concisão em uma BER é feita a partir da seguinte definição:

Definição 12 *Condição desnecessária - Dada a regra $R(a_1, \ldots, a_n) = x$ em uma BC A, dizemos que a condição a_k é desnecessária se a sua retirada do conjunto de antecedentes da regra produz uma BC B equivalente a A. Uma BC que não possua condições desnecessárias é dita concisa.*

Intuitivamente, observamos que condições desnecessárias podem se apresentar como simples repetições de condições regras ou como subordinações de regras, onde o conjunto de antecedentes de uma está contido no da outra. O teorema 2 descreve a condição sob a qual uma BER é concisa.

Teorema 2 *Para que uma BER seja concisa é necessário e suficiente que não existam duas regras* $R'(A) = x$ *e* $R''(B) = x$, *tal que* $A \subseteq B$ *ou* $B \subseteq A$, *onde* A *e* B *são conjuntos de vértices iniciais da BER.*

Prova:

P: Não existem duas regras $R(A) = x$ e $R'(B) = x$ com $A \subseteq B$ ou $B \subseteq A$.

Q: A BER não possui condições desnecessárias.

P *implica* Q por refutação:

1. Negue Q:

   $\neg Q$: A BER possui uma condição desnecessária.

2. Suponha a regra $R(A) = x$, com $A = \{a_1, \ldots, a_k, \ldots, a_n\}$ e a_k desnecessária.

   Para que a retirada de a_k de R não impeça a conclusão de x (mantenha a equivalência) é necessário que haja uma outra regra $R'(B) = x$, com $B = A - \{a_k\}$ e, portanto, $B \subset A$. No caso da igualdade $A = B$ é evidente que todas as condições de uma delas são desnecessárias.

Q *implica* P por refutação:

1. Negue P:

   $\neg P$: Existem duas regras $R(A) = x$ e $R'(B) = x$ com $A \subseteq B$ ou $B \subseteq A$.

2. Tome as regras $R(A) = x$ e $R'(B) = x$, com $A = \{a_1, \ldots, a_i, a_{i+1}, \ldots, a_n\}$ e $B = \{a_1, \ldots, a_i\}$.

A retirada de $\{a_{i+1}, \ldots, a_n\}$ de A não impede que x seja concluido pelas condições restantes através de $R'(B) = x$. Desta forma é mantida a equivalência e, portanto, as condições do conjunto $\{a_{i+1}, \ldots, a_n\}$ são desnecessárias.

O algoritmo para detecção de condições desnecessárias é o seguinte:

Algoritmo CONC

Dados: BER na forma de lista.

1. Para cada lista na BER chame-a *LISTACORR* e faça:

   (a) Seja $m+1$ o número de sublistas contidas em *LISTACORR*. Considerando que a primeira contém um vértice a que chamamos v e as demais as condições iniciais de cada cadeia de regras até v, teremos m sublistas de condições iniciais. Faça $i = 2$.

   (b) Enquanto $i < m+1$ faça:

      i. Tome a i-ésima sublista de *LISTACORR* e chame-a *I*. Faça $j = i+1$.

      ii. Enquanto $j \leq m+1$ faça:

         A. Tome a j-ésima sublista de *LISTACORR* e chame-a *J*.

         B. Se comprimento de $I \leq$ comprimento de J chame I de *MENOR* e J de *MAIOR*. Caso contrário, chame I de *MAIOR* e J de *MENOR*. Faça $l = 1$ e $DIF = 0$.

         C. Enquanto $l \leq$ número de condições em *MENOR* faça:

            I. $s = 1$

            II. Enquanto $s \leq$ número de condições em *MAIOR* e MENOR(l) $\neq$ MAIOR(s) faça $s = s+1$.

            III. Se $s >$ número de condições em *MAIOR* faça $DIF = 1$ e $l =$ número de condições em *MENOR* $+ 1$. Caso contrário faça $l = l+1$.

         D. Se $DIF = 0$ relate **condições desnecessárias** nas sublistas de ordem i e j de *LISTACORR*.

         E. Faça $j = j+1$.

iii. Faça $i = i + 1$.

Como ilustração, considere a base representada pelo GC da figura 17, cujas listas são as seguintes* :

1. $(f_1, (a), (b))$
2. $(g_1, (b,c))$
3. $(h_1, (c), (d,e))$
4. $(i_2, (f,g))$
5. $(j_2, (h))$

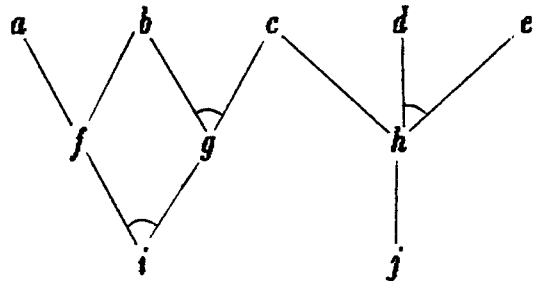

Figura 17: GC de uma BC cuja BER Apresenta Condições Desnecessárias

A execução do algoritmo BER sobre o grafo produz a BER expressa pelas listas:

1. $(f, (a), (b))$
2. $(g, (b,c))$
3. $(i, (b,c), (a,b,c))$

* No texto subseqüente só representaremos o nível dos vértices onde for necessário para compreensão da exposição.

4. $(h, (c), (d, e))$

5. $(j, (c), (d, e))$

A lista $(i, (b,c), (a, b,c))$ submetida ao algoritmo CONC acusará a existência de condições desnecessárias entre as sublistas 2 e 3 da mesma.

VI.4 - Complexidade do Algoritmo CONC

Para análise da complexidade de pior caso do CONC consideramos que as condições de todas as regras para cada condição não inicial sejam desnecessárias e definimos as variáveis d = número de condições não iniciais (listas da BER), m = número máximo de regras por condição não inicial e n = número máximo de condições iniciais por regra.

No passo 1 temos d ciclos referentes às d listas. No b são tomadas duas a duas as sublistas de cada lista, resultando em $\frac{m^2-m}{2}$ ciclos. No passo c temos a comparação de duas listas com n elementos cada, gerando n^2 ciclos. No total temos $d \times \frac{m^2-m}{2} \times n^2$, resultando numa complexidade $O(d \times m^2 \times n^2)$.

VI.5 - Obtenção de Bases sem Regras Incompatíveis

A partir do conceito de incompatibilidade de condições apresentada na seção IV.9, definimos uma BER sem regras incompatíveis da seguinte forma:

Definição 13 *Regras incompatíveis - Duas regras R' e R'' são incompatíveis se, ao ser disparada a regra R'', for atribuído a uma condição x um status diferente daquele já atribuído a x pelo disparo de R'.*

Há um caso particular de regras incompatíveis em que $R' = R''$, com a incompatibilidade localizada em alguma árvore de decorrência da conclusão da regra. A este tipo de regra chamamos inativável. Um tratamento independente dos dois tipos de incompatibilidades, entre regras e em regras, pode ser útil em

lógicas onde a atribuição de status às condições varie segundo a seqüencia dos disparos. Por este motivo descrevemos separadamente este tipo de regra na seção VI.7.

Definimos no teorema 3 as condições para que uma BER não tenha regras incompatíveis.

Teorema 3 *Para que uma BER não tenha regras incompatíveis é necessário e suficiente que para toda regra na forma* $R(a_1, a_2, \ldots, a_n) = x$, *com sua respectiva* $CEI(a_1, a_2, \ldots, a_n) = \{b_1, b_2, \ldots, b_m\}$, *tenhamos: (a) todo elemento* b_j, $j = 1, \ldots, m$, *com nível* $\neq 0$, *possui pelo menos um antecedente de cada regra onde ele é condição final também em* $CEI(a_1, a_2, \ldots, a_n)$; *e (b) toda regra que possui algum* b_j, $j = 1, \ldots, m$, *como condição (inicial ou final), tem pelo menos um* a_i, $i = 1, \ldots, n$, *em sua CEI.*

Prova:

P: Para toda regra na forma $R(a_1, a_2, \ldots, a_n) = x$, com sua CEI$(a_1, a_2, \ldots, a_n) = \{b_1, b_2, \ldots, b_m\}$, temos que: (a) todo elemento b_j, $j = 1, \ldots, m$, com nível $\neq 0$, possui pelo menos um antecedente de cada regra onde ele é condição final também em CEI$(a_1, a_2, \ldots, a_n)$; e (b) toda regra que possui algum b_j, $j = 1, \ldots, m$, como condição (inicial ou final), tem pelo menos um a_i, $i = 1, \ldots, n$, em sua CEI.

Q: Não há regras incompatíveis.

P *implica* Q por refutação:

1. Negue Q:

   $\neg Q$: Há duas regras $R'(A) = x$ e $R''(C) = y$ incompatíveis, com $A = \{a_1, \ldots, a_n\}$ e $C = \{c_1, \ldots, c_m\}$.

2. Seja CEI$(A) = \{\ldots, y, c_i, \ldots\}$ com c_i antecedente de y e CEI$(C) = \{\ldots, a_j, \ldots\}$. Há duas seqüências possíveis de disparo:

i) $R' \longrightarrow R''$

- Dispare $R' \Longrightarrow$ todos os elementos de A e x são V, y e c_i são F.
- Não se pode disparar R''.

ii) $R'' \longrightarrow R'$

- Dispare $R'' \Longrightarrow$ todos os elementos de C e y são V, a_j é F.
- R' não pode ser disparada.

Q *implica* P por refutação:

1. Negue P:

   $\neg P$: Há duas regras na forma $R'(A) = x$ e $R''(C) = y$, onde $A = \{a_1, a_2, \ldots, a_n\}$ e $C = \{c_1, c_2, \ldots, c_m\}$ em que: (a) CEI(A) = $\{\ldots, y, \ldots\}$ e y não possui antecedente em CEI(A); ou (b) CEI(A) = $\{\ldots, c_i, \ldots\}$, sem elemento de A em CEI(C).

2. caso 1 - CEI(A) = $\{\ldots, y, \ldots\}$ e y não possui antecedente em CEI(A):

   - Dispare $R' \Longrightarrow$ todos os elementos de A e x são V e y é F.
   - Dispare $R'' \Longrightarrow$ todos os elementos de C são V e é atribuido V a y que já é F $\Longrightarrow$ incompatibilidade contrariando Q.

3. caso 2 - CEI(A) = $\{\ldots, c_i, \ldots\}$ sem elemento de A em CEI(C). Há duas seqüencias de disparo a serem analisadas:

   i) $R' \longrightarrow R''$

   - Dispare $R' \Longrightarrow$ todos os elementos de A e x são V e c_i é F.
   - Não se pode disparar R''.

   ii) $R'' \longrightarrow R'$

   - Dispare $R'' \Longrightarrow$ todos os elementos de C e y são V.
   - Dispare $R' \Longrightarrow$ todos os elementos de A e x são V e é atribuido F a c_i que já é V $\Longrightarrow$ incompatibilidade contrariando Q.

O algoritmo para detecção de regras incompatíveis é o seguinte:

Algoritmo CONFL

Dados: BER e CEIs.

Para cada *CEI*, chame à sua lista de elementos de *CEIA* e à sua lista de argumentos de *INICA* e faça:

1. Enquanto houver vértice não marcado em *CEIA* tome o próximo, chame-o *VÉRTICE* e faça:

   (a) Se nível de *VÉRTICE* $\neq 0$ tome as listas de caminho de *VÉRTICE* na BER e faça:

      i. Enquanto houver arco não marcado incidente a *VÉRTICE* faça:

         A. Considere o próximo arco não marcado incidente a *VÉRTICE*, tome a sua lista de condições iniciais e chame-a *ANTC*.

         B. Faça $i = 1$ e $j = 1$.

         C. Enquanto $i \leq$ número de condições em *ANTC* e *ANTC(i)* $\neq$ *CEIA(j)* faça:

            I. $j = j + 1$.

            II. Se $j >$ número de condições em *CEIA* faça $j = 1$ e $i = i + 1$.

         D. Se $i >$ número de condições em *ANTC* relate **incompatibilidade** entre a regra que tem INICA como condições iniciais e a que tem *ANTC* como condições iniciais e *VÉRTICE* como condição final.

         E. Marque o arco considerado.

   (b) Caso contrário, para cada lista de caminho na BER, chame-a *LISTA* e faça:

      i. Enquanto houver lista de condições iniciais não marcada em *LISTA*, tome a próxima, chame-a *INICB* e faça:

         A. $i = 1$.

         B. Enquanto $i \leq$ número de condições em *INICB* e *VÉRTICE* $\neq$ *INICB(i)* faça $i = i + 1$.

C. Se $i \leq$ número de condições em *INICB* tome a *CEI* de *INICB*, chame-a *CEIB* e faça:

I. $j = 1$ e $i = 1$

II. Enquanto $i \leq$ número de condições em *INICA* e *INICA(i)* $\neq$ *CEIB(j)* faça:

1. $j = j + 1$.

2. Se $j >$ número de condições em *CEIB* faça $j = 1$ e $i = i + 1$.

III. Se $i \leq$ número de condições em *INICA* e *SEQÜÊNCIA* de *INICA* = *SEQÜÊNCIA* de *INICB* então relate **incompatibilidade consigo mesma**;

IV. Se $i >$ número de condições de *INICA* então relate **incompatibilidade** entre as duas regras que têm como condições iniciais *INICA* e *INICB*.

D. Marque a lista considerada.

(c) Marque o vértice considerado em *CEIA*.

Note que este algoritmo serve também para a detecção de regras incompatíveis consigo mesmas no passo **1.b.i.C.III**.

Como exemplo de regras incompatíveis considere a BER, cujo GC é representado na figura 18 e que na forma de listas se escreve:

1. $(w, (a, b, c, d))$

2. $(x, (f, g, h))$

3. $(y, (e, i, j))$

4. $(z, (j, l, m))$

5. $(v, (m, n, o,))$

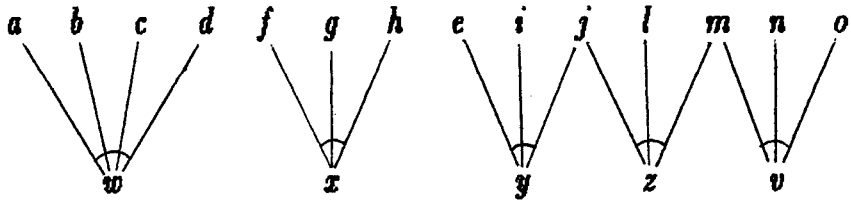

Figura 18: GC de uma BER com Regras Incompatíveis

Suas CEIs são representadas por listas com duas sublistas, onde a primeira contém as condições iniciais da regra e a segunda os vértices incompatíveis com ela, como segue:

1. $((a, b, c, d), (e, f))$
2. $((f, g, h), (n, o)$
3. $((m, n, o), (z))$

A aplicação de CONFL sobre as CEIs e a BER aponta as seguintes regras incompatíveis:

1. $R(a, b, c, d) = w$ e $R(e, i, j) = y$
2. $R(a, b, c, d) = w$ e $R(f, g, h) = x$
3. $R(f, g, h) = x$ e $R(m, n, o) = v$
4. $R(m, n, o) = v$ e $R(j, l, m) = z$

Nos três primeiros casos as regras são incompatíveis por falta de reflexividade na definição das suas CEIs. Considere a primeira: se a, b, c e d tiverem status V podemos concluir w e, pela $CEI(a, b, c, d)$, impedir o disparo da regra $R(e, i, j) = y$; entretanto, se tivermos e, i e j com status V podemos concluir y, mas não será impedido o disparo da regra $R(a, b, c, d) = w$ por falta de uma

CEI apropriada. O quarto caso é decorrente da falta de algum antecedente de z em CEI(m, n, o), o que permite que a regra $R(j, l, m) = z$ seja disparada mesmo quando $R(m, n, o) = v$, que é incompatível com z, também for.

VI.6 - Complexidade do Algoritmo CONFL

Este algoritmo possui dois passos básicos cuja execução depende do nível das condições, ou vértices, da lista de elementos de cada CEI: se nível $\neq 0$ executa-se o passo **a** e se nível $= 0$ o passo **b**. Nomeando-se as variáveis, temos $r =$ número de CEIs ou de regras, $m =$ número máximo de condições da lista de elementos de cada CEI, $n =$ número máximo de condições iniciais em cada regra e $a =$ número máximo de regras por condição não inicial. Para cálculo da complexidade de pior caso, analisamos a execução somente do passo **a** e depois somente do passo **b**.

Numa primeira abordagem, supondo que todas as condições da lista de elementos de cada CEI tenham nível $\neq 0$, no passo **a** teremos a complexidade $O(r \times m \times n \times a)$.

Caso todas as condições da lista de elementos de cada CEI tenham nível $= 0$, teremos $r \times m$ condições em CEIs processadas. Para cada uma destas condições são processadas $r \times n$ condições iniciais em regras na BER. Neste ponto temos em mãos as condições iniciais de uma regra *(INICA)* e as condições em uma CEI *(CEIB)* para serem comparadas, totalizando $n \times m$ comparações. Isto resulta em $r \times m \times r \times n \times n \times m$ comparações, ou seja, uma complexidade $O(r^2 \times m^2 \times n^2)$.

VI.7 - Descrição das Regras Inativáveis

Definição 14 *Regras inativáveis - Uma regra $R(a_1, \ldots, a_n) = x$ é inativável se seus antecedentes nunca possuem status V simultâneamente.*

A idéia subjacente é que, numa BC sem regras inativáveis, toda condição não inicial será concluida em alguma atribuição de status de veracidade às condições iniciais. Como exemplo de regra inativável considere o grafo da figura 19, cujas regras são expressas pelas listas:

Regras da BC:

1. $(e, (a, b))$
2. $(f, (c, d))$
3. $(g, (e, f))$

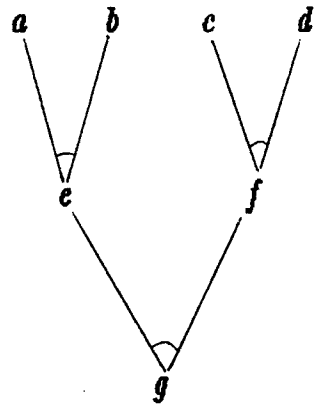

Figura 19: GC de uma BER com Regras Inativáveis

Considere as CIs $(e, (d))$ e $(c, (a))$. Pela execução do algoritmo BER obtemos as seguintes regras e CEIs:

Regras da BER:

1. $(e, (a, b))$
2. $(f, (c, d))$
3. $(g, (a, b, c, d))$

CEIs:

1. $((a, b), (d))$
2. $((c, d), (a))$
3. $((a, b, c, d), (a, d)$

Pela aplicação do algoritmo CONFL, a regra $(g,(a,b,c,d))$ da BER é apontada como incompatível consigo mesma em decorrência da regra $(g,(e,f))$ da BC ser inativável.

VI.8 - Relações entre uma BC e sua BER

Devido à abordagem por nós adotada enfocar ora a BC, no caso de detecção de encadeamento circular, ora a BER, nos demais casos, é importante verificarmos como estas bases se relacionam quanto aos problemas abordados.

Nesta seção mostraremos, informalmente, que:

I. Qualquer BC, com pelo menos uma condição fora de um encadeamento circular, pode ter uma BER.

II. Uma condição desnecessária em uma BC implica na existência de pelo menos uma condição desnecessária na BER. A recíproca não é verdadeira.

III. Um par de regras incompatíveis em uma BC implica na existência de pelo menos duas regras incompatíveis na BER.

IV. Uma regra inativável em uma BC implica na existência de pelo menos uma regra incompatível consigo mesma na BER.

I. Qualquer BC, com pelo menos uma condição fora de um encadeamento circular, tem uma BER.

P: Há uma BC com pelo menos uma condição fora de um encadeamento circular.

Q: Há uma BER.

Prova de P *implica* Q por refutação:

1. Negue Q:

¬Q: Não existe BER.

2. Considere uma BC com uma condição x que não é ancestral de si mesma. Neste caso o vértice correspondente a x no GC tem nível definido.

3. O algoritmo BER possui três passos básicos:

   i) coloca todos os vértices com nível 0 na BER.

   ii) gera árvores reduzidas para os outros vértices.

   iii) rearranja as árvores reduzidas.

4. O algoritmo BER gera vértices na BER a partir de vértices com nível definido na BC, que é o caso de x. Então existe BER com pelo menos x.

II. Uma condição desnecessária em uma BC implica na existência de pelo menos uma condição desnecessária na BER. A recíproca não é verdadeira.

P: Há uma condição desnecessária na BC.

Q: Há pelo menos uma condição desnecessária na BER.

Prova de P *implica* Q por refutação:

1. Negue Q:

¬Q: Não existe condição desnecessária na BER.

2. Suponha uma regra $R(a_1, \ldots, a_k, \ldots, a_n) = x$ na BC com uma condição a_k desnecessária.

3. Considere o conjunto de todas as árvores de decorrência de x geradas a partir de $\{a_1, \ldots, a_k, \ldots, a_n\}$ e m o número de árvores. A partir deste conjunto são criadas na BER as regras originadas por $R(a_1, \ldots, a_k, \ldots, a_n) = x$.

4. Considere $I_j, j = 1, \ldots, m$, o conjunto das condições iniciais de cada árvore de decorrência de x.

5. Considere o conjunto de todas as árvores de decorrência de a_k, p o número de árvores e $L_q, q = 1, \ldots, p$, o conjunto das condições iniciais de cada árvore.

6. Pela definição de árvore de decorrência há sempre uma L_q em I_j.

7. Para que a retirada das árvores de decorrência de a_k não altere $f(I_j)$ (mantenha a equivalência) é porque existe alguma outra regra que conclui x com condições iniciais $\{a_1, \ldots, a_n\}$ (sem a_k).

8. As regras na BER correspondentes aos dois casos são escritas da seguinte forma:

$R(I_j) = x$ e $R'(I_j - L_q) = x$

As condições L_q em I_j são desnecessárias.

9. Por 5 e 6 verifica-se que, a partir desta regra, a quantidade de condições desnecessárias na BER é $\geq$ a quantidade de condições desnecessárias na BC por causa das árvores de decorrência de a_k.

Mostramos que Q *implica* P é falso com um contra-exemplo na figura 20.

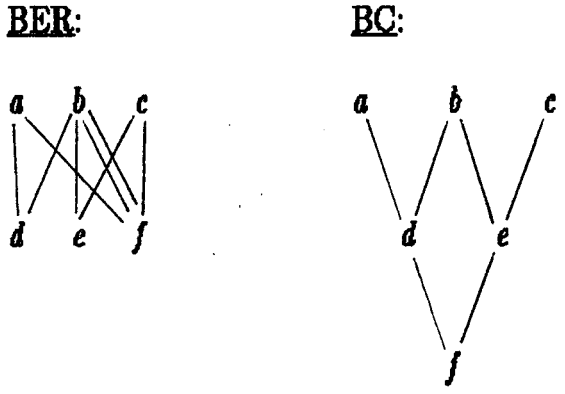

Figura 20: Condição Desnecessária na BER sem Correspondência na BC

Há duas regras $R(b) = f$ na BER, caracterizando b como desnecessária em uma delas. No entanto, a BC não possui condições desnecessárias. Para verificar isso, basta eliminar, uma a uma, as condições de cada regra e observar se a conclusão continua sendo alcançada a partir das mesmas condições iniciais que levaram àquela condição eliminada. Tome como exemplo a regra $R(d) = f$: supondo que d só pudesse ser concluída a partir de b e, se considerarmos a regra $R(e) = f$ onde e também pode ser concluída a partir de b, poderíamos

eliminar d, mantendo a equivalência. Entretanto, observe que d pode ser concluído a partir de a, permitindo que f também seja concluído a partir de a. Como não existe outra forma de se concluir f a partir de a, não podemos considerar d como condição desnecessária. A verificação pode ser feita de maneira análoga para as outras condições.

III. Um par de regras incompatíveis em uma BC implica na existência de pelo menos duas regras incompatíveis na BER.

P: Há duas regras incompatíveis na BC.

Q: Há pelo menos duas regras incompatíveis na BER.

Prova de P *implica* Q por refutação:

1. Negue Q:

¬Q: Não existem regras incompatíveis na BER.

2. Duas regras incompatíveis na BC podem ocorrer por dois motivos:

   i) $R(a_1, \ldots, a_k, \ldots, a_n) = x$ e $R'(b_1, \ldots, b_m) = y$, não necessariamente distintas, com $CI(a_k) = \{y\}$. Ocorre incompatibilidade porque não há algum ancestral de y também na $CI(a_k)$.
   Tome as árvores de decorrência de x, que originam pelo menos uma regra na BER e seja I_x qualquer conjunto de condições iniciais destas árvores. Faça o mesmo para y, considerando I_y como o conjunto de condições iniciais. Assim, temos $\text{CEI}(I_x) = \{y\}$ e $\text{CEI}(I_y) = \emptyset$, com tantas incompatibilidades quantas combinações de árvores de x com árvores de y houverem.

   ii) $R(a_1, \ldots, a_k, \ldots, a_n) = x$ e $R'(b_1, \ldots, b_j, \ldots, b_m) = y$, não necessáriamente distintas, com $\text{CI}(a_k) = \{b_j, y\}$. Ocorre incompatibilidade por falta de reflexividade.
   Repetindo (i), obtemos $\text{CEI}(I_x) = \{b_j, y\}$ e $\text{CEI}(I_y) = \emptyset$, com a mesma quantidade de incompatibilidades do caso (i).

IV. Uma regra inativável em uma BC implica na existência de pelo menos uma regra incompatível consigo mesma na BER.

P: Uma regra inativável em uma BC.

Q: Há pelo menos uma regra incompatível consigo mesma na BER.

Prova de P *implica* Q por refutação:

1. Negue Q:

¬Q: Não existe regra incompatível consigo mesma na BER.

2. Em uma regra $R(a_1, \ldots, a_k, \ldots, a_n) = x$ algum a_k só pode ser sempre falso, quando a regra é disparada sozinha, se algum a_i tem $\mathrm{CI}(a_i) = \{a_k\}$.

3. Considere as árvores de decorrência de a_i e de a_k com I_{ai} qualquer conjunto de condições iniciais das árvores de a_i e I_{ak} o mesmo para a_k.

4. Para que a_k seja sempre F quando a_i for V existe algum elemento de I_{ak} na $\mathrm{CEI}(I_{ai})$ e para que a_i seja sempre F quando a_k for V existe algum elemento de I_{ai} na $\mathrm{CEI}(I_{ak})$.

5. Considere as árvores de decorrência de x e I_x o conjunto de condições iniciais de qualquer delas. Ora, se $I_{ai} \subset I_x$ e $I_{ak} \subset I_x$ então $\mathrm{CEI}(I_x) \supset \mathrm{CEI}(I_{ai})$ e $\mathrm{CEI}(I_x) \supset \mathrm{CEI}(I_{ak})$.
   Portanto, por (4), há dois elementos de I_x em $CEI(I_x)$, gerando tantas regras incompatíveis consigo mesmas quantos I_x houver.

VI.9 - Tratamento dos Problemas Detectados

Encadeamentos circulares:

Os encadeamentos circulares detectados através do NÍVEL necessitam de uma decisão do engenheiro do conhecimento e/ou do(s) especialista(s) para sua solução. Esta tarefa pode ser auxiliada com o uso da facilidade de edição do BACO: a edição dos vértices no GC sem nível determinado, bem como das

relações entre eles, oferece uma visualização do problema, tornando-o mais fácil de ser solucionado.

Condições desnecessárias:

Aqui é importante ressaltar dois tipos de condições desnecessárias existentes na BER: um que é decorrente do "achatamento" da BC, com a eliminação das condições intermediárias, e outro, que ocorre diretamente na BC, pela repetição de antecedentes em regras com conseqüentes iguais. O primeiro caso pode ser ilustrado pela figura 21, cuja BER, expressa através de listas, é a seguinte:

1. $(d, (a), (b))$
2. $(e, (b), (c))$
3. $(f, (a), (b), (b), (c))$

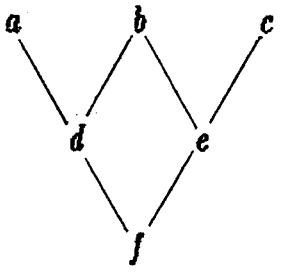

Figura 21: GC com Condições Desnecessárias por "Achatamento"

O segundo caso pode ser ilustrado pelas regras **SE** *f* **E** *g* **ENTÃO** *i* e **SE** *f* **ENTÃO** *i*, cuja solução pode ser a eliminação de uma das duas.

Aqui, como nos demais problemas abordados, a solução das condições denecessárias detectadas na BER pode ser bastante auxiliada pela edição da estrutura do conhecimento envolvida.

Regras Incompatíveis

No caso das regras incompatíveis, em geral, o que for detectado na BER é decorrente de erros na definição das CIs das condições da BC original e, portanto, tem sua solução a partir da revisão destas CIs. Como apoio a esta revisão pode-se editar a porção do GC correspondente às condições envolvidas entre *META* e as condições iniciais de cada par de listas de caminho em conflito.

Nas regras incompatíveis detectadas no passo **1.a** deve-se identificar qual das condições iniciais de uma regra será incluida na CI da outra. No caso do **passo 1.b** deve ser estabelecida a reflexividade entre as CEIs das regras incompatíveis. Caso a definição das CIs esteja correta, então o erro está na definição das regras.

Capítulo VII

Uma Aplicação

Neste capítulo é apresentada uma BC fictícia sobre a qual são aplicados, na seqüência, os algoritmos NÍVEL, BER, CONC e CONFL. Esta BC foi criada, propositalmente, com problemas de encadeamento circular, de condições desnecessárias, de regras incompatíveis e de regras inativáveis com vistas a avaliar a aplicabilidade dos algoritmos propostos. Note que a presença de conteúdo semântico para o nosso objetivo é irrelevante e, portanto, a BC foi definida sem esta característica.

VII.1 - Especificação da Base de Conhecimento

Para especificar a BC é necessário definir um conjunto de OAVs a partir do qual são escritas as regras da base. No nosso exemplo representamos este conjunto pelas letras de a a z, distribuídas segundo a seguinte convenção:

Condições iniciais - $a, b, c, d, e, f, g, h, i$
Condições intermediárias - $j, k, l, m, n, o, p, q, r, t, u, v, w$
Condições finais - s, x, y, z

As regras são relacionadas a seguir e a estrutura do conhecimento correspondente é apresentada no GC da figura 22.

1. SE a ENTÃO j

2. SE b ENTÃO k

3. SE c ENTÃO k

4. SE c ENTÃO l

5. SE *d* E *e* ENTÃO *l*

6. SE *e* E *f* ENTÃO *q*

7. SE *f* ENTÃO *m*

8. SE *g* ENTÃO *m*

9. SE *h* ENTÃO *m*

10. SE *h* ENTÃO *n*

11. SE *i* ENTÃO *n*

12. SE *j* E *k* ENTÃO *o*

13. SE *l* ENTÃO *p*

14. SE *m* ENTÃO *r*

15. SE *n* ENTÃO *r*

16. SE *j* ENTÃO *s*

17. SE *o* ENTÃO *s*

18. SE *k* E *o* ENTÃO *t*

19. SE *p* ENTÃO *t*

20. SE *p* ENTÃO *u*

21. SE *q* ENTÃO *u*

22. SE *q* E *r* ENTÃO *v*

23. SE *r* E *n* ENTÃO *w*

24. SE t **ENTÃO** a

25. SE t **ENTÃO** x

26. SE u **ENTÃO** x

27. SE t **ENTÃO** y

28. SE v **ENTÃO** y

29. SE u **E** w **ENTÃO** z

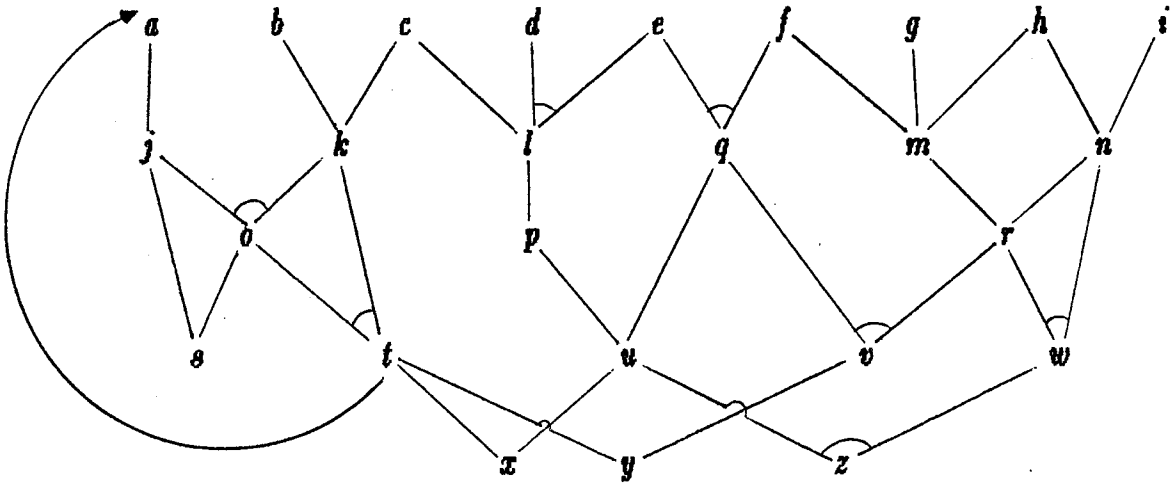

Figura 22: GC de uma BC Fictícia

As CIs definidas para as condições da BC são:

1. $(a, (e, f, g, h, i))$

2. $(t, (w))$

3. $(c, (f, g, h, i))$

4. $(d,(f,g,h,i))$

5. $(b,(f,g,h,i))$

6. $(x,(f,i))$

7. $(w,(a,d))$

VII.2 - Detecção de Encadeamentos Circulares

A entrada para o algoritmo NÍVEL é a BC representada através das seguintes listas:

1. $(j,(a))$

2. $(k,(b),(c))$

3. $(l,(c),(d,e))$

4. $(q,(e,f))$

5. $(m,(f),(g),(h))$

6. $(n,(h),(i))$

7. $(o,(j,k))$

8. $(p,(l))$

9. $(r,(m),(n))$

10. $(s,(j),(o))$

11. $(t,(k,o),(p))$

12. $(u,(p),(q))$

13. $(v, (q, r))$

14. $(w, (n, r))$

15. $(a, (t))$

16. $(x, (t), (u))$

17. $(y, (t), (v))$

18. $(z, (u, w))$

A execução do algoritmo NÍVEL sobre estas listas aponta os vértices a, j, o, s, t, x e y como impossíveis de ter seus níveis definidos, caracterizando a existência de encadeamento(s) circular(es). Para visualização do problema a figura 23 mostra a porção do GC correspondente a estes vértices.

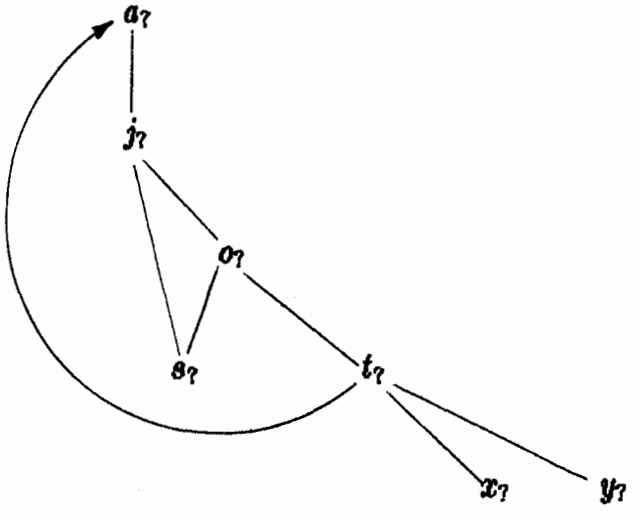

Figura 23: Vértices Envolvidos em Algum Encadeamento Circular

Vamos supor que a solução para o encadeamento seja a supressão da regra 1 (**SE** a **ENTÃO** j). O GC resultante com os níveis calculados é mostrado na figura 24.

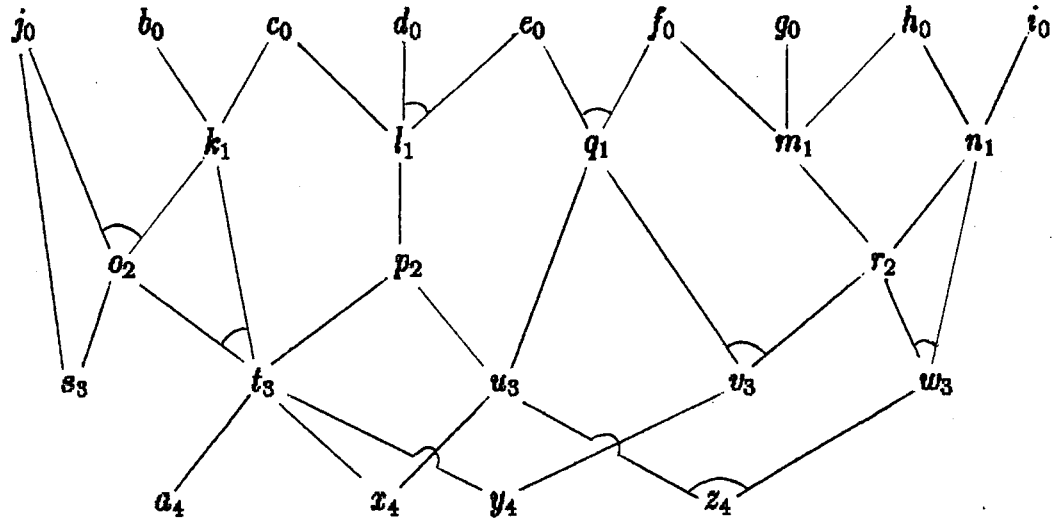

Figura 24: GC com Níveis Calculados

VII.3 - Geração da Base Equivalente Reduzida

O algoritmo BER processa as seguintes listas do GC por ordem crescente de nível:

1. (j_0)
2. (b_0)
3. (c_0)
4. (d_0)
5. (e_0)
6. (f_0)
7. (g_0)
8. (h_0)

9. (i_0)

10. $(k_1, (b), (c))$

11. $(l_1, (c), (d, e))$

12. $(q_1, (e, f))$

13. $(m_1, (f), (g), (h))$

14. $(n_1, (h), (i))$

15. $(o_2, (j, k))$

16. $(p_2, (l))$

17. $(r_2, (m), (n))$

18. $(s_3, (j), (o))$

19. $(t_3, (k, o), (p))$

20. $(u_3, (p), (q))$

21. $(v_3, (q, r))$

22. $(w_3, (n, r))$

23. $(a_4, (t))$

24. $(x_4, (t), (u))$

25. $(y_4, (t), (v))$

26. $(z_4, (u, w))$

As condições com nível 0 são simplesmente copiadas para a BER. Sua função é servir de base para a obtenção das árvores reduzidas para as condições com nível 1. Nas tabelas 2, 3, 4 e 5 são mostradas todas as listas de caminho referentes às árvores reduzidas geradas pelo algoritmo BER.

META	NR.SEQ.	LISTAS DE CAMINHO	CEI
j_0	1		
b_0	2		$\{f_0, g_0, h_0, i_0\}$
c_0	3		$\{e_0, f_0, g_0, h_0, i_0, w_3\}$
d_0	4		$\{f_0, g_0, h_0, i_0\}$
e_0	5		
f_0	6		
g_0	7		
h_0	8		
i_0	9		
k_1	10	b	$\{f_0, g_0, h_0, i_0\}$
	11	c	$\{e_0, f_0, g_0, h_0, i_0, w_3\}$
l_1	12	c	$\{e_0, f_0, g_0, h_0, i_0, w_3\}$
	13	d, e	$\{e_0, f_0, g_0, h_0, i_0, w_3\}$
q_1	14	e, f	$\{f_0, i_0\}$
m_1	15	f	
	16	g	
	17	h	$\{a_4, d_0\}$
n_1	18	h	$\{a_4, d_0\}$
	19	i	$\{a_4, d_0\}$
o_2	20	j, b	$\{e_0, f_0, g_0, h_0, i_0, w_3\}$
	21	j, c	$\{e_0, f_0, g_0, h_0, i_0, w_3\}$
p_2	22	c	$\{e_0, f_0, g_0, h_0, i_0, w_3\}$
	23	d, e	$\{e_0, f_0, g_0, h_0, i_0, w_3\}$
r_2	24	f	
	25	g	
	26	h	$\{a_4, d_0\}$
	27	h	$\{a_4, d_0\}$
	28	i	$\{a_4, d_0\}$
s_3	29	j	
	30	j, b	$\{e_0, f_0, g_0, h_0, i_0, w_3\}$

Tabela 2: Listas de Caminho Produzidas pelo Algoritmo BER (...)

META	NR.SEQ.	LISTAS DE CAMINHO	CEI
s_3	31	j, c	$\{e_0, f_0, g_0, h_0, i_0, w_3\}$
t_3	32	b, j, b	$\{e_0, f_0, g_0, h_0, i_0, w_3\}$
	33	b, j, c	$\{e_0, f_0, g_0, h_0, i_0, w_3\}$
	34	c, j, b	$\{e_0, f_0, g_0, h_0, i_0, w_3\}$
	35	c, j, c	$\{e_0, f_0, g_0, h_0, i_0, w_3\}$
	36	c	$\{e_0, f_0, g_0, h_0, i_0, w_3\}$
	37	d, e	$\{e_0, f_0, g_0, h_0, i_0, w_3\}$
u_3	38	c	$\{e_0, f_0, g_0, h_0, i_0, w_3\}$
	39	d, e	$\{e_0, f_0, g_0, h_0, i_0, w_3\}$
	40	e, f	$\{f_0, i_0\}$
v_3	41	e, f, f	$\{f_0, i_0\}$
	42	e, f, g	$\{f_0, i_0\}$
	43	e, f, h	$\{a_4, d_0, f_0, i_0\}$
	44	e, f, h	$\{a_4, d_0, f_0, i_0\}$
	45	e, f, i	$\{a_4, d_0, f_0, i_0\}$
w_3	46	h, f	$\{a_4, d_0\}$
	47	h, g	$\{a_4, d_0\}$
	48	h, h	$\{a_4, d_0\}$
	49	h, h	$\{a_4, d_0\}$
	50	h, i	$\{a_4, d_0\}$
	51	i, f	$\{a_4, d_0\}$
	52	i, g	$\{a_4, d_0\}$
	53	i, h	$\{a_4, d_0\}$
	54	i, h	$\{a_4, d_0\}$
	55	i, i	$\{a_4, d_0\}$
a_4	56	b, j, b	$\{e_0, f_0, g_0, h_0, i_0, w_3\}$
	57	b, j, c	$\{e_0, f_0, g_0, h_0, i_0, w_3\}$
	58	c, j, b	$\{e_0, f_0, g_0, h_0, i_0, w_3\}$
	59	c, j, c	$\{e_0, f_0, g_0, h_0, i_0, w_3\}$
	60	c	$\{e_0, f_0, g_0, h_0, i_0, w_3\}$

Tabela 3: (...) Listas de Caminho Produzidas pelo Algoritmo BER (...)

META	NR.SEQ.	LISTAS DE CAMINHO	CEI
a_4	61	d, e	$\{e_0, f_0, g_0, h_0, i_0, w_3\}$
x_4	62	b, j, b	$\{e_0, f_0, g_0, h_0, i_0, w_3\}$
	63	b, j, c	$\{e_0, f_0, g_0, h_0, i_0, w_3\}$
	64	c, j, b	$\{e_0, f_0, g_0, h_0, i_0, w_3\}$
	65	c, j, c	$\{e_0, f_0, g_0, h_0, i_0, w_3\}$
	66	c	$\{e_0, f_0, g_0, h_0, i_0, w_3\}$
	67	d, e	$\{e_0, f_0, g_0, h_0, i_0, w_3\}$
	68	c	$\{e_0, f_0, g_0, h_0, i_0, w_3\}$
	69	d, e	$\{e_0, f_0, g_0, h_0, i_0, w_3\}$
	70	e, f	$\{f_0, i_0\}$
y_4	71	b, j, b	$\{e_0, f_0, g_0, h_0, i_0, w_3\}$
	72	b, j, c	$\{e_0, f_0, g_0, h_0, i_0, w_3\}$
	73	c, j, b	$\{e_0, f_0, g_0, h_0, i_0, w_3\}$
	74	c, j, c	$\{e_0, f_0, g_0, h_0, i_0, w_3\}$
	75	c	$\{e_0, f_0, g_0, h_0, i_0, w_3\}$
	76	d, e	$\{e_0, f_0, g_0, h_0, i_0, w_3\}$
	77	e, f, f	$\{f_0, i_0\}$
	78	e, f, g	$\{f_0, i_0\}$
	79	e, f, h	$\{a_4, d_0, f_0, i_0\}$
	80	e, f, h	$\{a_4, d_0, f_0, i_0\}$
	81	e, f, i	$\{a_4, d_0, f_0, i_0\}$
z_4	82	c, h, f	$\{a_4, d_0, e_0, f_0, g_0, h_0, i_0, w_3\}$
	83	c, h, g	$\{a_4, d_0, e_0, f_0, g_0, h_0, i_0, w_3\}$
	84	c, h, h	$\{a_4, d_0, e_0, f_0, g_0, h_0, i_0, w_3\}$
	85	c, h, h	$\{a_4, d_0, e_0, f_0, g_0, h_0, i_0, w_3\}$
	86	c, h, i	$\{a_4, d_0, e_0, f_0, g_0, h_0, i_0, w_3\}$
	87	c, i, f	$\{a_4, d_0, e_0, f_0, g_0, h_0, i_0, w_3\}$
	88	c, i, g	$\{a_4, d_0, e_0, f_0, g_0, h_0, i_0, w_3\}$
	89	c, i, h	$\{a_4, d_0, e_0, f_0, g_0, h_0, i_0, w_3\}$
	90	c, i, h	$\{a_4, d_0, e_0, f_0, g_0, h_0, i_0, w_3\}$

Tabela 4: (...) Listas de Caminho Produzidas pelo Algoritmo BER (...)

A representação gráfica das árvores correspondentes às listas de caminho relacionadas, nível a nível, é mostrada nas figuras 25, 26, 27, 28 e 29.

META	NR.SEQ.	LISTAS DE CAMINHO	CEI
z_4	91	c, i, i	$\{a_4, d_0, e_0, f_0, g_0, h_0, i_0, w_3\}$
	92	d, e, h, f	$\{a_4, d_0, e_0, f_0, g_0, h_0, i_0, w_3\}$
	93	d, e, h, g	$\{a_4, d_0, e_0, f_0, g_0, h_0, i_0, w_3\}$
	94	d, e, h, h	$\{a_4, d_0, e_0, f_0, g_0, h_0, i_0, w_3\}$
	95	d, e, h, h	$\{a_4, d_0, e_0, f_0, g_0, h_0, i_0, w_3\}$
	96	d, e, h, i	$\{a_4, d_0, e_0, f_0, g_0, h_0, i_0, w_3\}$
	97	d, e, i, f	$\{a_4, d_0, e_0, f_0, g_0, h_0, i_0, w_3\}$
	98	d, e, i, g	$\{a_4, d_0, e_0, f_0, g_0, h_0, i_0, w_3\}$
	99	d, e, i, h	$\{a_4, d_0, e_0, f_0, g_0, h_0, i_0, w_3\}$
	100	d, e, i, h	$\{a_4, d_0, e_0, f_0, g_0, h_0, i_0, w_3\}$
	101	d, e, i, i	$\{a_4, d_0, e_0, f_0, g_0, h_0, i_0, w_3\}$
	102	e, f, h, f	$\{a_4, d_0, f_0, i_0\}$
	103	e, f, h, g	$\{a_4, d_0, f_0, i_0\}$
	104	e, f, h, h	$\{a_4, d_0, f_0, i_0\}$
	105	e, f, h, h	$\{a_4, d_0, f_0, i_0\}$
	106	e, f, h, i	$\{a_4, d_0, f_0, i_0\}$
	107	e, f, i, f	$\{a_4, d_0, f_0, i_0\}$
	108	e, f, i, g	$\{a_4, d_0, f_0, i_0\}$
	109	e, f, i, h	$\{a_4, d_0, f_0, i_0\}$
	110	e, f, i, h	$\{a_4, d_0, f_0, i_0\}$
	111	e, f, i, i	$\{a_4, d_0, f_0, i_0\}$

Tabela 5: (...) Listas de Caminho Produzidas pelo Algoritmo BER

$j \quad b \quad c \quad d \quad e \quad f \quad g \quad h \quad i$

Figura 25: BER com os Vértices do Nível 0

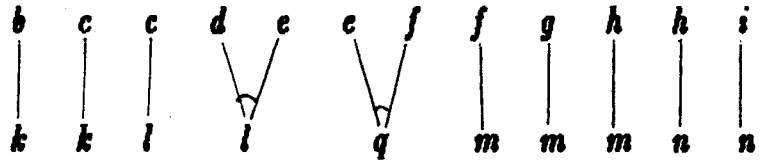

Figura 26: Árvores Reduzidas Geradas para os Vértices do Nível 1

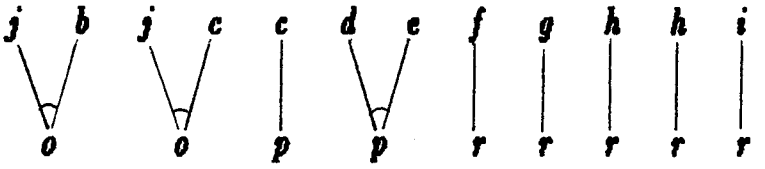

Figura 27: Árvores Reduzidas Geradas para os Vértices do Nível 2

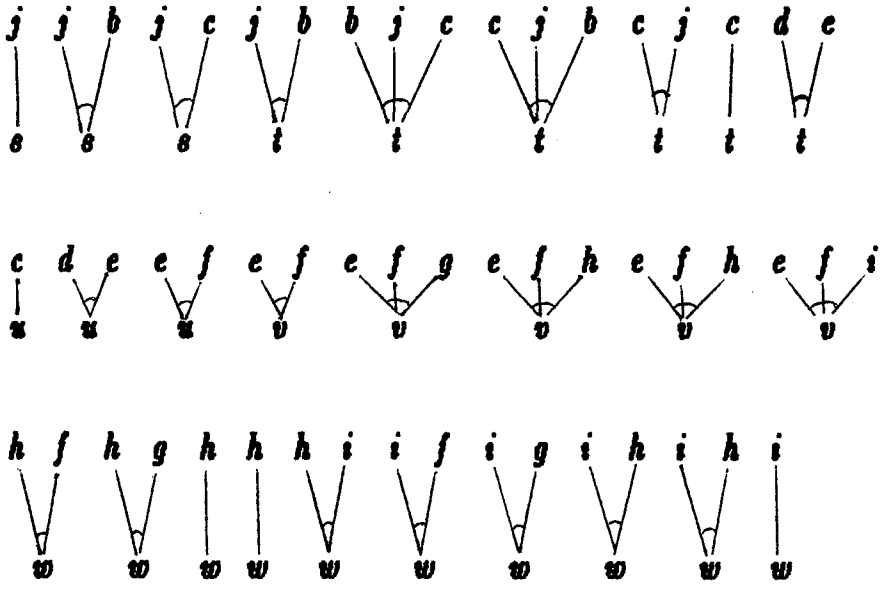

Figura 28: Árvores Reduzidas Geradas para os Vértices do Nível 3

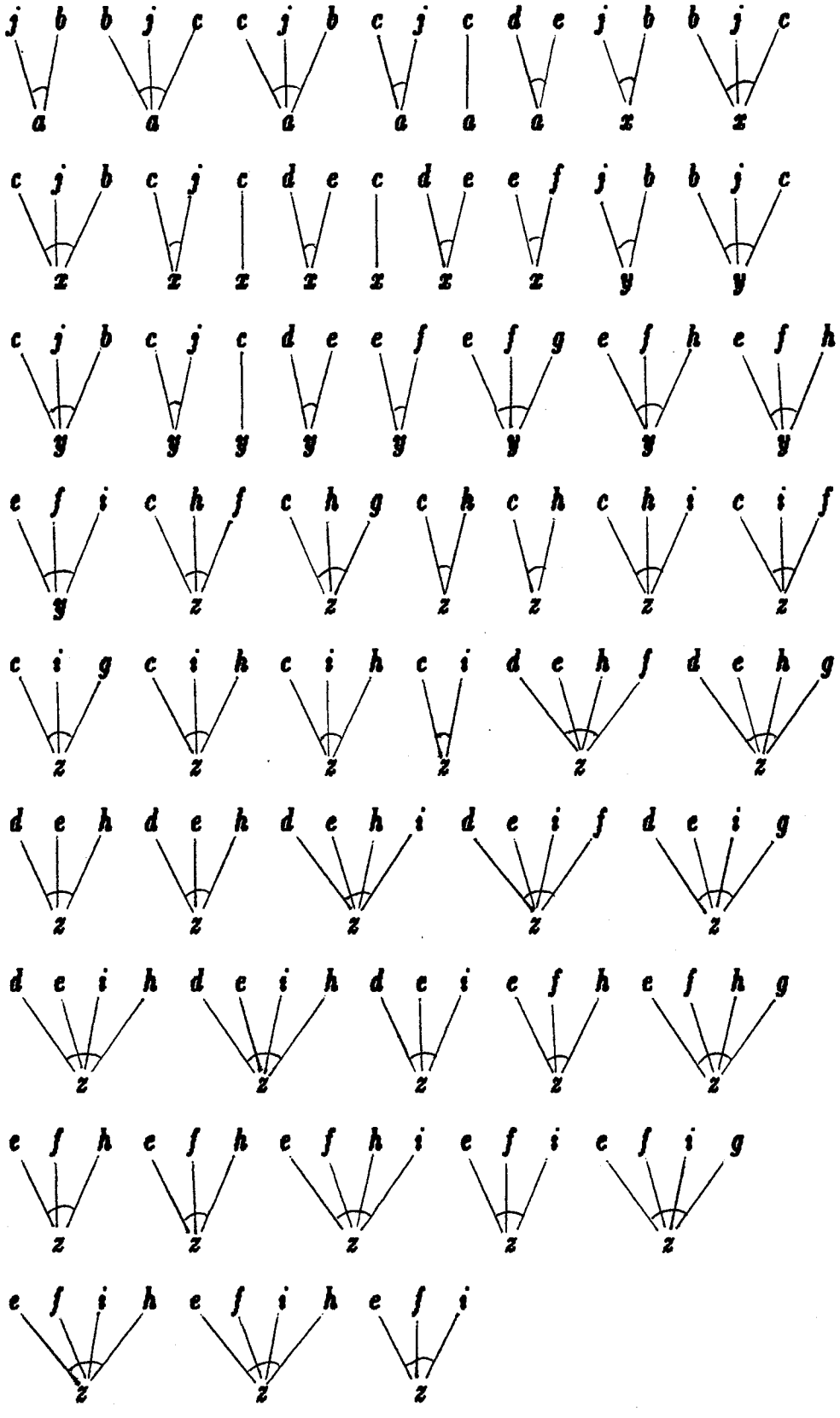

Figura 29: Árvores Reduzidas Geradas para os Vértices do Nível 4

No passo 3 as árvores reduzidas são rearranjadas de modo a que não haja vértices duplicados. Devido à dificuldade de se representar graficamente todas as relações entre os vértices, representamos apenas a condição não inicial t e suas relações com as condições iniciais na figura 30.

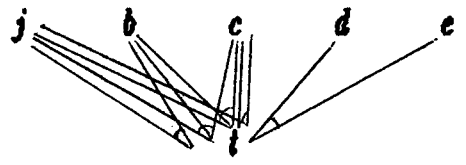

Figura 30: Parte da BER Referente à Condição t e seus Antecedentes

A lista $(t, (j, b), (b, j, c), (c, j, b), (c), (d, e))$ especifica as condições iniciais da BER a partir das quais se pode concluir t.

VII.4 - Detecção das Condições Desnecessárias

Neste ponto executamos o algoritmo CONC sobre as listas de caminho de cada vértice não inicial, geradas no passo 3 do BER e constantes das tabelas 2, 3, 4 e 5, apontando as regras com condições desnecessárias existentes nas tabelas de 6 a 10. As condições desnecessárias em uma mesma regra não são relacionadas pelo fato da sua identificação ser imediata. É o caso das regras 32, 35, 41, 48, 49, 55, 56, 59, 62, 65, 71, 74, 77, 84, 85, 91, 94, 95, 101, 102, 104, 105, 107 e 111.

META	REGRAS			
	NR.SEQ.	CONDIÇÕES	NR.SEQ.	CONDIÇÕES
r	26	*h*	27	*h*
s	29	*j*	30	*j, b*
	29	*j*	31	*j, c*
t	32	*j, b*	33	*b, j, c*
	32	*j, b*	34	*c, j, b*
	33	*b, j, c*	34	*c, j, b*
	33	*b, j, c*	35	*c, j*
	33	*b, j, c*	36	*c*
	34	*c, j, b*	35	*c, j*
	34	*c, j, b*	36	*c*
	35	*c, j*	36	*c*
v	41	*e, f*	42	*e, f, g*
	41	*e, f*	43	*e, f, h*
	41	*e, f*	44	*e, f, h*
	41	*e, f*	45	*e, f, i*
	43	*e, f, h*	44	*e, f, h*
w	46	*h, f*	48	*h*
	46	*h, f*	49	*h*
	47	*h, g*	48	*h*
	47	*h, g*	49	*h*
	48	*h*	49	*h*
	48	*h*	50	*h, i*
	48	*h*	53	*i, h*
	48	*h*	54	*i, h*
	49	*h*	50	*h, i*

Tabela 6: Regras com Condições Desnecessárias Detectadas na BER (...)

META	REGRAS			
	NR.SEQ.	CONDIÇOES	NR.SEQ.	CONDIÇOES
w	49	h	53	i, h
	49	h	54	i, h
	50	h, i	53	i, h
	50	h, i	54	i, h
	50	h, i	55	i
	51	i, f	55	i
	52	i, g	55	i
	53	i, h	54	i, h
	53	i, h	55	i
	54	i, h	55	i
a	56	j, b	57	b, j, c
	56	j, b	58	c, j, b
	57	b, j, c	58	c, j, b
	57	b, j, c	59	c, j
	57	b, j, c	60	c
	58	c, j, b	59	c, j
	58	c, j, b	60	c
	59	c, j	60	c
x	62	j, b	63	b, j, c
	62	j, b	64	c, j, b
	63	b, j, c	64	c, j, b
	63	b, j, c	65	c, j
	63	b, j, c	66	c
	63	b, j, c	68	c
	64	c, j, b	65	c, j
	64	c, j, b	66	c

Tabela 7: (...) Regras com Condições Desnecessárias Detectadas na BER (...)

META	REGRAS			
	NR.SEQ.	CONDIÇÕES	NR.SEQ.	CONDIÇÕES
x	64	c, j, b	68	c
	65	c, j	66	c
	65	c, j	68	c
	66	c	68	c
	67	d, e	69	d, e
y	71	j, b	72	b, j, c
	71	j, b	73	c, j, b
	72	b, j, c	73	c, j, b
	72	b, j, c	74	c, j
	72	b, j, c	75	c
	73	c, j, c	74	c, j
	73	c, j, b	75	c
	77	e, f	78	e, f, g
	77	e, f	79	e, f, h
	77	e, f	80	e, f, h
	77	e, f	80	e, f, h
	79	e, f, h	80	e, f, h
z	82	c, h, f	84	c, h
	82	c, h, f	85	c, h
	83	c, h, g	84	c, h
	83	c, h, g	85	c, h
	84	c, h	85	c, h
	84	c, h	86	c, h, i
	85	c, h	86	c, h, i
	86	c, h, i	89	c, i, h
	86	c, h, i	90	c, i, h

Tabela 8: (...) Regras com Condições Desnecessárias Detectadas na BER (...)

META	REGRAS			
	NR.SEQ.	CONDIÇÕES	NR.SEQ.	CONDIÇÕES
z	86	c, h, i	91	c, i
	88	c, i, g	91	c, i
	89	c, i, h	90	c, i, h
	89	c, i, h	91	c, i
	90	c, i, h	91	c, i
	92	d, e, h, f	94	d, e, h
	92	d, e, h, f	95	d, e, h
	92	d, e, h, f	102	e, f, h
	92	d, e, h, f	104	e, f, h
	92	d, e, h, f	105	e, f, h
	93	d, e, h, g	94	d, e, h
	93	d, e, h, g	95	d, e, h
	94	d, e, h	95	d, e, h
	94	d, e, h	96	d, e, h, i
	94	d, e, h	99	d, e, i, h
	94	d, e, h	100	d, e, i, h
	95	d, e, h	96	d, e, h, i
	95	d, e, h	99	d, e, i, h
	95	d, e, h	100	d, e, i, h
	96	d, e, h, i	99	d, e, i, h
	96	d, e, h, i	100	d, e, i, h
	96	d, e, h, i	101	d, e, i
	97	d, e, i, f	101	d, e, i
	97	d, e, i, f	107	e, f, i
	97	d, e, i, f	111	e, f, i
	98	d, e, i, g	101	d, e, i

Tabela 9: (...) Regras com Condições Desnecessárias Detectadas na BER (...)

META	CONDIÇÕES DESNECESSÁRIAS			
	NR.SEQ.	CONDIÇÕES	NR.SEQ.	CONDIÇÕES
z	99	d, e, i, h	100	d, e, i, h
	99	d, e, i, h	101	d, e, i
	100	d, e, i, h	101	d, e, i
	102	e, f, h	103	e, f, h, g
	102	e, f, h	104	e, f, h
	102	e, f, h	105	e, f, h
	102	e, f, h	106	e, f, h, i
	102	e, f, h	109	e, f, i, h
	102	e, f, h	110	e, f, i, h
	103	e, f, h, g	104	e, f, h
	103	e, f, h, g	105	e, f, h
	104	e, f, h	105	e, f, h
	104	e, f, h	106	e, f, h, i
	105	e, f, h	106	e, f, h, i
	106	e, f, h, i	107	e, f, i
	106	e, f, h, i	109	e, f, i, h
	106	e, f, h, i	100	e, f, i, h
	106	e, f, h, i	111	e, f, i
	107	e, f, i	108	e, f, i, g
	107	e, f, i	109	e, f, i, h
	107	e, f, i	110	e, f, i, h
	107	e, f, i	111	e, f, i
	108	e, f, i, g	111	e, f, i
	109	e, f, i, h	110	e, f, i, h
	109	e, f, i, h	111	e, f, i
	110	e, f, i, h	111	e, f, i

Tabela 10: (...) Regras com Condições Desnecessárias Detectadas na BER

Vamos supor, para simplificar a exposição subseqüente, que o engenheiro do conhecimento decida manter na BER somente as listas de caminho menos restritivas, ou seja aquelas que possuem um número menor de condições a serem satisfeitas para que se conclua determinada meta. Desta forma, devem ser eliminadas as listas com os números de seqüencia: 26, 30, 31, 33, 34, 35, 42, 43, 44, 45, 46, 47, 49, 50, 51, 52, 53, 54, 57, 58, 59, 63, 64, 65, 67, 68, 72, 73, 74, 78, 79, 80, 81, 82, 83, 85, 86, 87, 88, 89, 90, 92, 93, 95, 96, 97, 98, 99, 100, 103, 104, 105, 106, 108, 109, 110 e 111. As listas restantes são relacionadas nas tabelas 11 e 12 e servem de entrada para o algoritmo CONFL que detecta regras incompatíveis.

META	NR.SEQ.	LISTAS DE CAMINHO	CEI
k_1	10	b	$\{f_0, g_0, h_0, i_0\}$
	11	c	$\{e_0, f_0, g_0, h_0, i_0, w_3\}$
l_1	12	c	$\{e_0, f_0, g_0, h_0, i_0, w_3\}$
	*13	d, e	$\{e_0, f_0, g_0, h_0, i_0, w_3\}$
q_1	14	e, f	
m_1	15	f	
	16	g	
	17	h	$\{a_4, d_0\}$
n_1	18	h	$\{a_4, d_0\}$
	19	i	$\{a_4, d_0\}$
o_2	20	j, b	$\{e_0, f_0, g_0, h_0, i_0, w_3\}$
	21	j, c	$\{e_0, f_0, g_0, h_0, i_0, w_3\}$
p_2	22	c	$\{e_0, f_0, g_0, h_0, i_0, w_3\}$
	*23	d, e	$\{e_0, f_0, g_0, h_0, i_0, w_3\}$
r_2	24	f	
	25	g	
	27	h	$\{a_4, d_0\}$
	28	i	$\{a_4, d_0\}$

Tabela 11: Listas da BER após Eliminadas as Condições Desnecessárias (...)

META	NR.SEQ.	LISTAS DE CAMINHO	CEI
s_3	29	j	
t_3	32	j, b	$\{e_0, f_0, g_0, h_0, i_0, w_3\}$
	36	c	$\{e_0, f_0, g_0, h_0, i_0, w_3\}$
	*37	d, e	$\{e_0, f_0, g_0, h_0, i_0, w_3\}$
u_3	38	c	$\{e_0, f_0, g_0, h_0, i_0, w_3\}$
	*39	d, e	$\{e_0, f_0, g_0, h_0, i_0, w_3\}$
	*40	e, f	$\{f_0, i_0\}$
v_3	*41	e, f	$\{f_0, i_0\}$
w_3	48	h	$\{a_4, d_0\}$
	55	i	$\{a_4, d_0\}$
a_4	56	j, b	$\{e_0, f_0, g_0, h_0, i_0, w_3\}$
	60	c	$\{e_0, f_0, g_0, h_0, i_0, w_3\}$
	*61	d, e	$\{e_0, f_0, g_0, h_0, i_0, w_3\}$
x_4	62	j, b	$\{e_0, f_0, g_0, h_0, i_0, w_3\}$
	66	c	$\{e_0, f_0, g_0, h_0, i_0, w_3\}$
	*69	d, e	$\{e_0, f_0, g_0, h_0, i_0, w_3\}$
	*70	e, f	$\{f_0, i_0\}$
y_4	71	j, b	$\{e_0, f_0, g_0, h_0, i_0, w_3\}$
	75	c	$\{e_0, f_0, g_0, h_0, i_0, w_3\}$
	*76	d, e	$\{e_0, f_0, g_0, h_0, i_0, w_3\}$
	*77	e, f	$\{f_0, i_0\}$
z_4	*84	c, h	$\{a_4, d_0, e_0, f_0, g_0, h_0, i_0, w_3\}$
	*91	c, i	$\{a_4, d_0, e_0, f_0, g_0, h_0, i_0, w_3\}$
	*94	d, e, h	$\{a_4, d_0, e_0, f_0, g_0, h_0, i_0, w_3\}$
	*101	d, e, i	$\{a_4, d_0, e_0, f_0, g_0, h_0, i_0, w_3\}$
	*102	e, f, h	$\{a_4, d_0, f_0, i_0\}$
	*107	e, f, i	$\{a_4, d_0, f_0, i_0\}$

Tabela 12: (...) Listas da BER após Eliminadas as Condições Desnecessárias

VII.5 - Detecção de Regras Incompatíveis

A aplicação do algoritmo CONFL, que atende à finalidade desta seção, é feita sobre as listas de caminho e as CEIs das tabelas 11 e 12 As possíveis regras incompatíveis podem ser detectadas nos passos a ou b do algoritmo. Na tabela 13 são mostradas as CEIs onde existem condições com *nível = 0* mas que não possuem algum antecedente em cada lista de caminho também na CEI. Esta tabela foi montada a partir dos resultados do passo a.

AS SEGUINTES REGRAS:				SÃO INCOMPATÍVEIS COM:		
NR.	META	COND.	CEI	NR.	META	COND.
17	m	h	$\{a,d\}$	56	a	j,b
18	n	h	$\{a,d\}$	60	a	c
19	n	i	$\{a,d\}$			
27	r	h	$\{a,d\}$			
28	r	i	$\{a,d\}$			
48	w	h	$\{a,d\}$			
55	w	i	$\{a,d\}$			
102	z	e,f,h	$\{a,d,f,i\}$			
107	z	e,f,i	$\{a,d,f,i\}$			

Tabela 13: Incompatibilidades Detectadas no Passo a

Nas tabelas 14 e 15 são relacionadas as incompatibilidades decorrentes da falta de reflexividade das CEIs e foi montada a partir dos resultados do passo b. As regras incompatíveis consigo mesmas, também detectadas no passo b, estão assinaladas com "*" nas tabelas de entrada do algoritmo CONFL.

AS SEGUINTES REGRAS:				SAO INCOMPATÍVEIS COM:			
NR.	META	COND.	CEI	NR.	META	COND.	CEI
10	k	b	$\{f,g,h,i\}$	14	q	e,f	
				15	m	f	
				16	m	g	
				17	m	h	$\{a,d\}$
				18	n	h	$\{a,d\}$
				19	n	i	$\{a,d\}$
				24	r	f	
				25	r	g	
				27	r	h	$\{a,d\}$
				28	r	i	$\{a,d\}$
				40	u	e,f	$\{f,i\}$
				41	v	e,f	$\{f,i\}$
				48	w	h	$\{a,d\}$
				55	w	i	$\{a,d\}$
				70	x	e,f	$\{f,i\}$
				77	y	e,f	$\{f,i\}$
				84	z	c,h	$\{a,d,e,f,g,h,i,w\}$
				91	z	c,i	$\{a,d,e,f,g,h,i,w\}$
				94	z	d,e,h	$\{a,d,e,f,g,h,i,w\}$
				101	z	d,e,i	$\{a,d,e,f,g,h,i,w\}$
				102	z	e,f,h	$\{a,d,f,i\}$
				107	m	e,f,i	$\{a,d,f,i\}$
11	k	c	$\{e,f,g,h,i,w\}$	14	q	e,f	
12	l	c	$\{e,f,g,h,i,w\}$	15	m	f	
20	o	j,b	$\{e,f,g,h,i,w\}$	16	m	g	
21	o	j,c	$\{e,f,g,h,i,w\}$	17	m	h	$\{a,d\}$
22	p	c	$\{e,f,g,h,i,w\}$	18	n	h	$\{a,d\}$
32	t	j,b	$\{e,f,g,h,i,w\}$	19	n	i	$\{a,d\}$
36	t	c	$\{e,f,g,h,i,w\}$	24	r	f	
38	u	c	$\{e,f,g,h,i,w\}$	25	r	g	
56	a	j,b	$\{e,f,g,h,i,w\}$	27	r	h	$\{a,d\}$
60	a	c	$\{e,f,g,h,i,w\}$	28	r	i	$\{a,d\}$
62	x	j,b	$\{e,f,g,h,i,w\}$	40	u	e,f	$\{f,i\}$
66	x	c	$\{e,f,g,h,i,w\}$	41	v	e,f	$\{f,i\}$
71	y	j,b	$\{e,f,g,h,i,w\}$	48	w	h	$\{a,d\}$
75	y	c	$\{e,f,g,h,i,w\}$	55	w	i	$\{a,d\}$
				70	x	e,f	$\{f,i\}$
				77	y	e,f	$\{f,i\}$

Tabela 14: Incompatibilidades Detectadas no Passo b (...)

AS SEGUINTES REGRAS:				SAO INCOMPATÍVEIS COM:			
NR.	META	COND.	CEI	NR.	META	COND.	CEI
				84	z	c,h	$\{a,d,e,f,g,h,i,w\}$
				91	z	c,i	$\{a,d,e,f,g,h,i,w\}$
				94	z	d,e,h	$\{a,d,e,f,g,h,i,w\}$
				101	z	d,e,i	$\{a,d,e,f,g,h,i,w\}$
				102	z	e,f,h	$\{a,d,f,i\}$
				107	m	e,f,i	$\{a,d,f,i\}$
13	l	d,e	$\{e,f,g,h,i,w\}$	14	q	e,f	
23	p	d,e	$\{e,f,g,h,i,w\}$	15	m	f	
37	t	d,e	$\{e,f,g,h,i,w\}$	16	m	g	
39	u	d,e	$\{e,f,g,h,i,w\}$	24	r	f	
61	a	d,e	$\{e,f,g,h,i,w\}$	25	r	g	
69	x	d,e	$\{e,f,g,h,i,w\}$	40	u	e,f	$\{f,i\}$
76	y	d,e	$\{e,f,g,h,i,w\}$	41	v	e,f	$\{f,i\}$
				70	x	e,f	$\{f,i\}$
				77	y	e,f	$\{f,i\}$
40	u	e,f	$\{f,i\}$	14	q	e,f	
41	v	e,f	$\{f,i\}$	15	m	f	
70	x	e,f	$\{f,i\}$	19	n	i	$\{a,d\}$
77	y	e,f	$\{f,i\}$	24	r	f	
				28	r	i	$\{a,d\}$
				55	w	i	$\{a,d\}$
84	z	c,h	$\{a,d,e,f,g,h,i,w\}$	14	q	e,f	
91	z	c,i	$\{a,d,e,f,g,h,i,w\}$	15	m	f	
94	z	d,e,h	$\{a,d,e,f,g,h,i,w\}$	16	m	g	
101	z	d,e,i	$\{a,d,e,f,g,h,i,w\}$	24	r	f	
				25	r	g	
102	z	e,f,h	$\{a,d,f,i\}$	14	q	e,f	
107	z	e,f,i	$\{a,d,f,i\}$	15	m	f	
				19	n	i	$\{a,d\}$
				24	r	f	
				28	r	i	$\{a,d\}$
				55	w	i	$\{a,d\}$

Tabela 15: (...) Incompatibilidades Detectadas no Passo b

VIII - Conclusões

A partir do que foi apresentado podemos delinear conclusões relativas a dois aspectos: a utilidade do modelo na detecção de problemas na construção de BCs e a aplicabilidade da BER no processo de inferência. Neste capítulo vamos tentar identificar vantagens e desvantagens quanto a estes pontos. Além disso, no decorrer do trabalho, foram identificadas necessidades que apontamos como sugestões para pesquisa.

VIII.1 - Vantagens

Como instrumento de apoio ao engenheiro do conhecimento/especialista na modelagem do conhecimento, os procedimentos apresentados são úteis na medida em que apontam os problemas da BER que, com exceção de algumas redundâncias, são também problemas da BC original.

No caso específico de aplicações baseadas em raciocínio preciso onde um tempo de resposta mínimo é fundamental (exemplo: controle de processo baseado em conhecimento) o uso da BER no processo de inferência proporciona um incremento significativo na performance da busca por dois motivos:

1. Só existem dois níveis: o das condições iniciais e o das condições finais. Desta forma, qualquer conclusão pode ser obtida com uma única inferência a partir dos dados iniciais; e
2. Para o caso de encadeamento regressivo com várias opções de caminho, podem ser criadas heurísticas que apontem o caminho com menos condições iniciais a serem satisfeitas e que, portanto, pode ser o de menor custo, considerando-se custo na definição de NILSSON(1982).

Uma outra vantagem é observada em aplicações baseadas em raciocínio impreciso: o uso da BER facilita a identificação de casos como o da figura 31. Nesta situação, o cálculo do CF de c é feito considerando-se, indevidamente, o CF de a duas vezes.

Figura 31: Condição desnecessária que pode ser identificada com o uso da BER

VIII.2 - Desvantagens

O problema principal do modelo na detecção de problemas diz respeito às redundâncias apontadas a partir da BER e que nem sempre ocorrem na BC original. A identificação daquelas comuns às duas bases é um processo que demanda um esforço muito grande por parte do engenheiro do conhecimento.

Em relação ao uso da BER no processo de inferência temos que o seu maior obstáculo é a impossibilidade de se justificar as conclusões, uma vez que o rastreamento é impossível. Uma solução seria manter a BC original disponível para os casos em que o rastreamento é importante. Neste caso poderá haver uma sobrecarga no espaço disponível para a aplicação.

VIII.3 - Sugestões de Linhas de Pesquisa

Dos assuntos relacionados com o presente trabalho, consideramos como de maior relevância:

1. Estudo de uma forma de se determinar quando uma condição desnecessária na BER também o é na BC;
2. Especificação de um modelo para validação de bases de conhecimento onde se utilize raciocínio impreciso. Neste caso, quanto à detecção de problemas

na BC, deve-se considerar que condições incompatíveis podem ocorrer simultaneamente, uma vez que o seu status de veracidade é expresso por alguma medida de incerteza (não absoluta como verdadeiro ou falso). Quanto à BER, para viabilizar o seu uso no processo de inferência, é sugerida a utilização de fórmulas associadas a cada conjunto de condições iniciais a partir do qual se pode inferir determinada condição. Estas fórmulas conteriam os FCs de todas as condições envolvidas naquele caminho particular. Assim, a vantagem da melhor performance da BER em termos de busca é mantida com o uso de raciocínio impreciso; e

3. Análise de variações do conceito de nível como heurísticas de uso geral. Esta sugestão é resultado de observações que fizemos ao utilizarmos este conceito de nível. A exemplo da heurística sugerida em VIII.1, para a qual um nome mais apropriado seria "peso", acreditamos que outras variações podem ser introduzidas para direcionar melhor o foco do mecanismo de inferência.

Bibliografia

ALVARENGA,Rogério (1987); *Especificação e Desenvolvimento de um Protótipo de um Gerenciador de Bases de Conhecimento*; Tese de Mestrado; Universidade Federal do Rio de Janeiro - COPPE; Programa de Engenharia de Sistemas e Computação

BARR,A. e FEIGENBAUM,E.A. (1981); *The Handbook of Artificial Intelligence*; California; Kaufman

CAJUEIRO,Eduardo V.M. (1987); *Sistemas Especialistas: Um Estudo de Caso na Agricultura*; Tese de Mestrado; Universidade Federal do Rio de Janeiro - COPPE; Programa de Engenharia de Sistemas e Computação

CHAKRABARTI,P.P;GHOSE,S. e DeSARKAR,S.C. (1988); "Admissibility of AO* When Heuristics Overestimate"; *Artificial Intelligence (An International Journal)* 34(1988):97-113

DAMSKY, J.C.B. e OMAR, N. (1987); "Um Sistema para Análise de uma Base de Conhecimento"; *Anais do IV Simpósio Brasileiro de Inteligência Artificial*; Uberlândia - MG; 13 a 16 de outubro; pp. 279-286

DAVIS,Randall (1979); "Interactive Transfer of Expertise in Rule-Based Expert Systems"; *Artificial Intelligence (An International Journal)* 12(1979); pp. 121-157

DOMINGUES,Nélio (1988); "Redução de Redundâncias em Bases de Conhecimento"; *Anais do XXI Congresso Brasileiro de Informática*; Rio de Janeiro - RJ; 22 a 26 de agosto; pp. 550-555

DREYFUS, H. e DREYFUS, S. (1986); "Why Expert Systems Do Not Exhibit Expertise"; *IEEE EXPERT* (Summer); pp.86-90

DUDA,Richard O. e SHORTLIFFE,Edward H. (1983); "Expert Systems Research"; *SCIENCE* 220(4594) April; pp. 261-268

ENDERTON,Herbert B. (1972); *A Mathematical Introduction to Logic*; New York; Academic Press

FORSYTH,Richard (1984); *Expert Systems: Principles and Case Studies*; London; Chapman and Hall

FRENZEL JR.,Louis (1987); *Crash Course in Artificial Intelligence and Expert Systems*; New York

HARMON,P. e KING,D. (1985); *Expert Systems - Artificial Intelligence in Business*; New York; John Wiley and Sons

HOFSTADTER,Douglas R. (1979); *Gödel, Escher, Bach*; Basic Books; USA

LEVINE,Robert I., DRANG,Diane E. e EDELSON,B. (1988); *Inteligência Artificial e Sistemas Especialistas - Aplicações e Exemplos Práticos*; São Paulo; McGraw-Hill

LUCENA,Carlos (1987); *Inteligência Artificial e Engenharia de Software*; Rio de Janeiro; Jorge Zahar Editor Ltda.

MAHANTI,A. e BAGCHI,A. (1985); "AND/OR Graphs Heuristic Search Methods"; *Journal of the Association for Computing Machinery* 32(1); pp. 28-51

MANNA,Zohar (1987); *Mathematical Theory of Computation*; USA; McGraw-Hill

NGUYEN,Tin A., PERKINS,Walton A., LAFFEY,Thomas J. e PECORA Deanne (1987); "Knowledge Base Verification"; *AI Magazine* (Summer); pp. 69-75

NILSSON,Nils J. (1971); *Problem Solving Methods in Artificial Intelligence*; New York; McGraw-Hill

NILSSON,Nils J. (1982); *Principles of Artificial Intelligence*; Berlin; Springer-Verlag

PINHO,Antônio de A. (1988); "BACO-Um Gerenciador de Bases de Conhecimento para Desenvolvimento de Sistemas Especialistas do Tipo Diagnóstico"; Notas de Aula

PINHO,Antônio de A. e TELES,Aníbal(1988); "Consistência e Completude em Bases de Conhecimento", *Anais do V Simpósio Brasileiro de Inteligência Artificial*; Natal-RN; 7 a 11 de novembro

RICH,Elaine (1983);*Artificial Intelligence (International Edition)*; Singapore; McGraw-Hill

SESCONETTO BORGES, Adelina A. (1989); *Análise de Alguns Métodos de Raciocínio Impreciso para Sistemas Especialistas Baseados em Regras*; Tese de Mestrado; Universidade Federal do Rio de Janeiro - COPPE; Programa de Engenharia de Sistemas e Computação

SHORTLIFFE,Edward H. (1984); "Details of the Consultation System"; *Rule-Based Expert Systems (The MYCIN Experiments of the Stanford Heuristic Programming Project)*; Editado por Bruce G. Buchanan e Edward H. Shortliffe; Addison-Wesley Publishing Co.

SHORTLIFFE,Edward H. e BUCHANAN,Bruce G. (1984);*Rule-Based Expert Systems (The MYCIN Experiments of the Stanford Heuristic Programming Project)*; Addison-Wesley Publishing Co.

SUWA,Motoi, SCOTT,Carlisle A. e SHORTLIFFE,Edward H. (1984);"Completeness and Consistency in a Rule-Based Expert System"; *Rule-Based Expert Systems (The MYCIN Experiments of the Stanford Heuristic Programming Project)*; Editado por Bruce G. Buchanan e Edward H. Shortliffe; Addison-Wesley Publishing Co.

SZWARCFITER,Jayme L. (1984); *Grafos e Algoritmos Computacionais*; Rio de Janeiro; Campus

VASCONCELOS, Wamberto W.M.P. de (1989); *O Tempo Como Modelo: A Aplicação de Lógicas Temporais na Especificação Formal de Sistemas Distribuídos*; Tese de Mestrado; Universidade Federal do Rio de Janeiro - COPPE; Programa de Engenharia de Sistemas e Computação

VELOSO,Paulo S.A., SANTOS,Clésio S. dos, FURTADO,Antonio L, AZEREDO,Paulo A. (1983); *Estruturas de Dados*; Rio de Janeiro; Campus

WEISS, S.M. e KULIKOWSKI. C.A. (1988); *Guia Prático para Projetar Sistemas Especialistas*; Rio de Janeiro; LTC Editora S.A.

WINSTON,Patrick H. (1984); *Artificial Intelligence*; USA; Addison Wesley