UNIVERSIDADE DE SÃO PAULO

Instituto de Ciências Matemáticas e de Computação

ISSN 0103-2569

Descrição de um Algoritmo para Generalização de Regras de Associação

Marcos Aurélio Domingues

Solange Oliveira Rezende

Nº 228

RELATÓRIOS TÉCNICOS

São Carlos - SP

Março/2004

Descrição de um Algoritmo para Generalização de Regras de Associação[1]

Marcos Aurélio Domingues

Solange Oliveira Rezende

Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação
Departamento de Ciências de Computação e Estatística
C.P. 668, 13560-970 - São Carlos, SP - Brasil
e-mail: {mad, solange}@icmc.usp.br

Resumo: Um dos principais objetivos do processo de Mineração de Dados é que seus usuários finais possam analisar, compreender e utilizar o conhecimento extraído em Sistemas Inteligentes ou como apoio em processos de tomada de decisão. Entretanto, muitos dos algoritmos utilizados geram uma enorme quantidade de padrões, dificultando consideravelmente sua análise. Esse problema recebe uma maior ênfase em Regras de Associação, uma vez que essa técnica de Mineração de Dados procura identificar todos os comportamentos intrínsecos do conjunto de dados. Uma abordagem que pode auxiliar a análise de Regras de Associação é o uso de taxonomias na fase de Pós-processamento do conhecimento. Neste trabalho são detalhadamente descritos o algoritmo $\mathcal{IGART}$, que utiliza taxonomias para generalizar Regras de Associação, e o módulo computacional *RulEE-GAR*, que provê funcionalidades para analisar as regras generalizadas.

Palavras Chaves: Mineração de Dados, Pós-Processamento, Generalização de Regras, Taxonomias, Regras de Associação.

Março 2004

[1]Trabalho realizado com apoio da FAPESP e CAPES.

Sumário

1 Introdução **1**

2 Regras de Associação **2**

3 O Algoritmo $\mathcal{IGART}$ **3**

3.1 Função gera-subconjuntos 9
3.2 Função generaliza-regras 10
3.3 Função generaliza-uma-regra 12
3.4 Função desfaz-última-generalização 13
3.5 Função remove-regras-repetidas 13
3.6 Função ordena-lexicograficamente 14
3.7 Função calcula-TC 15
3.8 Exemplo de Funcionamento do Algoritmo $\mathcal{IGART}$ 17

4 Implementação do Algoritmo $\mathcal{IGART}$ **22**

4.1 Biblioteca de Classes do Algoritmo $\mathcal{IGART}$ 22
4.1.1 Classe ARtaxonomy 23
4.1.2 Classe ARrule 24
4.1.3 Classe ARigart 27
4.2 Arquivos de Entrada e Saída 31
4.2.1 Arquivos de Entrada 31
4.2.2 Arquivos de Saída 34
4.3 Interface com o Usuário 36

5 O Módulo Computacional *RulEE-GAR* **37**

5.1 Alterações na Base de Dados 38
5.2 Alterações no Módulo de Gerenciamento 43
5.2.1 Classe Rule 43
5.2.2 Classe Project 44
5.2.3 Classe Ruleset 44
5.3 Descrição da Interface do Módulo *RulEE-GAR* 47
5.4 Generalização de Regras de Associação Utilizando o Módulo *RulEE-GAR* 50
5.5 Análise de Regras de Associação Generalizadas Utilizando o Módulo *RulEE-GAR* 53

6 Considerações Finais **59**

A A Sintaxe Padrão para Conjuntos de Dados Transacionais **59**

Referências **61**

Lista de Figuras

1 Exemplo de uma taxonomia para vestuário 4
2 Generalização de Regras de Associação usando uma taxonomia 4
3 Generalização de Regras de Associação usando duas taxonomias 5
4 Processo proposto para generalização de Regras de Associação 7
5 Exemplo 1 – Conjunto de Taxonomias $\mathcal{T}$ 18
6 Relacionamentos e métodos das classes do algoritmo $\mathcal{IGART}$ 31
7 Exemplo de um arquivo de dados transacionais .apr.data 31
8 Exemplo de um arquivo de regras .apr.dcar 32
9 Exemplo do arquivo de taxonomias .tax com as taxonomias da Figura 10 32
10 Um conjunto de taxonomias $\mathcal{T}$ com 4 taxonomias 33
11 Um conjunto $\mathcal{T}$ contendo uma taxonomia de 3 níveis 33
12 Exemplo do arquivo .tax contendo a taxonomia de 3 níveis da Figura 11 . 33
13 Exemplo de um arquivo de relatório .gar.report 34
14 Exemplo de um arquivo de regras generalizadas .gar.dcar 35
15 Exemplo do relatório .gar.dcar.report 35
16 Contextualização e funcionalidades do módulo computacional *RulEE-GAR* no ambiente *RulEE* 38
17 Diagrama físico da Base de Dados do ambiente *RulEE* 42
18 Relacionamento entre as classes do ambiente *RulEE*. 47
19 *Login* no ambiente *RulEE* 47
20 *Menu* principal do ambiente *RulEE* 47
21 Interface para generalização de Regras de Associação 48
22 Interface para análise das Regras de Associação generalizadas 48
23 Sistema de ajuda (*Help*) do módulo *RulEE-GAR* 49
24 Seleção de um conjunto de Regras de Associação para ser generalizado . 51
25 Definição das taxonomias para a generalização das regras 52
26 Término do processo de generalização das Regras de Associação 53
27 Consulta a uma base de Regras de Associação generalizadas 55
28 Visualização do arquivo de dados transacionais (*link Data Set*) 56
29 Visualização do arquivo de regras originais (*link Rule Set*) 56
30 Visualização do arquivo de regras generalizadas (*link Generalized Rule Set*) 56

31 Visualização do arquivo de taxonomias (*link Taxonomy Set*) 56
32 Tela de visualização de uma Regra de Associação generalizada expandida 57
33 Tela de visualização das Regras de Associação que geraram uma regra generalizada 57
34 Tela de visualização dos valores das medidas suporte e confiança de uma regra generalizada 58
35 Tela de visualização de itens originais a partir de um item generalizado . 58
36 Representação de um arquivo de dados transacionais .trans 60
37 Exemplo de um arquivo de dados transacionais 60
38 Produções da gramática G para Bases de Dados transacionais 61

Lista de Tabelas

1 Tabela de Contingência 15
2 Exemplo 1 – Relação de itens comprados por transação 17
3 Exemplo 1 – Conjunto de Regras de Associação R 17
4 Exemplo 1 – Subconjuntos de Regras de Associação com conseqüentes semelhantes 18
5 Exemplo 1 – Conjunto de Regras de Associação generalizadas Rg 22

Lista de Algoritmos

1 $\mathcal{IGART}$ 8
2 Função gera-subconjuntos 9
3 Função generaliza-regras 10
4 Função generaliza-uma-regra 12
5 Função desfaz-última-generalização 13
6 Função remove-regras-repetidas 14
7 Função ordena-lexicograficamente 14
8 Função calcula-TC 16

1 Introdução

A evolução da computação impulsionada pelo aumento do poder de processamento dos computadores, pelo armazenamento contínuo de grandes quantidades de dados a baixo custo, pela introdução de novas tecnologias de transmissão e disseminação de dados etc., tem dado às organizações a capacidade de armazenar informações detalhadas sobre cada transação que efetuam, gerando grandes Bases de Dados. As organizações reconhecem o valor das informações contidas em suas bases e têm investido na aquisição e desenvolvimento de ferramentas de análise que produzam informações e conhecimento úteis.

Durante anos, métodos manuais têm sido utilizados para transformar dados em conhecimento. Porém, o uso desses métodos tem se tornado dispendioso (em termos financeiros e de tempo), subjetivo e inviável, quando aplicados a grandes Bases de Dados.

Devido aos problemas com os métodos manuais, tornou-se necessário o desenvolvimento de processos de análise automática, como o Processo de Extração de Conhecimento de Bases de Dados ou Mineração de Dados. Esse processo, de natureza iterativa e interativa, tem despontado por seu desempenho em diversos domínios, na extração de padrões válidos, novos, e potencialmente úteis dos dados (Fayyad et al., 1996).

Embora tenha se tornado necessária a utilização do processo de Mineração de Dados para extrair padrões a partir de dados, sua aplicação pode gerar uma elevada quantidade de padrões (conhecimento), muitos dos quais podem não ser importantes, relevantes ou interessantes para o usuário. Fornecer ao usuário uma grande quantidade de padrões não é produtivo pois, geralmente, ele procura poucos padrões que sejam interessantes. Portanto, é de vital importância o desenvolvimento de técnicas de apoio no sentido de fornecer aos usuários apenas os padrões mais interessantes (Silberschatz & Tuzhilin, 1995).

O problema de se gerar grandes quantidades de padrões recebe uma maior ênfase em Regras de Associação, uma das técnicas de Mineração de Dados que recentemente tem despertado grande interesse (Baesens et al., 2000). Na área acadêmica pesquisas vêm sendo desenvolvidas com essa técnica e as organizações têm utilizado os resultados no comércio, em contratos de seguro, na saúde, no geoprocessamento, na biologia molecular entre outras áreas (Liu et al., 2000; Clementini et al., 2000; Semenova et al., 2001).

Uma abordagem para solucionar o problema da grande quantidade de padrões extraídos pela técnica de Regras de Associação é o uso de taxonomias (Srikant & Agrawal, 1997; Liu et al., 2000; Adamo, 2001). As taxonomias refletem uma visão coletiva ou individual de como os itens podem ser hierarquicamente classificados, podendo ser utilizadas para

eliminar regras não interessantes e/ou redundantes (Adamo, 2001).

Diante desse contexto, neste relatório são descritos o algoritmo $\mathcal{IGART}$ e o módulo computacional *RulEE-GAR*. O algoritmo $\mathcal{IGART}$ (*Interactive Generalization of Association Rules using Taxonomies* – Generalização Interativa de Regras de Associação usando Taxonomias) utiliza taxonomias para generalizar Regras de Associação. Já o módulo *RulEE-GAR*, além de facilitar o uso do algoritmo $\mathcal{IGART}$ durante a identificação de taxonomias e generalização de regras, provê funcionalidades para analisar as Regras de Associação generalizadas. Além disso, neste relatório também é definida uma sintaxe padrão para representar conjuntos de dados transacionais.

Este relatório está organizado da seguinte maneira: na Seção 2 é apresentada a técnica de mineração de Regras de Associação. Na Seção 3 é descrito o algoritmo $\mathcal{IGART}$. A implementação do algoritmo realizada neste trabalho é descrita na Seção 4. Na Seção 5 é descrito o módulo computacional *RulEE-GAR*. Na Seção 6 são apresentadas algumas considerações finais sobre este relatório. Por fim, no Apêndice A é apresentada a sintaxe padrão definida para representar conjuntos de dados transacionais.

2 Regras de Associação

Na classificação usualmente empregada em Mineração de Dados, a técnica de Regras de Associação pode ser categorizada como uma *Atividade de Mineração de Dados Descritiva* (Weiss & Indurkhya, 1998; Rezende et al., 2003).

Uma Regra de Associação caracteriza o quanto a presença de um conjunto de itens nos registros de uma Base de Dados implica na presença de algum outro conjunto distinto de itens nos mesmos registros (Agrawal & Srikant, 1994). Desse modo, o objetivo das Regras de Associação é encontrar tendências que possam ser usadas para entender e explorar padrões de comportamento dos dados. Por exemplo, observando os dados de vendas de um supermercado, sabe-se que 80% dos clientes que compram o produto Q também adquirem, na mesma ocasião, o produto W. Nessa regra 80% corresponde à sua confiabilidade.

O formato de uma Regra de Associação pode ser representado como uma implicação $LHS \Rightarrow RHS$, em que LHS e RHS são, respectivamente, o lado esquerdo (*Left Hand Side*) e o lado direito (*Right Hand Side*) da regra, definidos por conjuntos disjuntos de itens. As Regras de Associação podem ser definidas como descrito a seguir (Agrawal & Srikant, 1994):

Seja D uma Base de Dados composta por um conjunto de itens $A = \{a_1, ..., a_m\}$ ordenados lexicograficamente e por um conjunto de transações $T = \{t_1, ..., t_n\}$, na qual cada transação $t_i \in T$ é composta por um conjunto de itens tal que $t_i \subseteq A$.

A Regra de Associação é uma implicação na forma $LHS \Rightarrow RHS$, em que $LHS \subset A$, $RHS \subset A$ e $LHS \cap RHS = \varnothing$. A regra $LHS \Rightarrow RHS$ ocorre no conjunto de transações T com confiança $conf$ se em $conf\%$ das transações de T em que ocorre LHS ocorre também RHS. A regra $LHS \Rightarrow RHS$ tem suporte sup se em $sup\%$ das transações em D ocorre $LHS \cup RHS$.

O valor do suporte mede a força da associação entre LHS e RHS, e não relaciona possíveis dependências de RHS com LHS. Por outro lado, a confiança mede a força da implicação lógica descrita pela regra. Usualmente, valores de suporte e confiança mínimos são definidos pelo usuário antes da mineração das Regras de Associação: a definição de altos valores gera apenas regras triviais, já a definição de baixos valores gera um grande volume de conhecimento no formato de regras, dificultando a análise do usuário. Uma maneira de superar as dificuldades na análise dessas regras é utilizar-se de taxonomias durante o seu Pós-processamento. Na próxima seção é descrito o algoritmo $\mathcal{IGART}$, que utiliza taxonomias para generalizar regras, facilitando, dessa maneira, a análise das Regras de Associação obtidas.

3 O Algoritmo $\mathcal{IGART}$

O fato das Regras de Associação permitirem identificar associações entre itens e conjuntos de itens de uma Base de Dados faz com que os algoritmos produzam grandes quantidades de regras, muitas das quais não são interessantes para o usuário (Liu et al., 2000).

Devido a essa grande quantidade de regras, a análise e a compreensão do conhecimento torna-se difícil para o usuário. A aplicação de taxonomias em Regras de Associação pode ser utilizada para reduzir o volume de regras extraídas e por conseqüência, facilitar a análise e compreensão do conhecimento.

As taxonomias refletem uma caracterização coletiva ou individual de como os itens podem ser hierarquicamente classificados (Adamo, 2001). Na Figura 1 é apresentado um exemplo de uma taxonomia. Nesse exemplo pode-se verificar que: camiseta é uma roupa leve, bermuda é uma roupa leve, roupa leve é um tipo de roupa, sandália é um tipo de calçado, etc.

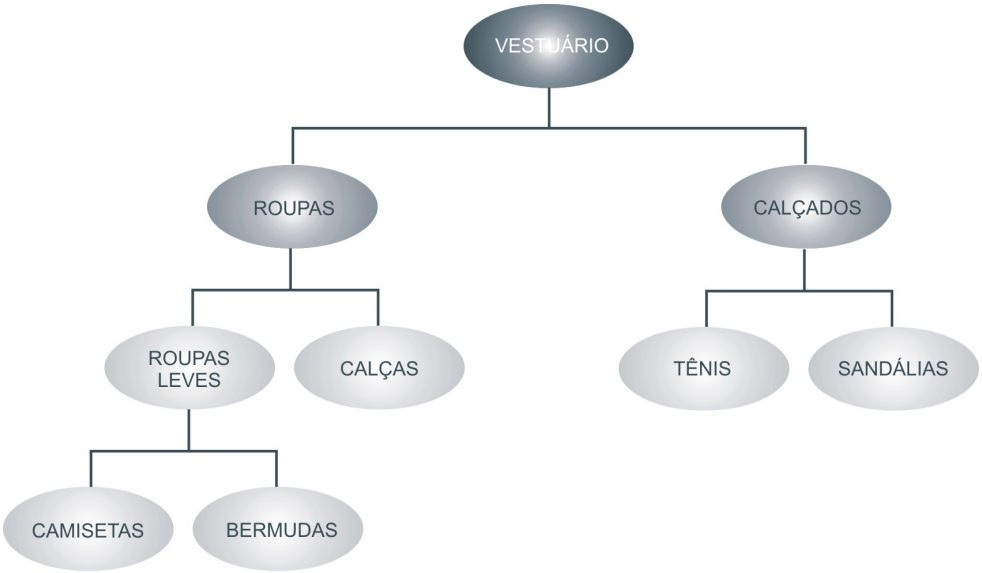

Figura 1: Exemplo de uma taxonomia para vestuário

É possível verificar que as Regras de Associação podem ser generalizadas utilizando taxonomias. Esse fato é ilustrado nas Figuras 2 e 3.

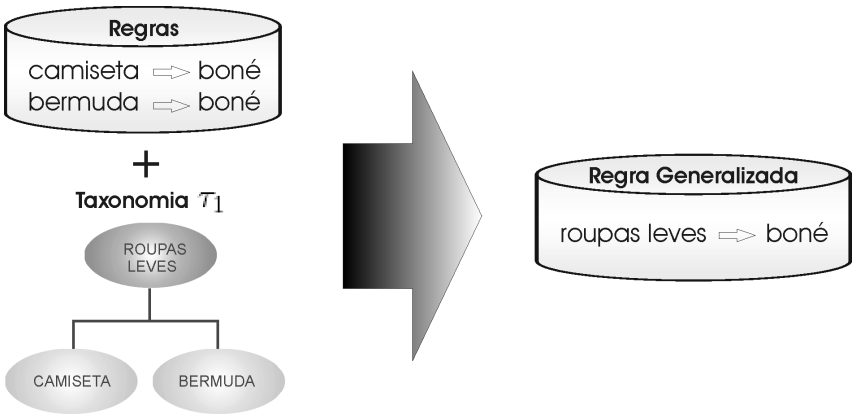

Figura 2: Generalização de Regras de Associação usando uma taxonomia

Na Figura 2 é ilustrada a generalização das regras

$$\mathsf{camiseta} \Rightarrow \mathsf{boné} \text{ e}$$
$$\mathsf{bermuda} \Rightarrow \mathsf{boné},$$

utilizando a taxonomia roupas leves (Taxonomia τ_1 na Figura 2) para generalizar os itens camiseta e bermuda e gerar a Regra de Associação generalizada

$$\text{roupas leves} \Rightarrow \text{boné}.$$

Durante a generalização, há a substituição dos itens camiseta e bermuda pelo item roupas leves gerando duas regras roupas leves ⇒ boné. Em seguida, uma das regras é removida permanecendo apenas uma regra generalizada roupas leves ⇒ boné.

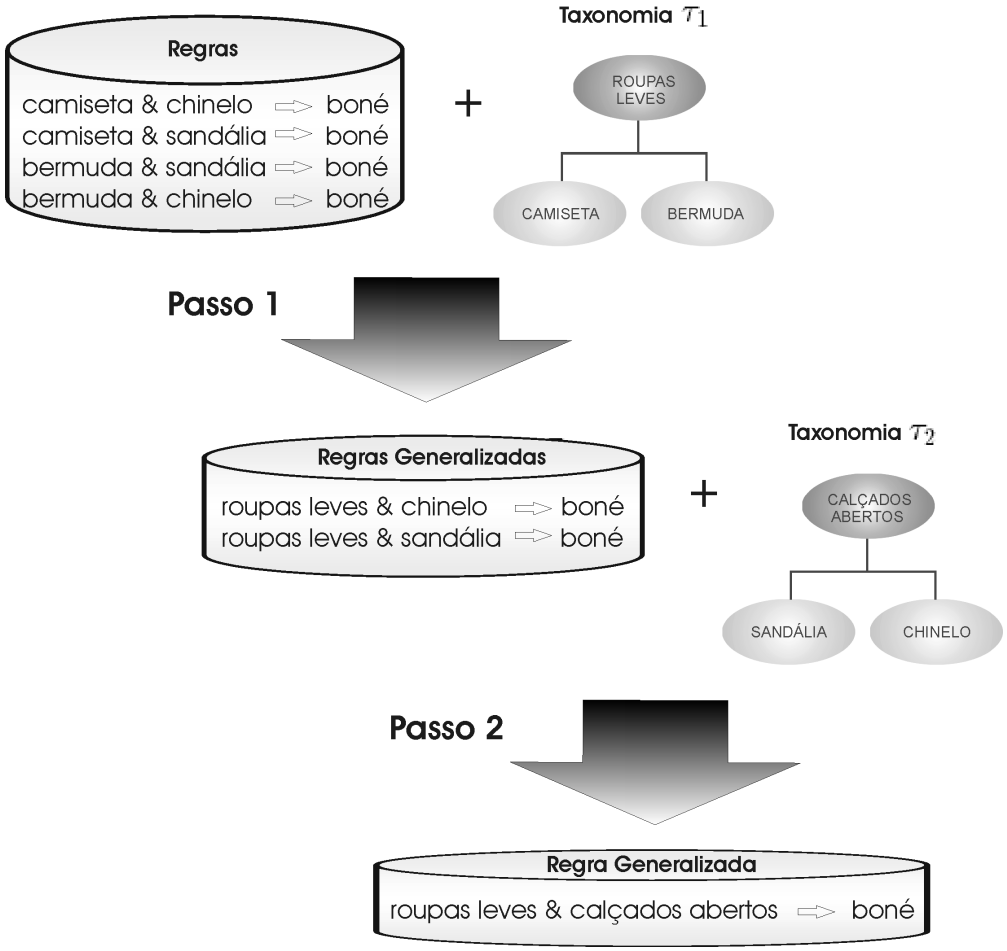

Figura 3: Generalização de Regras de Associação usando duas taxonomias

De modo semelhante, na Figura 3 é ilustrada a generalização de quatro Regras de Associação aplicando duas taxonomias. Inicialmente, os itens camiseta e bermuda das regras

$$\text{camiseta \& chinelo} \Rightarrow \text{boné},$$
$$\text{camiseta \& sandália} \Rightarrow \text{boné},$$
$$\text{bermuda \& sandália} \Rightarrow \text{boné e}$$
$$\text{bermuda \& chinelo} \Rightarrow \text{boné},$$

são substituídos pelo item roupas leves (que representa uma generalização) gerando duas regras roupas leves & chinelo $\Rightarrow$ boné e duas regras roupas leves & sandália $\Rightarrow$ boné. Em seguida, as regras repetidas são removidas permanecendo apenas as regras

$$\text{roupas leves \& chinelo} \Rightarrow \text{boné e}$$
$$\text{roupas leves \& sandália} \Rightarrow \text{boné.}$$

As duas regras resultantes do passo 1 (Figura 3), são novamente generalizadas, sendo os itens chinelo e sandália substituídos pelo item calçados abertos (que representa outra generalização), gerando duas regras roupas leves & calçados abertos $\Rightarrow$ boné. Em seguida, as regras repetidas são removidas permanecendo apenas a Regra de Associação generalizada

$$\text{roupas leves \& calçados abertos} \Rightarrow \text{boné.}$$

O comportamento ilustrado pelas Figuras 2 e 3 foi identificado em diversos conjuntos de Regras de Associação analisados. A partir dessa análise foi proposto um processo para generalizar Regras de Associação que é apresentado na Figura 4.

Neste trabalho, o processo proposto generaliza apenas um dos lados (LHS ou RHS) das Regras de Associação. Inicialmente as regras (representadas na sintaxe padrão definida por Melanda & Rezende (2003)) são agrupadas em subconjuntos que apresentam antecedente ou conseqüente semelhante. Se o processo for utilizado para generalizar o lado esquerdo das regras (LHS), os subconjuntos são gerados utilizando conseqüentes (RHS) semelhantes e se o processo for utilizado para generalizar o lado direito das regras (RHS), os subconjuntos são gerados utilizando antecedentes (LHS) semelhantes. Na Figura 4 é ilustrado o processo de generalização do lado esquerdo das regras, por conseqüência, os subconjuntos são agrupados utilizando semelhanças no lado direito das regras. Em seguida são utilizadas as taxonomias para generalizar cada subconjunto, e, ao final da generalização, os itens de cada regra são ordenados lexicograficamente e a regra é armazenada em um conjunto de Regras de Associação generalizadas.

Ao final do processo, deve-se calcular a Tabela de Contingência para cada Regra de Associação que foi generalizada, uma vez que a sintaxe padrão proposta por Melanda & Rezende (2003) inclue a regra acompanhada de sua Tabela de Contingência. A Tabela de Contingência de uma regra representa a sua cobertura com relação a Base de Dados que foi utilizada na mineração da regra. Além disso, os valores da tabela podem ser utilizados para calcular diversas medidas de avaliação de regras, entre as quais pode-se citar o *framework* de medidas de avaliação proposto por Lavrač et al. (1999). No processo

de generalização proposto, o cálculo da Tabela de Contingência é realizado utilizando a Base de Dados que foi empregada na mineração das Regras de Associação ao qual se deseja generalizar. Com o cálculo da Tabela de Contingência encerra-se o processo de generalização.

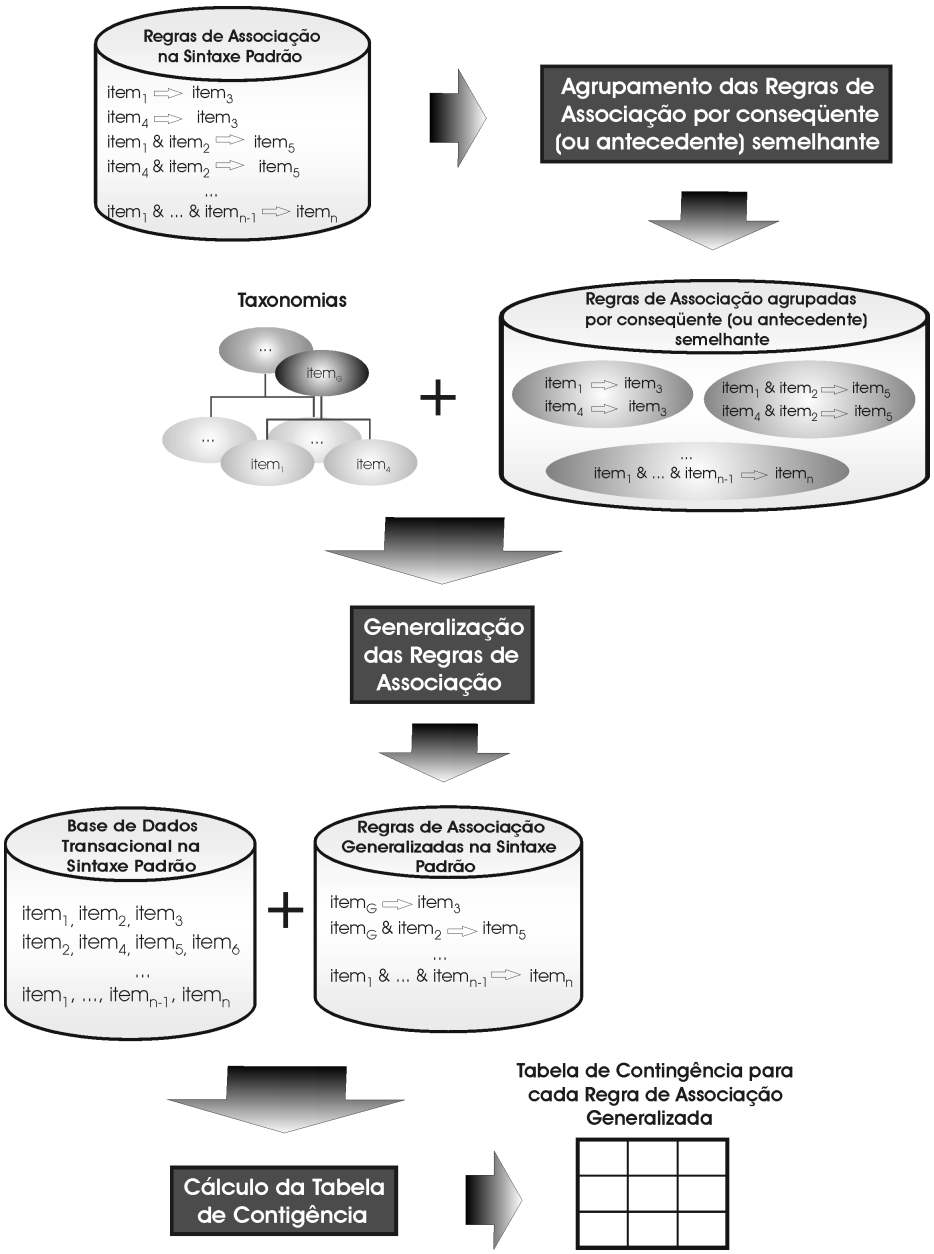

Figura 4: Processo proposto para generalização de Regras de Associação

O algoritmo $\mathcal{IGART}$ (*Interactive Generalization of Association Rules using Taxonomies* – Generalização Interativa de Regras de Associação usando Taxonomias), responsável pelo processo de generalização de Regras de Associação proposto, é ilustrado no Algoritmo 1. O algoritmo recebe como entrada um conjunto de Regras de Associação

R, um conjunto de Taxonomias $\mathcal{T}$, uma Base de Dados D e a definição do *lado* da regra a ser generalizado – lado esquerdo (antecedente da regra) ou lado direito (conseqüente da regra). A notação $\overline{lado}$ define o lado das regras que não será generalizado.

Algoritmo 1 $\mathcal{IGART}$

Require: Um conjunto de Regras de Associação R, um conjunto de Taxonomias $\mathcal{T}$, uma Base de Dados D e a definição do *lado* da regra a ser generalizado – lado esquerdo (antecedente da regra) ou lado direito (conseqüente da regra).

1: $Rg := \varnothing$;
2: $\widehat{E}$:= gera-subconjuntos(R, $\overline{lado}$);
3: **for all** subconjunto $E \subseteq \widehat{E}$ **do**
4: generaliza-regras(E, $\mathcal{T}$, *lado*);
5: ordena-lexicograficamente(E, *lado*);
6: $Rg := Rg \cup E$;
7: **end for**
8: **for all** regra $r \in Rg$ **do**
9: **if** r é uma regra generalizada **then**
10: calcula-TC(r, $\mathcal{T}$, D);
11: **end if**
12: **end for**
13: **return** Rg;

Inicialmente o algoritmo inicializa com "$\varnothing$" (vazio) o conjunto Rg que irá armazenar as Regras de Associação generalizadas. Em seguida são gerados os subconjuntos de regras que possuem antecedente ou conseqüente semelhante. Se o algoritmo for utilizado para generalizar o lado esquerdo das regras, os subconjuntos são gerados utilizando conseqüentes semelhantes e se o algoritmo for utilizado para generalizar o lado direito das regras, os subconjuntos são gerados utilizando antecedentes semelhantes. A geração dos subconjuntos é realizada com a função gera-subconjuntos (linha 2 do Algoritmo 1). Nessa função, o parâmetro $\overline{lado}$ indica que os subconjuntos são gerados utilizando o lado das regras que não será generalizado. A função gera-subconjuntos é descrita na Seção 3.1.

O próximo passo do algoritmo (linhas 3 a 7 do Algoritmo 1) consiste na generalização das Regras de Associação contidas em cada um dos subconjuntos de regras $E \subseteq \widehat{E}$, na ordenação lexicográfica dos itens de cada regra pertencente aos subconjuntos de regras e no armazenamento das regras no conjunto Rg. A generalização é realizada utilizando a função generaliza-regras, descrita na Seção 3.2. A ordenação lexicográfica dos itens das regras é realizada utilizando a função ordena-lexicograficamente, que é descrita na Seção 3.6.

Após a generalização é calculada a Tabela de Contingência para cada Regra de Associação generalizada contida no conjunto Rg (linhas 8 a 12 do Algoritmo 1). O cálculo da Tabela

de Contingência é realizado utilizando a função calcula-TC, descrita na Seção 3.7. Por fim, o algoritmo (na linha 13) retorna o conjunto de Regras de Associação generalizadas Rg. Nas próximas seções são descritas as funções utilizadas pelo algoritmo $\mathcal{IGART}$.

3.1 Função gera-subconjuntos

Para gerar subconjuntos contendo Regras de Associação com antecedentes ou conseqüentes semelhantes, o algoritmo $\mathcal{IGART}$ utiliza a função gera-subconjuntos apresentada no Algoritmo 2. A função recebe como entrada um conjunto de Regras de Associação R e a definição do *lado* da regra a ser utilizado para construir os subconjuntos – lado esquerdo (antecedente da regra) ou lado direito (conseqüente da regra).

Algoritmo 2 Função gera-subconjuntos

Require: Um conjunto de Regras de Associação R e a definição do *lado* da regra a ser utilizado para construir os subconjuntos – lado esquerdo (antecedente da regra) ou lado direito (conseqüente da regra).

1: $\widehat{Rt} := \varnothing$;
2: **for all** regra $r \in R$ **do**
3: $\quad Rt := \varnothing$;
4: $\quad rt := r$;
5: $\quad$ **for all** regra $r \in R$ **do**
6: $\quad\quad$ **if** $rt[lado] = r[lado]$ **then**
7: $\quad\quad\quad Rt := Rt + r$;
8: $\quad\quad\quad$ remove r de R;
9: $\quad\quad$ **end if**
10: $\quad$ **end for**
11: $\quad \widehat{Rt} := \widehat{Rt} + Rt$;
12: **end for**
13: **return** $\widehat{Rt}$; //retorna um conjunto contendo subconjuntos de Regras de Associação.

Inicialmente o algoritmo inicializa com "$\varnothing$" (vazio) o conjunto $\widehat{Rt}$ que irá armazenar os subconjuntos de Regras de Associação (linha 1 do Algoritmo 2). Já na linha 3 é inicializado com "$\varnothing$" (vazio) a variável Rt, que irá armazenar temporariamente cada subconjunto de regras. Em seguida, uma das regras $r \in R$ é armazenada na variável temporária rt. A regra armazenada em rt será utilizada para construir um subconjunto que contenha regras cujo lado esquerdo ou direito é igual ao da regra em rt. Cada subconjunto de Regras de Associação é construído no laço (*loop*) representado pelas linhas 5 a 10 do Algoritmo 2. Na linha 6 é verificado para cada regra $r \in R$, se a mesma apresenta o lado, definido para construir os subconjuntos, igual ao da regra armazenada em rt. Se o lado de ambas as regras são iguais, a regra r é armazenada em Rt e removida de R (linhas 7 e 8). Após realizar a verificação para todas as regras $r \in R$ tem-se um

subconjunto Rt que será armazenado no conjunto $\widehat{Rt}$, que armazena os subconjuntos de Regras de Associação (linha 11). Se ainda houver regras em R, o processo descrito é realizado novamente, senão o conjunto contendo subconjuntos de Regras de Associação é retornado pela função (linha 13 do Algoritmo 2).

3.2 Função generaliza-regras

A função generaliza-regras é apresentada no Algoritmo 3. Essa função recebe como entrada um conjunto de Regras de Associação R (com antecedentes ou conseqüentes semelhantes – dependendo do lado a ser generalizado), um conjunto de Taxonomias $\mathcal{T}$ e a definição do *lado* da regra a ser generalizado – lado esquerdo (antecedente da regra) ou lado direito (conseqüente da regra).

Algoritmo 3 Função generaliza-regras

Require: Um conjunto de Regras de Associação R (com antecedentes ou conseqüentes semelhantes – dependendo do lado a ser generalizado), um conjunto de Taxonomias $\mathcal{T}$ e a definição do *lado* da regra a ser generalizado – lado esquerdo (antecedente da regra) ou lado direito (conseqüente da regra).

1: **for all** taxonomia $\tau \in \mathcal{T}$ **do**
2: **for all** regra $r \in R$ **do**
3: generaliza-uma-regra(r, τ, *lado*);
4: **end for**
5: $Rgt := \varnothing$;
6: $\widehat{Et}$:= gera-subconjuntos(R, *lado*);
7: **for all** subconjunto $Et \subseteq \widehat{Et}$ **do**
8: **if** número-regras(Et) = número-itens-especialização-máxima(τ) **then**
9: remove-regras-repetidas(Et);
10: **else**
11: desfaz-última-generalização(Et, τ, *lado*);
12: **end if**
13: $Rgt := Rgt \cup Et$;
14: **end for**
15: $R := Rgt$;
16: **end for**

Na função generaliza-regras, a generalização das Regras de Associação é realizada considerando cada uma das taxonomias $\tau \in \mathcal{T}$. A generalização de cada regra $r \in R$ para uma determinada taxonomia $\tau \in \mathcal{T}$ (linhas 2 a 4 do Algoritmo 3) é realizada utilizando a função generaliza-uma-regra, descrita na Seção 3.3. Em seguida, a função generaliza-regras inicializa com "$\varnothing$" (vazio) o conjunto Rgt, que irá armazenar temporariamente as Regras de Associação generalizadas (linha 5), e gera subconjuntos de regras que apresentam o *lado* da generalização (antecedente ou conseqüente) semelhantes (linha 6). O lado

da regra que não está sendo generalizado ($\overline{lado}$) já apresenta itens semelhantes, devido a execução da função gera-subconjuntos no Algoritmo 1.

Um dos motivos para gerar os subconjuntos com regras generalizadas semelhantes é facilitar a remoção das regras que são repetidas. Outro motivo é a verificação da igualdade do número de regras em cada subconjunto $Et \subseteq \widehat{Et}$ e do número de itens que são especializações máximas na tanoxomia. Essa verificação valida uma generalização e evita que a mesma induza um conhecimento que não existe. Por exemplo, na Figura 2 na página 4 é possível verificar que o número de itens que são especializações máximas na taxonomia (2 itens) é igual ao número de regras generalizadas (2 regras considerando as repetidas). Assim, a generalização é válida e as Regras de Associação generalizadas repetidas são removidas.

Uma generalização não é válida se o número de regras em um subconjunto for menor do que o número de itens que são especializações máximas na taxonomia. Por exemplo, se na Figura 2 na página 4 existisse apenas a regra camiseta $\Rightarrow$ boné para ser generalizada, e caso houvesse a generalização dessa regra, a mesma seria inválida pois haveria uma regra generalizada e dois itens que são considerados especializações máximas na taxonomia. Assim, a regra generalizada roupas leves $\Rightarrow$ boné estaria representando (pela presença do item roupas leves) a generalização das regras camiseta $\Rightarrow$ boné e bermuda $\Rightarrow$ boné, quando na verdade foi realizada apenas a generalização da regra camiseta $\Rightarrow$ boné, ocorrendo, dessa maneira, a indução de um conhecimento que não existe. Cabe ressaltar que em nenhum momento o número de regras armazenadas em um subconjunto poderá ser maior do que o número de itens que são especializações máximas em uma taxonomia, uma vez que isso implicaria na presença de Regras de Associação repetidas no subconjunto.

Na linha 8 do Algoritmo 3 é realizada a comparação do número de regras em cada subconjunto $Et \subseteq \widehat{Et}$ e do número de itens que são especializações máximas na taxonomia. Se o número de regras e o número de itens são iguais, as Regras de Associação que foram generalizadas são consideradas válidas e as regras repetidas são removidas (linha 9 do Algoritmo 3) utilizando a função remove-regras-repetidas, descrita na Seção 3.5. Porém, se o número de regras e o número de itens não são iguais, as Regras de Associação que foram generalizadas são consideradas inválidas e a generalização é desfeita (linha 11 do Algoritmo 3) utilizando a função desfaz-última-generalização, descrita na Seção 3.4. Em seguida, as Regras de Associação que foram ou não generalizadas são armazenadas temporariamente no conjunto de regras Rgt. Por fim, o conjunto de regras Rgt é atribuído ao conjunto R (linha 15 do Algoritmo 3) para que o mesmo possa ser generalizado novamente utilizando uma outra taxonomia $\tau \in \mathcal{T}$.

3.3 Função generaliza-uma-regra

A função generaliza-uma-regra é apresentada no Algoritmo 4. Essa função recebe como entrada uma Regra de Associação r, uma Taxonomia $\mathcal{T}$ e a definição do *lado* da regra a ser generalizado – lado esquerdo (antecedente da regra) ou lado direito (conseqüente da regra).

Algoritmo 4 Função generaliza-uma-regra

Require: Uma Regra de Associação r, uma Taxonomia $\mathcal{T}$ e a definição do *lado* da regra a ser generalizado – lado esquerdo (antecedente da regra) ou lado direito (conseqüente da regra).

1: **if** *lado* = *esquerdo* **then**
2: **if** apenas um item a do antecedente da regra r é generalizado por $\mathcal{T}$ **then**
3: r.última-generalização := a;
4: a := $\mathcal{T}$.item-generalização-máxima;
5: **end if**
6: **else**
7: **if** apenas um item a do conseqüente da regra r é generalizado por $\mathcal{T}$ **then**
8: r.última-generalização := a;
9: a := $\mathcal{T}$.item-generalização-máxima;
10: **end if**
11: **end if**

Inicialmente a função generaliza-uma-regra verifica qual é o lado da regra a ser generalizado. Se o lado da regra a ser generalizado for o esquerdo (antecedente da regra), as linhas 2 a 5 do Algoritmo 4 são executadas. Na linha 2 é verificado se apenas um item a da regra r é generalizado pela taxonomia $\mathcal{T}$, ou seja, se apenas um item a no lado da generalização é igual a um dos itens que são especializações máximas na taxonomia. Se a verificação for verdadeira, o item da regra é armazenado em r.última-generalização (linha 3) para ser usado futuramente caso haja a necessidade de desfazer a generalização. Em seguida, o item da regra é substituído pelo item de generalização máxima da taxonomia (linha 4 do Algoritmo 4).

Caso o lado da regra a ser generalizado seja o lado direito (conseqüente da regra) são executadas as linhas 7 a 10 do Algoritmo 4. Nessas linhas são executadas as mesmas tarefas das linhas 2 a 5, porém, para o lado direito das regras.

É importante ressaltar que as verificações ocorridas nas linhas 2 e 7 são de grande importância para a generalização das Regras de Associação, pois as mesmas evitam que uma regra generalizada apresente, entre seus itens, duas ou mais generalizações iguais.

3.4 Função desfaz-última-generalização

No Algoritmo 5 é apresentada a função que desfaz a última generalização de um conjunto de regras R quando essa generalização é considerada inválida. A função desfaz-última-generalização recebe como entrada um conjunto de Regras de Associação R, uma taxonomia $\mathcal{T}$ e a definição do *lado* da regra a ser generalizado – lado esquerdo (antecedente da regra) ou lado direito (conseqüente da regra).

Algoritmo 5 Função desfaz-última-generalização

Require: Um conjunto de Regras de Associação R, uma taxonomia $\mathcal{T}$ e a definição do *lado* da regra a ser generalizado – lado esquerdo (antecedente da regra) ou lado direito (conseqüente da regra).

```
1: if lado = esquerdo then
2:   for all regra r ∈ R do
3:     if há algum item a no antecedente da regra r que foi generalizado por T then
4:       a := r.última-generalização;
5:     end if
6:   end for
7: else
8:   for all regra r ∈ R do
9:     if há algum item a no conseqüente da regra r que foi generalizado por T then
10:      a := r.última-generalização;
11:    end if
12:  end for
13: end if
```

A função desfaz-última-generalização é executada quando um generalização é considerada inválida, ou seja, quando o número de regras em um subconjunto for menor do que o número de itens que são especializações máximas na taxonomia $\mathcal{T}$. A sua execução evita que o processo de generalização de Regras de Associação induza um conhecimento que não exista, como foi exemplificado na Seção 3.2.

3.5 Função remove-regras-repetidas

A função remove-regras-repetidas, apresentada no Algoritmo 6, remove as Regras de Associação repetidas que estão contida em um conjunto de regras. A função recebe como entrada um conjunto de Regras de Associação R contendo apenas regras repetidas.

Algoritmo 6 Função remove-regras-repetidas

Require: Um conjunto de Regras de Associação R.

1: $rt := r$;
2: $R := \varnothing$;
3: $R := R + rt$;

Na linha 1 do Algoritmo 6 é armazenada na variável rt uma regra qualquer $r \in R$. Como todas as regras armazenadas em R são iguais, na linha 2 o conjunto de regras é inicializado com "$\varnothing$" (vazio) e na linha 3, a regra em rt é armazenada novamente no conjunto R. Desse modo, tem-se a remoção das regras repetidas armazenadas em um conjunto.

3.6 Função ordena-lexicograficamente

Para ordenar lexicograficamente os itens de uma Regra de Associação foi realizada uma adaptação no algoritmo de ordenação *Bubble Sort*. O algoritmo *Bubble Sort* adaptado, que é apresentado no Algoritmo 7, foi denominado de função ordena-lexicograficamente e recebe como entrada um conjunto de Regras de Associação R e a definição do *lado* da regra cujos itens serão ordenados – lado esquerdo (antecedente da regra) ou lado direito (conseqüente da regra).

Algoritmo 7 Função ordena-lexicograficamente

Require: Um conjunto de Regras de Associação R e a definição do *lado* da regra cujos itens serão ordenados – lado esquerdo (antecedente da regra) ou lado direito (conseqüente da regra).

1: **for all** regra $r \in R$ **do**
2: **for** $(i := 0;\ i < (r[lado].size - 1);\ i++)$ **do**
3: **for** $(j := 0;\ j < (r[lado].size - 1 - i);\ j++)$ **do**
4: **if** $r[lado].item[j+1] < r[lado].item[j]$ **then**
5: $at := r[lado].item[j]$;
6: $r[lado].item[j] := r[lado].item[j+1]$;
7: $r[lado].item[j+1] := at$;
8: **end if**
9: **end for**
10: **end for**
11: **end for**

Nas linhas 2 a 10 da função ordena-lexicograficamente, cada item (no lado da regra que deve ser ordenado) é comparado com o seu sucessor para verificar a ordenação lexicografica. Nas linhas 4 a 8, os itens são comparados com os seus sucessores e caso não estejam ordenados lexicograficamente, o item e seu sucessor tem suas posições alteradas na regra.

O laço (*loop*), da linha 1 a 11 do Algoritmo 7, garante que o processo de ordenação seja realizado para cada regra contida no conjunto de Regras de Associação R.

3.7 Função calcula-TC

A Tabela de Contingência para uma Regra de Associação r, no formato $LHS \Rightarrow RHS$, representa a sua cobertura com relação a Base de Dados que foi utilizada para minerar essa regra. Na Tabela 1, LHS denota o antecedente da regra e $\overline{LHS}$ o complemento de LHS. O mesmo conceito se aplica para o conseqüente RHS e $\overline{RHS}$. $n(LHS)$ denota o número de transações no qual LHS é verdadeiro e $n(\overline{LHS})$ denota o número de transações na qual LHS é falso. $n(LHS\ RHS)$ denota o número de transações no qual LHS e RHS são verdadeiros, $n(\overline{LHS}\ RHS)$ denota o número de transações no qual $\overline{LHS}$ é falso e RHS é verdadeiro, de maneira similar para os demais elementos e N denota o número total de transações.

Tabela 1: Tabela de Contingência. Fonte:(Lavrač et al., 1999)

	RHS	$\overline{RHS}$	
LHS	$n(LHS\ RHS)$	$n(LHS\ \overline{RHS})$	$n(LHS)$
$\overline{LHS}$	$n(\overline{LHS}\ RHS)$	$n(\overline{LHS}\ \overline{RHS})$	$n(\overline{LHS})$
	$n(RHS)$	$n(\overline{RHS})$	N

O cálculo da Tabela de Contingência para as Regras de Associação generalizadas (no processo proposto) é realizado pela função calcula-TC apresentada no Algoritmo 8. A função recebe como entrada uma Regra de Associação r, um conjunto de Taxonomias $\mathcal{T}$ e uma Base de Dados D.

Os valores da Tabela de Contingência são inicializados com "0" (zero). Na linha 2 do Algoritmo 8, os itens que foram generalizados na Regra de Associação são recuperados para que o cálculo dos valores da Tabela de Contingência possa ser realizado. A recuperação desses itens é necessária para a realização da comparação entre os itens da regra e os da Base de Dados, uma vez que os itens da base não estão generalizados.

No Algoritmo 8, o laço (*loop*) representado pelas linhas 3 a 28 permite verificar a cobertura de uma regra r para todas as transações pertencentes a uma Base de Dados D. Nas linhas 5 a 11 são verificadas a cobertura do antecedente (LHS) de uma Regra de Associação r e calculados os valores de $n(LHS)$ (se o antecedente da regra cobre a transação) e $n(\overline{LHS})$ (se o antecedente da regra não cobre a transação) para a Tabela de Contingência. Já nas linhas 12 a 18 são verificadas a cobertura do conseqüente (RHS) de uma Regra

de Associação r e calculados os valores de $n(RHS)$ (se o conseqüente da regra cobre a transação) e $n(\overline{RHS})$ (se o conseqüente da regra não cobre a transação). Por fim, nas linhas 19 a 27 do Algoritmo 8 são calculados os valores $n(LHS\ RHS)$, $n(LHS\ \overline{RHS})$, $n(\overline{LHS}\ RHS)$ e $n(\overline{LHS}\ \overline{RHS})$. O valor do número de transações N é calculado na linha 4 do Algoritmo 8.

Algoritmo 8 Função calcula-TC

Require: Uma Regra de Associação r, um conjunto de Taxonomias $\mathcal{T}$ e uma Base de Dados D.

1: Inicializa todos os valores da Tabela de Contingência com 0;
2: Recupera os itens de r que foram generalizados utilizando $\mathcal{T}$;
3: **for all** transação da Base de Dados D **do**
4: incrementa o número de transações N;
5: **if** todos os itens do antecedente da regra r são cobertos pela transação **then**
6: *antecedente* := **true**;
7: incrementa o valor de $n(LHS)$ na Tabela de Contingência;
8: **else**
9: *antecedente* := **false**;
10: incrementa o valor de $n(\overline{LHS})$ na Tabela de Contingência;
11: **end if**
12: **if** todos os itens do conseqüente da regra r são cobertos pela transação **then**
13: *consequente* := **true**;
14: incrementa o valor de $n(RHS)$ na Tabela de Contingência;
15: **else**
16: *consequente* := **false**;
17: incrementa o valor de $n(\overline{RHS})$ na Tabela de Contingência;
18: **end if**
19: **if** (*antecedente* = **true**) **and** (*consequente* = **true**) **then**
20: incrementa o valor de $n(LHS\ RHS)$ na Tabela de Contingência;
21: **else if** (*antecedente* = **true**) **and** (*consequente* = **false**) **then**
22: incrementa o valor de $n(LHS\ \overline{RHS})$ na Tabela de Contingência;
23: **else if** (*antecedente* = **false**) **and** (*consequente* = **true**) **then**
24: incrementa o valor de $n(\overline{LHS}\ RHS)$ na Tabela de Contingência;
25: **else**
26: incrementa o valor de $n(\overline{LHS}\ \overline{RHS})$ na Tabela de Contingência;
27: **end if**
28: **end for**

3.8 Exemplo de Funcionamento do Algoritmo $\mathcal{IGART}$

No Exemplo 1, páginas 17 a 22, é ilustrado o funcionamento do algoritmo $\mathcal{IGART}$ aplicado a um pequeno conjunto de Regras de Associação.

Exemplo 1 *Seja D uma Base de Dados que contém um conjunto de itens* $A = \{$bermuda, boné, camiseta, chinelo, sandália, short, tênis$\}$ *e um conjunto de transações* $T = \{1, 2, 3, 4, 5, 6\}$*, no qual a relação de itens comprados por cada transação* t_i *é apresentada na Tabela 2.*

Tabela 2: Exemplo 1 – Relação de itens comprados por transação

Transações	**Itens comprados**
1	camiseta, chinelo, boné, tênis
2	camiseta, sandália, boné, tênis
3	bermuda, chinelo, boné, tênis
4	bermuda, sandália, boné, tênis
5	sandália, chinelo, short, tênis
6	sandália, camiseta, short, tênis

A Base de Dados D foi submetida ao algoritmo de mineração de Regras de Associação Apriori[1] *com o valor do suporte mínimo (***sup-min***) igual a* 10% *e da confiança mínima (***conf-min***) igual a* 50%*. O algoritmo gerou 101 Regras de Associação, das quais foram selecionadas 8 regras para compor o conjunto R, apresentado na Tabela 3.*

Tabela 3: Exemplo 1 – Conjunto de Regras de Associação R

Regras	$n(LHS\ RHS)$	$n(LHS\ \overline{RHS})$	$n(\overline{LHS}\ \overline{RHS})$	$n(\overline{LHS}\ RHS)$
bermuda ⇒ tênis	2	0	0	4
camiseta ⇒ tênis	3	0	0	3
bermuda & chinelo ⇒ boné	1	0	2	3
bermuda & sandália ⇒ boné	1	0	2	3
camiseta & sandália ⇒ boné	1	1	1	3
camiseta & chinelo ⇒ boné	1	0	2	3
chinelo & sandália ⇒ short	1	0	4	1
camiseta & sandália ⇒ short	1	1	3	1

[1]Uma implementação gratuita do algoritmo *Apriori* pode ser obtida no site `http://fuzzy.cs.uni-magdeburg.de/~borgelt/software.html`

Na tabela são apresentados apenas o valores de $n(LHS\ RHS)$, $n(LHS\ \overline{RHS})$, $n(\overline{LHS}\ RHS)$ *e* $n(\overline{LHS}\ \overline{RHS})$ *da Tabela de Contingência de cada regra, uma vez que a partir desses valores é possível obter os demais valores da Tabela de Contingência de uma regra.*

Por fim, na Figura 5 é apresentado o conjunto de Taxonomias $\mathcal{T}$ *que será fornecido ao algoritmo* $\mathcal{IGART}$ *para generalizar o conjunto de Regras de Associação R.*

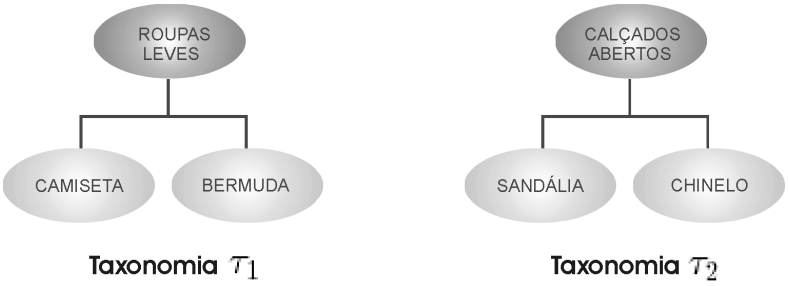

Figura 5: Exemplo 1 – Conjunto de Taxonomias $\mathcal{T}$

Neste exemplo, o algoritmo irá tentar generalizar o antecedente das regras (lado = esquerdo). A seqüência de passos ilustrando o seu funcionamento é descrito a seguir.

Inicialmente a função gera-subconjuntos do algoritmo $\mathcal{IGART}$ *(linha 2 do Algoritmo 1) é executada gerando os subconjuntos de regras que apresentam conseqüentes semelhantes. Os subconjuntos gerados são apresentados na Tabela 4. Em seguida, as linhas 3 a 7 do Algoritmo 1 são executadas na tentativa generalizar cada subconjunto de regras.*

Tabela 4: Exemplo 1 – Subconjuntos de Regras de Associação com conseqüentes semelhantes

Subconjuntos	**Regras**
Subconjunto E_1	bermuda ⇒ tênis camiseta ⇒ tênis
Subconjunto E_2	bermuda & chinelo ⇒ boné bermuda & sandália ⇒ boné camiseta & sandália ⇒ boné camiseta & chinelo ⇒ boné
Subconjunto E_3	chinelo & sandália ⇒ short camiseta & sandália ⇒ short

Tentativa de generalização do subconjunto E_1

Na tentativa de generalizar cada uma das regras armazenadas no subconjunto E_1, a função generaliza-regras (linha 4 do Algoritmo 1) e, por conseqüência, a função generaliza-uma-regra (linha 3 do Algoritmo 3) são executadas. Inicialmente na função generaliza-uma-regra é verificado que ambas as regras bermuda ⇒ tênis *e* camiseta ⇒ tênis *possuem apenas um item no antecedente que pode ser generalizado utilizando a taxonomia* roupas leves *– que generaliza os itens* camiseta *e* bermuda *e é representada pela Taxonomia τ_1 na Figura 5. Assim, é armazenado em r.última-generalização de cada regra o item a ser generalizado, sendo esse item substituído pelo de generalização máxima da Taxonomia τ_1, gerando duas regras* roupas leves ⇒ tênis. *Em seguida, dentro da função generaliza-regras é executada a função gera-subconjuntos (linha 6 do Algoritmo 3) que gera um subconjunto contendo as duas regras* roupas leves ⇒ tênis. *Nas linhas 7 a 14 do Algoritmo 3 é verificado que o número de regras no subconjunto gerado (2 regras) é igual o número de itens que são especializações máximas na Taxonomia τ_1 (2 itens). A verificação valida a generalização e as regras repetidas são removidas (linhas 8 e 9 do Algoritmo 3).*

A regra resultante roupas leves ⇒ tênis *é armazenada temporariamente no conjunto de regras generalizadas Rgt que é atribuído ao conjunto R na tentativa de generalizá-lo, novamente, utilizando a taxonomia* calçados abertos *– que generaliza os itens* sandália *e* chinelo *e é representada pela Taxonomia τ_2 na Figura 5. O processo descrito é executado novamente, sendo que, dessa vez, nenhum item da regra* roupas leves ⇒ tênis *é generalizado pela Taxonomia τ_2.*

O conjunto de regras R contendo apenas a regra roupas leves ⇒ tênis *é retornado ao algoritmo $\mathcal{IGART}$, onde os itens da regra são ordenados lexicograficamente e a regra é armazenada no conjunto Rg. As linhas 3 a 7 do Algoritmo 1 são executadas novamente na tentativa de generalizar o subconjunto de regras E_2.*

Tentativa de generalização do subconjunto E_2

Na tentativa de generalizar cada uma das regras armazenadas no subconjunto E_2, a função generaliza-regras (linha 4 do Algoritmo 1) e, por conseqüência, a função generaliza-uma-regra (linha 3 do Algoritmo 3) são executadas novamente. Inicialmente na função generaliza-uma-regra é verificado que cada uma das quatro regras:

bermuda & chinelo ⇒ boné,
bermuda & sandália ⇒ boné,

camiseta & sandália $\Rightarrow$ boné *e*
camiseta & chinelo $\Rightarrow$ boné

possui apenas um item no antecedente que pode ser generalizado pela Taxonomia τ_1*, ilustrada na Figura 5. Assim, é armazenado em r.última-generalização de cada regra o item a ser generalizado, sendo esse item substituído pelo de generalização máxima da Taxonomia* τ_1*. Desse modo, na função generaliza-uma-regra são geradas duas regras* roupas leves & chinelo $\Rightarrow$ boné *e duas regras* roupas leves & sandália $\Rightarrow$ boné*. Em seguida, é executada a função gera-subconjuntos (linha 6 do Algoritmo 3) que gera dois subconjuntos: um contendo as duas regras* roupas leves & chinelo $\Rightarrow$ boné *e o outro contendo as duas regras* roupas leves & sandália $\Rightarrow$ boné*. Nas linhas 7 a 14 do Algoritmo 3 é verificado que o número de regras em cada subconjunto (2 regras) é igual o número de itens que são especializações máximas na Taxonomia* τ_1 *(2 itens). A verificação valida a generalização e as regras repetidas em cada subconjunto são removidas (linhas 8 e 9 do Algoritmo 3).*

As regras roupas leves & chinelo $\Rightarrow$ boné *e* roupas leves & sandália $\Rightarrow$ boné *são armazenadas temporariamente no conjunto de regras generalizadas Rgt que é atribuído ao conjunto R na tentativa de generalizá-lo utilizando a Taxonomia* τ_2*, ilustrada na Figura 5.*

A função generaliza-uma-regra é novamente executada, permitindo verificar que as regras roupas leves & chinelo $\Rightarrow$ boné *e* roupas leves & sandália $\Rightarrow$ boné *possuem apenas um item no antecedente que pode ser generalizado utilizando a Taxonomia* τ_2*. Assim, é armazenado em r.última-generalização de cada regra o item a ser generalizado, sendo esse item substituído pelo de generalização máxima da Taxonomia* τ_2*. Dessa maneira, são geradas duas regras* roupas leves & calçados abertos $\Rightarrow$ boné *pela função generaliza-uma-regra. Em seguida, dentro da função generaliza-regras é executada a função gera-subconjuntos (linha 6 do Algoritmo 3) que gera um subconjunto contendo as duas regras* roupas leves & calçados abertos $\Rightarrow$ boné*. Nas linhas 7 a 14 do Algoritmo 3 é verificado que o número de regras do subconjunto gerado (2 regras) é igual o número de itens que são especializações máximas na Taxonomia* τ_2 *(2 itens). A verificação valida a generalização e as regras repetidas são removidas (linhas 8 e 9 do Algoritmo 3). As regras são armazenadas temporariamente no conjunto de regras generalizadas Rgt que é atribuído ao conjunto R, o qual não pode mais ser generalizado pois não há mais taxonomias em* $\mathcal{T}$ *para generalizá-lo.*

O conjunto de regras R contendo a regra roupas leves & calçados abertos $\Rightarrow$ boné *é retornado ao algoritmo* $\mathcal{IGART}$*, onde os itens da regra são ordenados lexicograficamente e a regra é armazenada no conjunto Rg. As linhas 3 a 7 do Algoritmo 1 são executadas novamente na tentativa de generalizar o subconjunto de regras* E_3*.*

Tentativa de generalização do subconjunto E_3

A função generaliza-regras (linha 4 do Algoritmo 1) e a função generaliza-uma-regra (linha 3 do Algoritmo 3) são executadas novamente na tentativa de generalizar cada uma das regras armazenadas no subconjunto E_3. Inicialmente na função generaliza-uma-regra é verificado que para a Taxonomia τ_1 (Figura 5), a regra camiseta & sandália ⇒ short *possui apenas um item no antecedente que pode ser generalizado. Assim, é armazenado em r.última-generalização da regra o item a ser generalizado, sendo esse item substituído pelo de generalização máxima da Taxonomia τ_1. Já a regra* chinelo & sandália ⇒ short *não é generalizada, pois a mesma não apresenta itens que possam ser generalizados utilizando a Taxonomia τ_1. Desse modo, na função generaliza-uma-regra são geradas as regras* chinelo & sandália ⇒ short *e* roupas leves & sandália ⇒ short. *Em seguida, dentro da função generaliza-regras é executada a função gera-subconjuntos (linha 6 do Algoritmo 3) que gera dois subconjuntos: um contendo a regra* chinelo & sandália ⇒ short *e o outro contendo a regra* roupas leves & sandália ⇒ short. *Nas linhas 7 a 14 do Algoritmo 3 é verificado que o número de regras em cada subconjunto (1 regra) é diferente do número de itens que são especializações máximas na Taxonomia τ_1 (2 itens). A verificação invalida o processo de generalização e as generalizações são desfeitas (linha 11 do Algoritmo 3). As regras são armazenadas temporariamente no conjunto de regras generalizadas Rgt que é atribuído ao conjunto R na tentativa de generalizá-lo utilizando a Taxonomia τ_2, ilustrada na Figura 5.*

Na função generaliza-uma-regra é novamente verificado que, para a Taxonomia τ_2, a regra camiseta & sandália ⇒ short *possui apenas um item no antecedente que pode ser generalizado. Assim, é armazenado em r.última-generalização da regra o item a ser generalizado, sendo esse item substituído pelo de generalização máxima da Taxonomia τ_2. Já a regra* chinelo & sandália ⇒ short *não pode ser generalizada pois a mesma apresenta dois itens que podem ser generalizados pela mesma taxonomia:* chinelo *e* sandália. *Dessa maneira, na função generaliza-uma-regra são geradas duas regras:* camiseta & calçados abertos ⇒ short *e* chinelo & sandália ⇒ short. *Em seguida, é executada a função gera-subconjuntos (linha 6 do Algoritmo 3) que gera um subconjunto contendo a regra* chinelo & sandália ⇒ short *e outro contendo a regra* camiseta & calçados abertos ⇒ short. *Nas linhas 7 a 14 do Algoritmo 3 é verificado que o número de regras em cada subconjunto (1 regra) é diferente do número de itens que são especializações máximas na Taxonomia τ_2 (2 itens). A verificação invalida o processo de generalização e as generalizações são desfeitas (linha 11 do Algoritmo 3). As regras são armazenadas temporariamente no conjunto de regras generalizadas Rgt que é atribuído ao conjunto R, o qual não pode ser mais generalizado pois não há mais taxonomias em $\mathcal{T}$ para generalizá-lo.*

O conjunto de regras R da função generaliza-regras é retornado ao algoritmo $\mathcal{IGART}$, onde os itens de cada regra são ordenados lexicograficamente e as regras são armazenada no conjunto Rg. Como não há mais subconjuntos para serem generalizados, as linhas 8 a 12 do algoritmo $\mathcal{IGART}$ são executadas para realizar o cálculo da Tabela de Contingência das regras generalizadas armazenadas em Rg. O conjunto de Regras de Associação generalizadas resultante com parte das devidas Tabelas de Contingência é apresentado na Tabela 5.

Tabela 5: Exemplo 1 – Conjunto de Regras de Associação generalizadas *Rg*

Regras	$n(LHS\ RHS)$	$n(LHS\ \overline{RHS})$	$n(\overline{LHS}\ \overline{RHS})$	$n(\overline{LHS}\ RHS)$
roupas leves ⇒ tênis	5	0	0	1
roupas leves & calçados abertos ⇒ boné	4	1	1	0
chinelo & sandália ⇒ short	1	0	4	1
camiseta & sandália ⇒ short	1	1	3	1

*No conjunto de Regras de Associação generalizadas Rg, apresentado na Tabela 5, os valores da Tabela de Contingência das regras que não foram generalizadas (*chinelo & sandália ⇒ short *e* camiseta & sandália ⇒ short*) são semelhantes aos valores, apresentados na Tabela 3, para essas mesmas regras. Já os valores da Tabela de Contingência das regras que foram generalizadas (*roupas leves & calçados abertos ⇒ boné *e* roupas leves ⇒ tênis*) não possuem nenhuma relação com os valores da Tabela de Contingência das Regras de Associação não generalizadas apresentadas na Tabela 3.*

Na próxima seção é descrita a implementação do algoritmo $\mathcal{IGART}$ realizada neste trabalho.

4 Implementação do Algoritmo $\mathcal{IGART}$

O objetivo desta seção é descrever a biblioteca de classes desenvolvida neste trabalho e que implementa o Algoritmo $\mathcal{IGART}$, sendo relatada a funcionalidade dos métodos implementados, os arquivos de entrada e de saída e a interface do usuário com o algoritmo.

4.1 Biblioteca de Classes do Algoritmo $\mathcal{IGART}$

A implementação do algoritmo $\mathcal{IGART}$, realizada neste trabalho, é composta de três classes: **ARtaxonomy**, **ARrule** e **ARigart**. As classe foram implementadas utilizando

a linguagem de programação *Perl* [2] (Till, 1996), que é a linguagem utilizada no desenvolvimento do projeto *Discover* (Baranauskas & Batista, 2000; Rezende et al., 2004), ao qual este trabalho faz parte.

4.1.1 Classe ARtaxonomy

A classe **ARtaxonomy** é responsável pelo armazenamento das taxonomias utilizadas no processo de generalização das Regras de Associação e por prover métodos que permitem acessar as taxonomias. Os métodos dessa classe são descritos a seguir. Para cada método é apresentado o seu nome, a sua funcionalidade, os parâmetros de entrada e o tipo de dado retornado pelo método.

Método: *new*
Funcionalidade: método construtor da classe utilizado para armazenar uma nova taxonomia.
Parâmetros de Entrada:

Parâmetro	Tipo	Descrição
$father	*string*	Item de generalização máxima na taxonomia.
@children	*array* de *string*	Itens que são especializações máximas na taxonomia.

Tipo de Retorno: retorna um ponteiro para o objeto que representa a taxonomia.

Método: *init*
Funcionalidade: método que relaciona os parâmetros $father e @children para cada taxonomia.
Parâmetros de Entrada:

Parâmetro	Tipo	Descrição
$father	*string*	Item de generalização máxima na taxonomia.
@children	*array* de *string*	Itens que são especializações máximas na taxonomia.

Tipo de Retorno: não há retorno.

[2] Practical Extraction and Report Language.

Método: *getChildrenItens*
Funcionalidade: método que retorna os itens que são especializações máximas na taxonomia (@children).
Parâmetros de Entrada: não recebe parâmetro.
Tipo de Retorno: retorna um *array* de *string* que representa os itens de especialização máxima na taxonomia.

Método: *getNumberChildrenItens*
Funcionalidade: método que retorna o número de itens que são especializações máximas na taxonomia.
Parâmetros de Entrada: não recebe parâmetro.
Tipo de Retorno: retorna um *integer* que indica o número de itens que são especializações máximas na taxonomia.

Método: *getOneChild*
Funcionalidade: método que retorna um dos itens que são especializações máximas na taxonomia.
Parâmetros de Entrada:

Parâmetro	Tipo	Descrição
$index	*integer*	Posição no *array* @children de um dos itens de especialização máxima.

Tipo de Retorno: retorna uma *string* que representa um dos itens de especialização máxima na taxonomia.

Método: *getFatherItem*
Funcionalidade: método que retorna o item de generalização máxima na taxonomia.
Parâmetros de Entrada: não recebe parâmetro.
Tipo de Retorno: retorna uma *string* ($father) que representa o item de generalização máxima na taxonomia.

4.1.2 Classe ARrule

A classe **ARrule** é responsável pelo armazenamento das regras que serão generalizadas e das Regras de Associação generalizadas. Isso é possível porque as Regras de Associação antes e depois da generalização são representadas na sintaxe padrão definida em Melanda & Rezende (2003). Além disso, a classe **ARrule** provê métodos que permitem acessar as

regras armazenadas. Os métodos dessa classe são descritos a seguir. Para cada método é apresentado o seu nome, a sua funcionalidade, os parâmetros de entrada e o tipo de dado retornado pelo método.

Método: *new*
Funcionalidade: método construtor da classe utilizado para armazenar uma Regra de Associação.
Parâmetros de Entrada:

Parâmetro	Tipo	Descrição
$side	*string*	Lado da regra que será generalizado: *left* (esquerdo) ou *right* (direito).
$rule ID	*integer*	Número de identificação da regra.
$LHS	*string*	Lado esquerdo da regra.
$RHS	*string*	Lado direito da regra.
$TC	*string*	Tabela de Contigência da regra.

Tipo de Retorno: retorna um ponteiro para o objeto que representa a regra.

Método: *generalizeOneRule*
Funcionalidade: método que generaliza uma Regra de Associação, se a regra possui apenas um item que pode ser generalizado.
Parâmetros de Entrada:

Parâmetro	Tipo	Descrição
$oneTaxonomy	ponteiro	Endereço que referencia uma taxonomia.

Tipo de Retorno: não há retorno.

Método: *init*
Funcionalidade: método que utiliza o valor do parâmetro $side para configurar o lado da regra a ser generalizado ($LHS ou $RHS).
Parâmetros de Entrada:

Parâmetro	Tipo	Descrição
$side	*string*	Lado da regra que será generalizado: *left* (esquerdo) ou *right* (direito).
$LHS	*string*	Lado esquerdo da regra.
$RHS	*string*	Lado direito da regra.

Tipo de Retorno: não há retorno.

Método: *calcTC*

Funcionalidade: método que calcula a freqüência relativa dos valores $f(LHS\ RHS)$, $f(LHS\ \overline{RHS})$, $f(\overline{LHS}\ RHS)$ e $f(\overline{LHS}\ \overline{RHS})$, mais o valor N da Tabela de Contigência de uma Regra de Associação. As freqüências relativas correspondem aos valores da tabela divididos por N.

Parâmetros de Entrada:

Parâmetro	Tipo	Descrição
$databaseFile	*string*	Endereço da Base de Dados utilizada na mineração das Regras de Associação que serão generalizadas.
$taxonomiesSet	ponteiro	Endereço que referencia o conjunto de taxonomias que serão utilizadas na generalização.

Tipo de Retorno: não há retorno.

Método: *setSpecRuleIDSet*

Funcionalidade: método que armazena o número de identificação das regras que foram generalizadas e deram origem a uma Regra de Associação generalizada.

Parâmetros de Entrada:

Parâmetro	Tipo	Descrição
@idSet	*array* de *integer*	Números de identificação das Regras de Associação.

Tipo de Retorno: não há retorno.

Método: *setRuleID*

Funcionalidade: método que configura o número de identificação de uma regra.

Parâmetros de Entrada:

Parâmetro	Tipo	Descrição
$id	*integer*	Número de identificação de uma Regra de Associação.

Tipo de Retorno: não há retorno.

Método: *undoGeneralization*

Funcionalidade: método que desfaz a generalização de uma Regra de Associação.

Parâmetros de Entrada: não recebe parâmetro.

Tipo de Retorno: não há retorno.

Método: *increaseQtyItensGen*
Funcionalidade: método que incrementa uma variável $qty_itens_gen. Essa variável contabiliza o número de itens generalizados em uma Regra de Associação.
Parâmetros de Entrada: não recebe parâmetro.
Tipo de Retorno: não há retorno.

Método: *getPosItemGen*
Funcionalidade: método que retorna a posição do último item que foi generalizado na regra.
Parâmetros de Entrada: não recebe parâmetro.
Tipo de Retorno: retorna um *integer* indicando a posição do último item que foi generalizado na regra.

Método: *resetRule*
Funcionalidade: método que configura uma Regra de Associação generalizada como sendo uma regra não generalizada.
Parâmetros de Entrada: não recebe parâmetro.
Tipo de Retorno: não há retorno.

Método: *getSideGen*
Funcionalidade: método que retorna o lado da regra ($LHS ou $RHS) que será generalizado.
Parâmetros de Entrada: não recebe parâmetro.
Tipo de Retorno: retorna uma *string* contendo o lado esquerdo ou direito da regra ($LHS ou $RHS).

Método: *getSpecRuleIDSet*
Funcionalidade: método que retorna os números de identificação das regras que foram generalizadas e deram origem a uma Regra de Associação generalizada.
Parâmetros de Entrada: não recebe parâmetro.
Tipo de Retorno: retorna um *array* de *integer* contendo os números de identificação das regras.

4.1.3 Classe ARigart

A classe **ARigart** implementa o algoritmo $\mathcal{IGART}$, que utiliza um conjunto de taxonomias, definido por um usuário, para generalizar Regras de Associação. Os métodos dessa

classe são descritos a seguir. Para cada método é descrito o seu nome, a sua funcionalidade, os parâmetros de entrada e o tipo de dado retornado pelo método.

Método: *new*
Funcionalidade: método construtor da classe. Pode ser utilizado em qualquer aplicação.
Parâmetros de Entrada:

Parâmetro	Tipo	Descrição
$stem	*string*	Radical do nome do arquivo que contém as Regras de Associação, do nome do arquivo que contém as taxonomias e do nome do arquivo de dados utilizado na mineração das regras que serão generalizadas. O nome dos arquivos de regras, de taxonomias e de dados (sem as devidas extensões) devem ser iguais.
$side	*string*	Lado das Regras de Associação que será generalizado. O parâmetro $side pode assumir os valores *left* ou *right*.
$report	*integer*	O parâmetro indica o tipo de relatório que deve ser gerado pelo algoritmo. O parâmetro pode assumir os valores 0 ou 1. O valor 0 irá gerar apenas um arquivo contendo as informações do processo de generalização e as regras generalizadas. Um exemplo do relatório é ilustrado na Figura 15. Já o valor 1 irá gera 2 arquivos: um contendo informações sobre o processo de generalização e outro contendo apenas as regras generalizadas. Um exemplo do relatório é ilustrado nas Figuras 13 e 14.
$sort	*integer*	O parâmetro indica se as taxonomias utilizadas na generalização devem ser ordenadas ou não (em ordem decrescente) com relação ao número de especializações máximas de cada taxonomia. A ordenação decrescente da taxonomia pode possibilitar que um número maior de regras sejam generalizadas, uma vez que o arquivo de taxonomias é lido seqüencialmente e a ordenação colocaria no início do arquivo as taxonomias que apresentam uma maior quantidade de itens que são especializações máximas. O parâmetro pode assumir os valores 0 (não ordenar) ou 1 (ordenar).

Tipo de Retorno: retorna um ponteiro para o objeto que representa a classe.

Método: *newWeb*
Funcionalidade: método construtor da classe utilizado apenas no Módulo Computacional para Regras de Associação – *RulEE-GAR* (descrito na Seção 5).
Parâmetros de Entrada:

Parâmetro	Tipo	Descrição
$stem	*string*	Radical do nome do arquivo que contém as Regras de Associação e do nome do arquivo que contém as taxonomias. O nome dos arquivos de regras e taxonomias (sem as devidas extensões) devem ser iguais.
$side	*string*	Lado das Regras de Associação que será generalizado. O parâmetro $side pode assumir os valores *left* ou *right*.
$databaseFile	*string*	Endereço do arquivo de dados utilizado na mineração das regras que serão generalizadas.

Tipo de Retorno: retorna um ponteiro para o objeto que representa a classe.

Método: *alg_ igart*
Funcionalidade: método que implementa o algoritmo $\mathcal{IGART}$.
Parâmetros de Entrada: não recebe parâmetro. Os parâmetros para o algoritmo são configurados no construtor da classe.
Tipo de Retorno: retorna um ou dois arquivos (dependendo do tipo de relatório) contendo as informações do processo de generalização e as Regras de Associação generalizadas.

Método: *sortTaxonomies*
Funcionalidade: método auxiliar que realiza a ordenação decrescente das taxonomias comparando o número de itens que são especializações máximas em cada taxonomia.
Parâmetros de Entrada:

Parâmetro	Tipo	Descrição
@taxonomiesSet	*array* de objetos do tipo ARtaxonomy	*Array* que armazena o conjunto de taxonomias que serão utilizadas no processo de generalização das regras.

Tipo de Retorno: não há retorno.

Método: *groupAssocRules*
Funcionalidade: método que agrupa as Regras de Associação comparando o antecedente ou o conseqüente das regras. O método facilita a geração de subconjuntos de regras que apresentam antecedentes ou conseqüentes semelhantes.
Parâmetros de Entrada: não recebe parâmetro.
Tipo de Retorno: retorna um arquivo no formato texto contendo as Regras de Associação agrupadas.

Método: *generalizeRules*
Funcionalidade: método que generaliza um conjunto de Regras de Associação.
Parâmetros de Entrada:

Parâmetro	Tipo	Descrição
@rulesSet	*array* de objetos do tipo ARrule	*Array* que armazena as Regras de Associação que serão generalizadas.
@taxonomiesSet	*array* de objetos do tipo ARtaxonomy	*Array* que armazena as taxonomias que serão utilizadas no processo de generalização das regras.

Tipo de Retorno: não há retorno.

Método: *makeReports*
Funcionalidade: método auxiliar que gera os relatórios de saída do algoritmo $\mathcal{IGART}$.
Parâmetros de Entrada: não recebe parâmetro.
Tipo de Retorno: retorna um ou dois arquivos (dependendo do tipo de relatório) contendo as informações do processo de generalização e as regras generalizadas.

Método: *getNotAnalyzed*
Funcionalidade: método auxiliar que retorna a primeira posição que possui valor 0 em um *array* contendo apenas 0s (zeros) e 1s (uns).
Parâmetros de Entrada:

Parâmetro	Tipo	Descrição
@boolArray	*array* booleano	*Array* que armazena apenas 0s (zeros) e 1s (uns).

Tipo de Retorno: retorna um *integer* indicando a posição no *array*.

Na Figura 6 é apresentado um diagrama que contém os relacionamentos e os métodos de

cada classe do algoritmo $\mathcal{IGART}$ implementado neste trabalho.

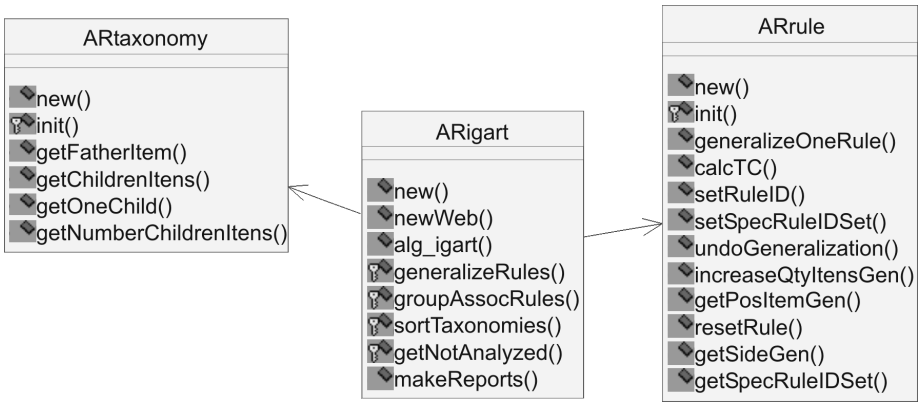

Figura 6: Relacionamentos e métodos das classes do algoritmo $\mathcal{IGART}$

4.2 Arquivos de Entrada e Saída

Nesta seção são descritos os arquivos de entrada utilizados pelo algoritmo $\mathcal{IGART}$, e os arquivos de saída gerados por esse algoritmo.

4.2.1 Arquivos de Entrada

O algoritmo $\mathcal{IGART}$ utiliza 3 arquivos de entrada no formato texto. O primeiro arquivo, que possui a extensão .apr.data, contém o conjunto de dados transacionais que foram utilizados na mineração das Regras de Associação que serão generalizadas. O arquivo segue o formato padrão apresentado no Apêndice A, na qual cada transação é representada por uma linha do arquivo e os valores (itens) em cada transação são separados por um espaço em branco. Na Figura 7 é apresentado um arquivo de dados contendo 8 transações.

```
camiseta chinelo bone tenis
camiseta sandalia bone tenis
bermuda camiseta
bermuda chinelo bone tenis
bermuda sandalia bone tenis
bone tenis camiseta
sandalia chinelo short tenis
sandalia camiseta short tenis
```

Figura 7: Exemplo de um arquivo de dados transacionais .apr.data

O segundo arquivo, que possui a extensão .apr.dcar, contém o conjunto de Regras de Associação que será generalizado. O conjunto de regras está representado na sintaxe

padrão para Regras de Associação definida em Melanda & Rezende (2003). Assim, cada linha do arquivo representa uma Regra de Associação que contém um identificador (ID), o lado esquerdo da regra (LHS), o lado direito da regra (RHS) e a freqüência relativa dos valores $f(LHS\ RHS)$, $f(LHS\ \overline{RHS})$, $f(\overline{LHS}\ RHS)$ e $f(\overline{LHS}\ \overline{RHS})$ da Tabela de Contingência, além do número total de transações consideradas N. Os valores da Tabela de Contingência são delimitados por "[ ]" (colchetes). Cada regra possui as informações apresentadas separadas por "," (vírgula):

$$ID,LHS,RHS,[f(LHS\ RHS),f(LHS\ \overline{RHS}),f(\overline{LHS}\ RHS),f(\overline{LHS}\ \overline{RHS}),N]$$

Em Melanda & Rezende (2003) também é apresentada uma biblioteca de classes que converte as Regras de Associação geradas pelo algoritmo *Apriori* e pelos *softwares Mineset*, *MagnumOpus* e *Weka* para o formato padrão descrito. Na Figura 8 é apresentado um exemplo de um arquivo de Regras de Associação contendo 8 regras que fazem parte do conjunto de regras geradas a partir da Base de Dados apresentada na Figura 7.

```
[R0001],bermuda,tenis,[0.333333,0.000000,0.000000,0.666667,6]
[R0002],camiseta,tenis,[0.500000,0.000000,0.000000,0.500000,6]
[R0003],bermuda & chinelo,bone,[0.166667,0.000000,0.333333,0.500000,6]
[R0004],bermuda & sandalia,bone,[0.166667,0.000000,0.333333,0.500000,6]
[R0005],chinelo & camiseta,bone,[0.166667,0.000000,0.333333,0.500000,6]
[R0006],camiseta & sandalia,bone,[0.166667,0.166667,0.500000,0.166667,6]
[R0007],chinelo & sandalia,short,[0.166667,0.000000,0.666667,0.166667,6]
[R0008],camiseta & sandalia,short,[0.166667,0.166667,0.500000,0.166667,6]
```

Figura 8: Exemplo de um arquivo de regras .apr.dcar

O terceiro e último arquivo utilizado pelo algoritmo $\mathcal{IGART}$ possui a extensão .tax e contém o conjunto de taxonomias que será utilizado para generalizar as Regras de Associação. Cada linha do arquivo representa uma taxonomia que contém um item representando a generalização máxima na taxonomia e, um ou mais itens representando especializações máximas na taxonomia. Os itens que representam especializações máximas são delimitados por "( )" (parênteses) e separados por "," (vírgula). Na Figura 9 é apresentado um arquivo de taxonomias contendo as 4 taxonomias ilustradas na Figura 10.

```
roupas_leves(camiseta,bermuda)
calcados_abertos(sandalia,chinelo)
roupas_sociais(calca,camisa,sapato)
calcados_sociais(sapato)
```

Figura 9: Exemplo do arquivo de taxonomias .tax com as taxonomias da Figura 10

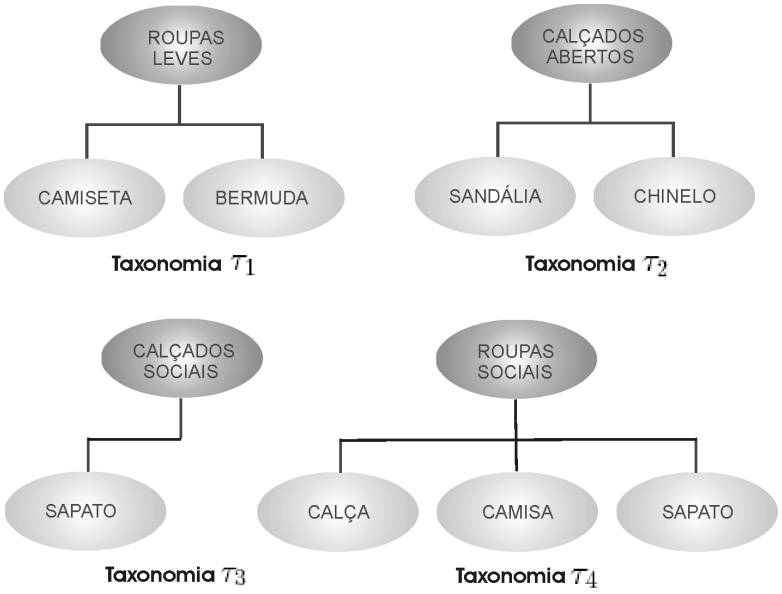

Figura 10: Um conjunto de taxonomias $\mathcal{T}$ com 4 taxonomias

Caso o usuário deseje utilizar taxonomias que contenham mais do que 2 níveis, como ilustrado na Figura 11, o mesmo deve iniciar a escrita do arquivo de taxonomias (.tax) pelos níveis inferiores da taxonomia e terminar pelo nível superior, como é ilustrado na Figura 12.

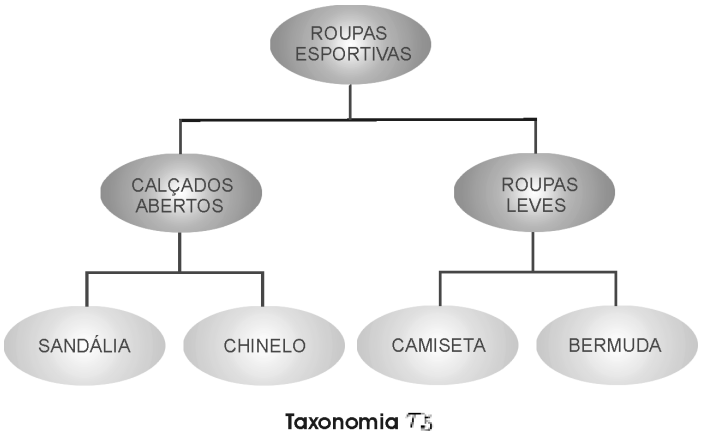

Figura 11: Um conjunto $\mathcal{T}$ contendo uma taxonomia de 3 níveis

```
calcados_abertos(sandalia,chinelo)
roupas_leves(camiseta,bermuda)
roupas_esportivas(calcados_abertos,roupas_leves)
```

Figura 12: Exemplo do arquivo .tax contendo a taxonomia de 3 níveis da Figura 11

4.2.2 Arquivos de Saída

O algoritmo $\mathcal{IGART}$ pode gerar 2 tipos de relatório como saída: um contendo 2 arquivos no formato texto e outro contendo apenas 1 arquivo no formato texto. O primeiro tipo de relatório separa as informações, geradas durante o processo de generalização das Regras de Associação, das regras generalizadas. O arquivo que apresenta as informações do processo de generalização possui a extensão .gar.report e é composto por um cabeçalho, nome dos arquivos de entrada utilizados pelo algoritmo, nome dos arquivos de saída gerados pelo algoritmo, número de regras antes e depois do processo de generalização e uma relação dos identificadores das regras generalizadas com os identificadores das que deram origem a essas regras. Já o arquivo que apresenta as Regras de Associação generalizadas possui a extensão .gar.dcar. O arquivo de regras generalizadas está representado na sintaxe padrão para Regras de Associação definida em Melanda & Rezende (2003), sendo que, os itens das regras que são generalizações estão delimitados por "( )" (parênteses). Exemplos dos arquivos .gar.dcar e .gar.report são apresentados nas Figuras 14 e 13.

```
Association Rules Generalization using IGART Algorithm         Copyright (c) Marcos Aurélio Domingues
Date: Fri Dec  5 12:52:29 2003


ALGORITHM INPUT FILE:
DataBase: files_teste_dis/teste.apr.data
Taxonomies Set: files_teste_dis/teste.tax
Rules Set: files_teste_dis/teste.apr.dcar

ALGORITHM OUTPUT FILE:
Generalized Rules: files_teste_dis/teste.gar.dcar
Report File: files_teste_dis/teste.gar.report

# Rules Set: 8
# Generalized Rules: 4

Generalized Rule ID <=> Source Rule IDs
[R0001] <=> [R0002;R0001]
[R0002] <=> [R0006;R0004;R0003;R0005]
[R0003] <=> [R0008]
[R0004] <=> [R0007]
```

Figura 13: Exemplo de um arquivo de relatório .gar.report

```
[R0001],(roupas_leves),tenis,[0.833333,0.000000,0.000000,0.166667,6]
[R0002],(calcados_abertos) & (roupas_leves),bone,[0.666667,0.166667,0.166667,0.000000,6]
[R0003],camiseta & sandalia,short,[0.166667,0.166667,0.500000,0.166667,6]
[R0004],chinelo & sandalia,short,[0.166667,0.000000,0.666667,0.166667,6]
```

Figura 14: Exemplo de um arquivo de regras generalizadas .gar.dcar

O segundo tipo de relatório coloca as informações geradas durante o processo de generalização das Regras de Associação e as próprias regras generalizadas em um único arquivo, como é apresentado na Figura 15.

```
Association Rules Generalization using IGART Algorithm          Copyright (c) Marcos Aurélio Domingues
Date: Fri Dec  5 12:52:33 2003


ALGORITHM INPUT FILE:
DataBase: files_teste_dis/teste.apr.data
Taxonomies Set: files_teste_dis/teste.tax
Rules Set: files_teste_dis/teste.apr.dcar

ALGORITHM OUTPUT FILE:
Report File: files_teste_dis/teste.gar.dcar.report

# Rules Set: 8
# Generalized Rules: 4

[Generalized Rule ID], LHS, RHS, [Contigency Table (LHSRHS,LHSnotRHS,notLHSnotRHS,notLHSRHS,N)], [Source Rule IDs]

[R0001],(roupas_leves),tenis,[0.833333,0.000000,0.000000,0.166667,6],[R0001;R0002]
[R0002],(calcados_abertos) & (roupas_leves),bone,[0.666667,0.166667,0.166667,0.000000,6],[R0003;R0005;R0004;R0006]
[R0003],chinelo & sandalia,short,[0.166667,0.000000,0.666667,0.166667,6],[R0007]
[R0004],camiseta & sandalia,short,[0.166667,0.166667,0.500000,0.166667,6],[R0008]
```

Figura 15: Exemplo do relatório .gar.dcar.report

O arquivo possui a extensão .gar.dcar.report e é composto por um cabeçalho, nome dos arquivos de entrada utilizados pelo algoritmo, nome dos arquivos de saída gerados pelo algoritmo, número de regras antes e depois do processo de generalização e as Regras de Associação generalizadas. As regras generalizadas estão representadas na sintaxe padrão definida em Melanda & Rezende (2003), com as exceções de que os itens das regras que são generalizações estão delimitados por "( )" (parênteses) e que ao final de cada regra é colocado entre "[ ]" (colchetes), os identificadores das regras que originaram a regra generalizada. Esses identificadores são separados por ";" (ponto e vírgula).

4.3 Interface com o Usuário

Para interagir com o algoritmo $\mathcal{IGART}$, e conseqüentemente utilizar as funcionalidades disponíveis, o usuário precisa executar os métodos implementados nas classes que compõem o algoritmo. Os métodos podem ser invocados de três maneiras. A primeira maneira consiste em executar os métodos dentro do próprio programa do usuário. Nesse caso, o usuário precisa executar primeiramente o construtor da classe (por meio do método *new*) e em seguida executar os métodos que desejar.

A segunda maneira consiste em utilizar um *script* (denominado **generalize.pl**) que atua como uma interface entre o algoritmo e o usuário. Esse *script*, executado por linha de comando do sistema operacional, oculta do usuário detalhes da implementação do algoritmo $\mathcal{IGART}$. A linha de comando é executada como descrito a seguir:

>**generalize.pl** <**stem**> <**side**> **[**<**parâmetros adicionais**>**]**

O primeiro parâmetro utilizado na execução é sempre o *stem*, ou seja, o radical dos nomes do conjunto de dados, conjunto de Regras de Associação e conjunto de taxonomias (não considerando a extensão, todos os arquivos devem ter o mesmo nome). O segundo parâmetro utilizado é o *side*. Esse parâmetro, que aceita os valores *left* (lado esquerdo da regra) e *right* (lado direito da regra), define o lado da regra que será generalizado. Já o terceiro parâmetro é a opção do usuário (gerar apenas um relatório de saída e ordenar as taxonomias antes de generalizar as Regras de Associação). As possíveis opções utilizadas para executar o *script* **generalize.pl**, juntamente com suas respectivas linhas de comando são descritas a seguir:

-onlyreport: o padrão do algoritmo é gerar dois arquivos de relatórios, um arquivo contendo as regras generalizadas (Figura 14) e outro arquivo contendo as informações do processo de generalização das Regras de Associação (Figura 13). A opção **-onlyreport** gera apenas um arquivo de relatório contendo as informações do processo de generalização e as Regras de Associação generalizadas juntas, como ilustrado na Figura 15.

>**generalize.pl** <**stem**> <**side**> **-onlyreport**

-sort: essa opção ordena (em ordem decrescente) o conjunto de taxonomias antes de generalizar o conjunto de Regras de Associação. A ordenação decrescente das taxonomias é realizada utilizando o número de especializações máximas das taxonomias como valores de comparação. Essa ordenação possibilita uma maior generalização das regras, uma vez que as taxonomias que apresentam os maiores números de espe-

cializações máximas serão consideradas primeiro durante o processo de generalização. Cabe ressaltar que o uso dessa opção pode ocasionar uma maior generalização, apenas se o arquivo .tax possui duas ou mais taxonomias sobre o mesmo conjunto de itens. Por exemplo, considerando os itens short, bermuda e camiseta, a opção **-sort** pode ocasionar uma maior generalização das regras se o arquivo .tax possui taxonomias como roupa_esporte(short,bermuda) e roupa_esporte(short,bermuda,camiseta), ou seja, duas ou mais taxonomias sobre o mesmo conjunto de itens.

>**generalize.pl** <**stem**> <**side**> **-sort**

A terceira maneira de utilizar o algoritmo $\mathcal{IGART}$ é por meio do Módulo Computacional para Regras de Associação – *RulEE-GAR*, descrito na próxima seção.

5 O Módulo Computacional *RulEE-GAR*

Nesta seção é apresentado o módulo computacional *RulEE-GAR*, que tem como objetivo fornecer funcionalidades para generalizar Regras de Associação e também para analisar as regras generalizadas. A generalização é realizada utilizando o algoritmo $\mathcal{IGART}$, descrito na Seção 3. Já para prover as funcionalidades de análise de Regras de Associação generalizadas, o módulo utiliza a Base de Dados e a Biblioteca de Classes desenvolvidas para o Ambiente de Exploração de Regras *RulEE* (Paula, 2003).

O ambiente *RulEE* foi desenvolvido com o objetivo de viabilizar tanto a análise quanto a disponibilização de regras de Classificação, Regressão e Associação. Nesse ambiente, o processo de análise e disponibilização das regras está baseado na idéia de se utilizar um Repositório de Regras e Medidas, que é a entidade principal do ambiente. No repositório devem ser armazenados os conjuntos de regras, os valores das medidas e as consultas realizadas pelos usuários. A contextualização do módulo *RulEE-GAR* no ambiente *RulEE*, bem como as suas funcionalidades são ilustradas na Figura 16. Os elementos que estão no retângulo pontilhado da Figura 16 correspondem ao ambiente *RulEE*.

Como pode ser verificado na Figura 16, o módulo computacional *RulEE-GAR* foi desenvolvido como um Módulo de Pós-Processamento do ambiente *RulEE* e embora o mesmo possua uma estrutura que facilita a inclusão de novos módulos, a mesma teve que ser alterada para a inclusão do módulo *RulEE-GAR*. As alterações realizadas são descritas na próxima seção.

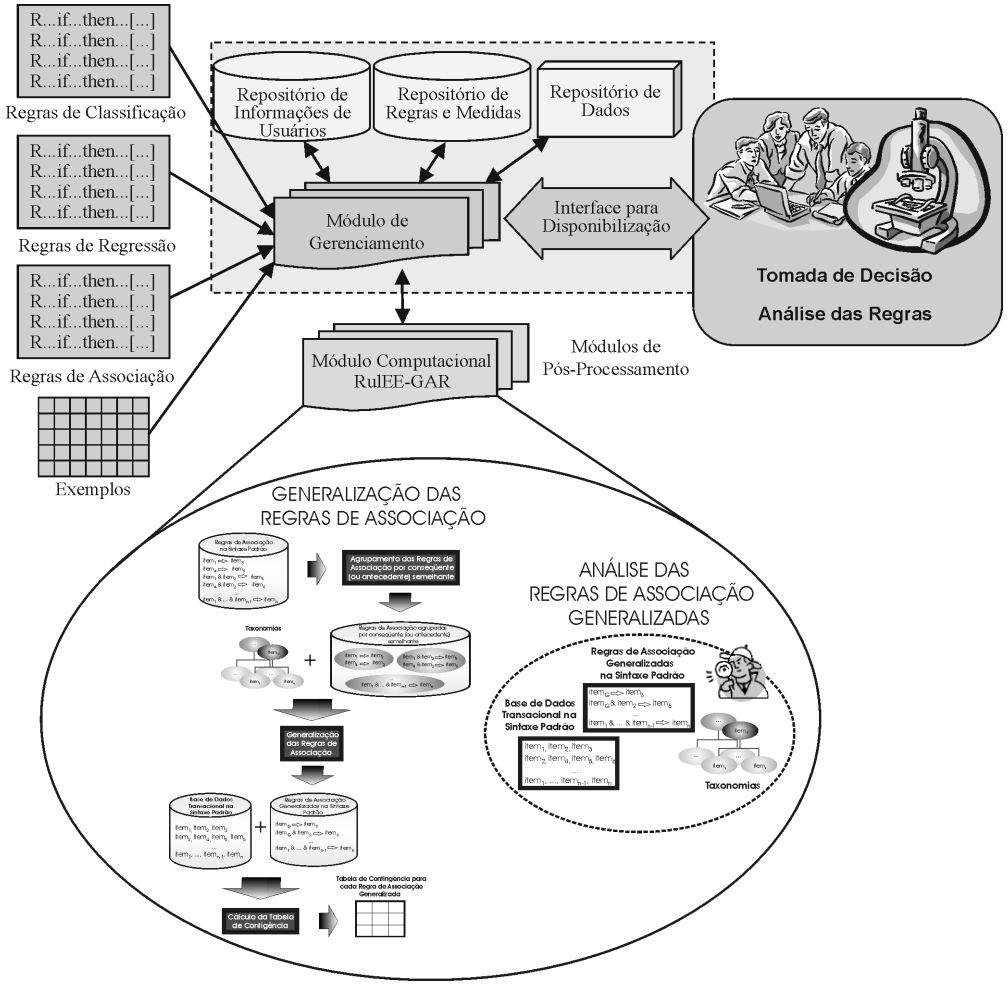

Figura 16: Contextualização e funcionalidades do módulo computacional *RulEE-GAR* no ambiente *RulEE*

5.1 Alterações na Base de Dados

Para armazenar Regras de Associação generalizadas, as tabelas da Base de Dados destinadas ao armazenamento dos conjuntos de regras foram alteradas para que as mesmas pudessem armazenar conjuntos de regras generalizadas (tabelas RULESET e RULE). Além disso, foram criadas três novas tabelas: GENERALIZATION_ITEM (que armazena as taxonomias utilizadas na generalização das regras), RULE_GENERALIZATION (que armazena a relação entre as regras generalizadas e as regras originais) e RULE_GENERALIZATION_ITEM (que armazena a relação entre as regras generalizadas e as taxonomias que foram utilizadas na sua generalização). O diagrama físico da Base de Dados do ambiente *RulEE* é apresentado na Figura 17 na página 42. As alterações realizadas e as novas tabelas criadas, para permitir o armazenamento de Regras de As-

sociação generalizadas no ambiente *RulEE*, estão destacadas por retângulos em negrito, na Figura 17, e são descritas a seguir.

Tabela: RULESET

Descrição: tabela que armazena os conjuntos de regras disponibilizados no ambiente. Os campos acrescidos a essa tabela para que a mesma permita o armazenamento de conjuntos de Regras de Associação generalizadas são descritos a seguir.

Campos Adicionados:

Nome do Campo	Definição	Descrição
GENERALIZED	ENUM('YES', 'NO') NOT NULL	Indica se o conjunto de regras é generalizado.
GENERALIZED_FROM	INTEGER	Se o conjunto de regras é generalizado, essa coluna armazena o identificador do conjunto de regras originais.
GENERALIZATION	TEXT	Se o conjunto de regras é generalizado, essa coluna armazena o conjunto de taxonomias em formato texto.
GENERALIZATION_FILE	VARCHAR(200)	Caso seja um conjunto de regras generalizado, esse campo armazena o endereço e o nome do arquivo que irá armazenar o conjunto de regras.

Tabela: RULE

Descrição: tabela que armazena as regras pertencentes aos conjuntos de regras cadastrados no ambiente. O campo acrescido a essa tabela para que a mesma permita o armazenamento de Regras de Associação generalizadas é descrito a seguir.

Campo Adicionado:

Nome do Campo	Definição	Descrição
GENERALIZED	ENUM('YES', 'NO') NOT NULL	Indica se o conjunto de regras é generalizado.

Tabela: GENERALIZATION_ITEM

Descrição: tabela que armazena as taxonomias utilizadas na generalização de um conjunto de Regras de Associação. Os campos dessa tabela são descritos a seguir.

Campos:

Nome do Campo	Definição	Descrição
RULESET_ID	INTEGER NOT NULL	Identificador do conjunto de regras generalizadas.
GENERALIZATION_ITEM_ID	VARCHAR(50) NOT NULL	Armazena o item de generalização máxima de uma taxonomia. Por exemplo, na taxonomia suco(laranja, limão), esse campo armazena o item suco.
GENERALIZATION_ITEM	VARCHAR(100) NOT NULL	Armazena os itens de especialização máxima de uma taxonomia. Por exemplo, na taxonomia suco(laranja, limão), esse campo armazena os itens laranja, limão.

Restrições de Integridade:

Nome da Restrição	Definição
PK_GENERALIZATION_ITEM	PRIMARY KEY(RULESET_ID, GENERALIZATION_ITEM_ID)
FK_GENERALIZATION_ITEM_RULESET	FOREIGN KEY(RULESET_ID) REFERENCES RULESET

Tabela: RULE_GENERALIZATION

Descrição: tabela que armazena o relacionamento NxM da tabela RULE. Esse relacionamento permite a recuperação das regras originais que deram origem a uma Regra de Associação generalizada. Os campos dessa tabela são descritos a seguir.

Campos:

Nome do Campo	Definição	Descrição
RULE_ID	INTEGER NOT NULL	Identificador da Regra de Associação original.
GENERALIZATION_RULE_ID	INTEGER NOT NULL	Identificador da Regra de Associação generalizada.

Restrições de Integridade:

Nome da Restrição	Definição
PK_RULE_GENERALIZATION	PRIMARY KEY(RULE_ID, GENERALIZATION_RULE_ID)
FK_RULE_GENERALIZATION_RULE_1	FOREIGN KEY(RULE_ID) REFERENCES RULE
FK_RULE_GENERALIZATION_RULE_2	FOREIGN KEY(GENERALIZATION_RULE_ID) REFERENCES RULE

Tabela: RULE_GENERALIZATION_ITEM

Descrição: tabela que armazena o relacionamento NxM da tabela RULE com a tabela GENERALIZATION_ITEM. Esse relacionamento permite a recuperação das taxonomias que foram utilizadas na generalização das Regras de Associação. Os campos dessa tabela são descritos a seguir.

Campos:

Nome do Campo	Definição	Descrição
RULE_ID	INTEGER NOT NULL	Identificador da Regra de Associação generalizada.
RULESET_ID	INTEGER NOT NULL	Identificador do conjunto de Regras de Associação generalizadas.
GENERALIZATION_ITEM_ID	INTEGER NOT NULL	Item de generalização máxima de uma taxonomia. Por exemplo, na taxonomia suco(laranja, limão), esse campo armazena o item suco.

Restrições de Integridade:

Nome da Restrição	Definição
PK_RULE_GENER_ITEM	PRIMARY KEY(RULE_ID, RULESET_ID, GENERALIZATION_ITEM_ID)
FK_RULE_GENER_ITEM_GENER_ITEM	FOREIGN KEY(GENERALIZATION_ITEM_ID, RULESET_ID) REFERENCES GENERALIZATION_ITEM
FK_RULE_GENER_ITEM_RULE	FOREIGN KEY(RULE_ID) REFERENCES RULE

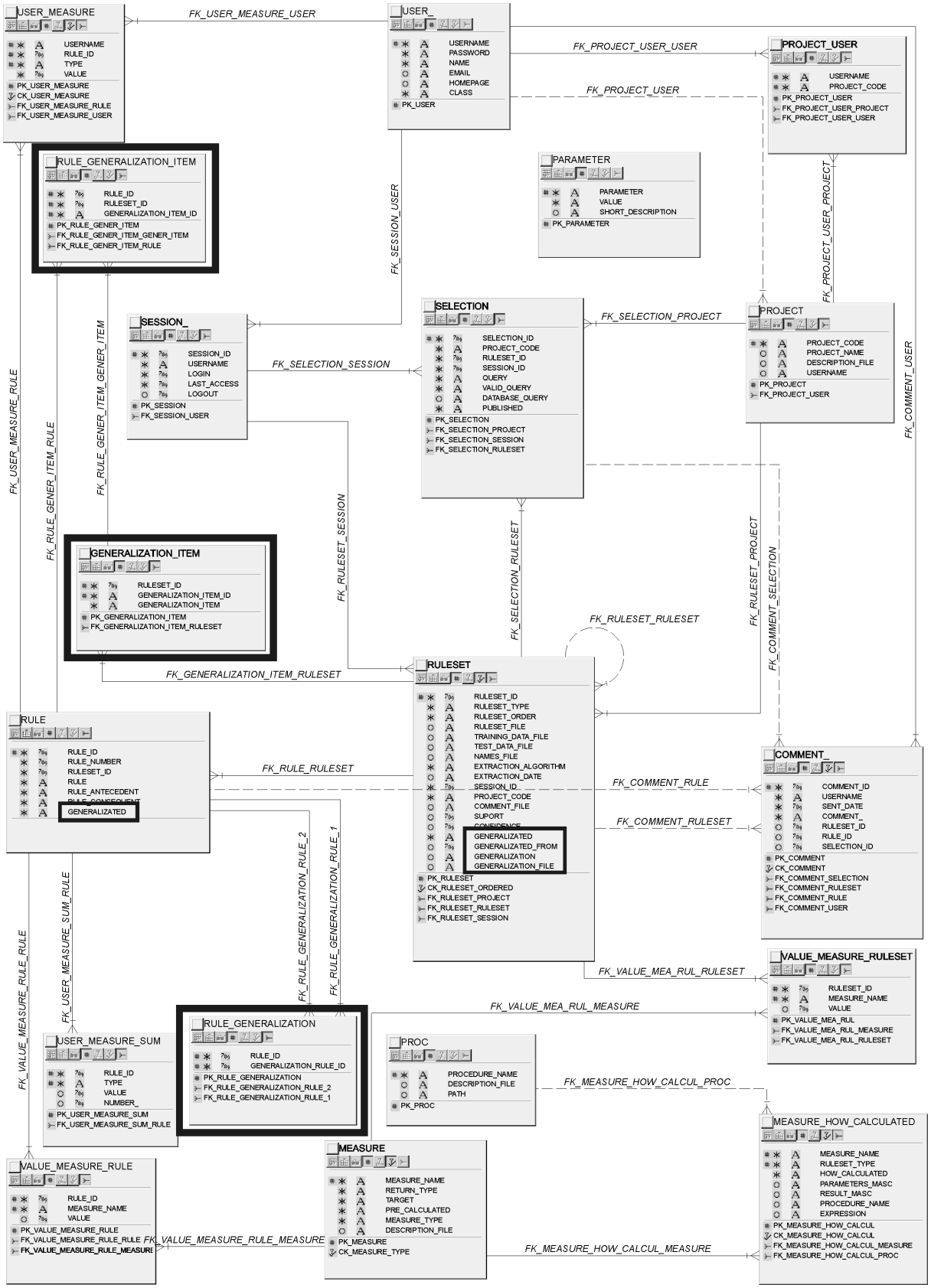

Figura 17: Diagrama físico da Base de Dados do ambiente *RulEE*

5.2 Alterações no Módulo de Gerenciamento

Para o ambiente *RulEE* armazenar e acessar as Regras de Associação generalizadas foram implementados vários métodos para as classes que compõem o seu Módulo de Gerenciamento. Uma descrição dos métodos implementados, em suas respectivas classes, é apresentada a seguir.

5.2.1 Classe Rule

Essa classe representa uma regra cadastrada no ambiente. Os métodos implementados, para que a classe possa armazenar uma Regra de Associação generalizada, são descritos a seguir.

Método: *IsGeneralizated*
Funcionalidade: verifica se uma regra é ou não generalizada.
Parâmetros de Entrada: não recebe parâmetro.
Tipo de Retorno: não há retorno.

Método: *GetGeneralizationItems*
Funcionalidade: retorna as taxonomias que foram utilizadas na generalização de uma regra.
Parâmetros de Entrada: não recebe parâmetro.
Tipo de Retorno: um vetor contendo as taxonomias.

Método: *Remove*
Funcionalidade: remove uma regra. Esse método já existe no ambiente e foi alterado para remover uma Regra de Associação generalizada. Além disso, apenas o usuário que inseriu as regras na Base de Dados pode removê-las.
Parâmetros de Entrada: não recebe parâmetro.
Tipo de Retorno: não há retorno.

Método: *GetGeneralizatedBy*
Funcionalidade: retorna as regras que são generalizações de uma dada regra.
Parâmetros de Entrada: não recebe parâmetro.
Tipo de Retorno: um vetor de objetos do tipo Rule.

Método: *GetOriginalRules*
Funcionalidade: retorna as regras originais de uma dada regra generalizada.
Parâmetros de Entrada: não recebe parâmetro.
Tipo de Retorno: um vetor de objetos do tipo Rule.

5.2.2 Classe Project

Essa classe representa um projeto existente no ambiente *RulEE*, sendo utilizada para retornar informações referentes a um determinado projeto, como sua descrição e seus conjuntos de regras. O único método implementado nessa classe para a manipulação de Regras de Associação generalizadas é descrito a seguir.

Método: *GetGeneralizatedRulesets*
Funcionalidade: retorna os conjuntos de Regras de Associação generalizadas relacionados a um determinado projeto.
Parâmetros de Entrada: não recebe parâmetro.
Tipo de Retorno: um vetor de objetos do tipo Ruleset.

5.2.3 Classe Ruleset

Essa classe encapsula um conjunto de regras de classificação, regressão e associação. Os métodos implementados para que a classe possa manipular Regras de Associação generalizadas são descritos a seguir.

Método: *NewGeneralizationRuleset*
Funcionalidade: insere um conjunto de Regras de Associação generalizadas no ambiente *RulEE*.
Parâmetros de Entrada:

Parâmetro	Tipo	Descrição
$ruleset_file_name	*string*	Nome do arquivo contendo o conjunto de regras generalizado.
$generalization_file_name	*string*	Nome do arquivo contendo o conjunto de taxonomias.
$original_ruleset_id	*integer*	Identificador do conjunto de regras original utilizado na generalização.

$comment_file_name	*string*	Nome do arquivo contendo comentários sobre o Ruleset.
$data_file_name	*string*	Nome do arquivo contendo os dados utilizados para gerar o Ruleset.
$test_file_name	*string*	Nome do arquivo contendo os dados de teste do Ruleset.
$names_file_name	*string*	Nome do arquivo contendo a descrição dos arquivos de dados.
$project_code	*string*	Código do projeto ao qual o Ruleset estará associado.
$suport	*real*	Valor do suporte mínimo utilizado na mineração do Ruleset.
$confidence	*real*	Valor da confiança mínima utilizada na mineração do Ruleset.
$session	*integer*	Número da sessão do ambiente utilizada para a inserção do conjunto de regras. Pode ser substituído por $username e $passord, ou seja, pelo username e senha do usuário no ambiente *RulEE*.

Tipo de Retorno: retorna o ponteiro para o objeto que representa o conjunto de Regras de Associação generalizadas criado na Base de Dados.

Método: *GetGeneralizationItems*
Funcionalidade: retorna as taxonomias que foram utilizadas na generalização de um conjunto de regras.
Parâmetros de Entrada: não recebe parâmetro.
Tipo de Retorno: um vetor contendo as taxonomias.

Método: *Remove*
Funcionalidade: remove um conjunto de regras. Esse método já existe no ambiente e foi alterado para remover um conjunto de Regras de Associação generalizadas. Além disso, apenas o usuário que inseriu o conjunto de regras na Base de Dados pode removê-lo.
Parâmetros de Entrada: não recebe parâmetro.
Tipo de Retorno: não há retorno.

Método: *IsGeneralizated*
Funcionalidade: verifica se um conjunto de regras é ou não generalizado.
Parâmetros de Entrada: não recebe parâmetro.
Tipo de Retorno: não há retorno.

Método: *GetGeneralizatedBy*
Funcionalidade: retorna os conjuntos de regras que são generalizações de um conjunto de regras.
Parâmetros de Entrada: não recebe parâmetro.
Tipo de Retorno: um vetor de objetos do tipo Ruleset.

Método: *GetOriginalRules*
Funcionalidade: retorna o conjunto de regras original utilizado na generalização.
Parâmetros de Entrada: não recebe parâmetro.
Tipo de Retorno: um objeto do tipo Ruleset.

Método: *GetGeneralization*
Funcionalidade: retorna um conjunto de taxonomias armazenado no ambiente *RulEE*.
Parâmetros de Entrada: não recebe parâmetro.
Tipo de Retorno: um vetor contendo as taxonomias.

Método: *GetGeneralizationFileName*
Funcionalidade: retorna o nome de um arquivo no formato texto que contém um conjunto de taxonomias já inserido no ambiente *RulEE*.
Parâmetros de Entrada: não recebe parâmetro.
Tipo de Retorno: uma *string* indicando o nome do arquivo.

O diagrama de classes que apresenta os relacionamentos entre as classes do ambiente *RulEE* é ilustrado na Figura 18.

Na próxima seção é descrita a interface do módulo *RulEE-GAR* e apresentados exemplos de uso da interface na generalização de Regras de Associação e na análise das regras generalizadas.

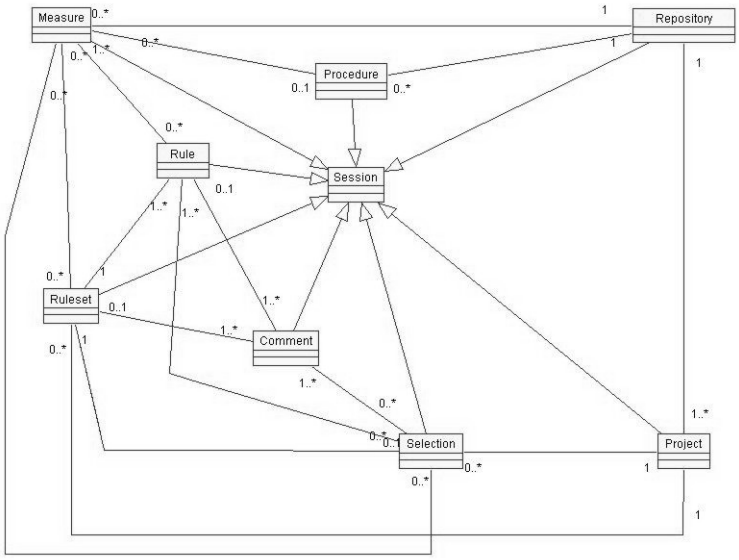

Figura 18: Relacionamento entre as classes do ambiente *RulEE*. Fonte: (Paula, 2003)

5.3 Descrição da Interface do Módulo *RulEE-GAR*

A interface do módulo é carregada utilizando o ambiente *RulEE* (disponível no endereço `http://143.107.231.137/rulee/index.html`), ilustrado nas Figuras 19 e 20. É importante destacar que, para utilizar a interface do módulo *RulEE-GAR* é necessário que o *browser* esteja com a opção *Java Script* habilitada, caso contrário, muitas de suas funcionalidades não estarão acessíveis.

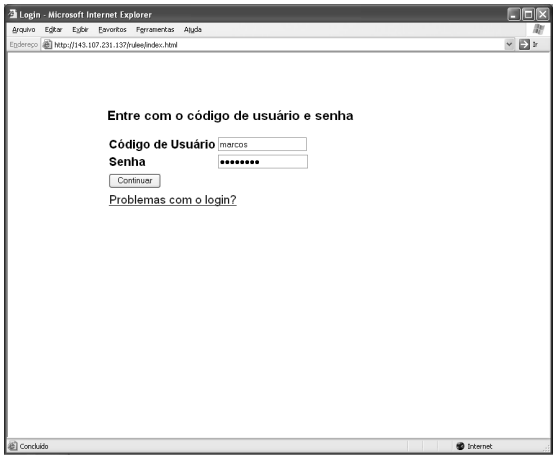

Figura 19: *Login* no ambiente *RulEE*

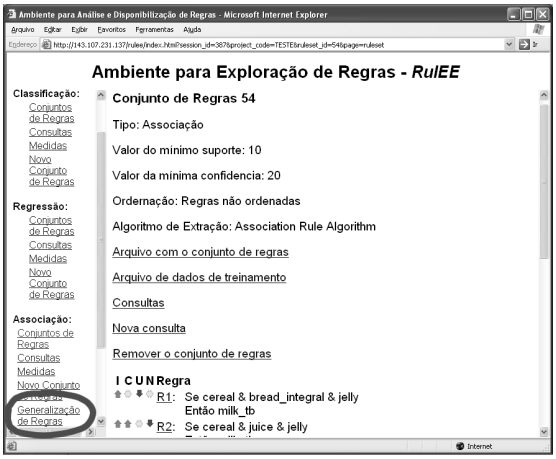

Figura 20: *Menu* principal do ambiente *RulEE*

Após a autenticação do usuário no ambiente *RulEE*, pode-se acessar o módulo

RulEE-GAR, ilustrado nas Figuras 21 e 22, por meio do *link* Generalização de Regras (Figura 20).

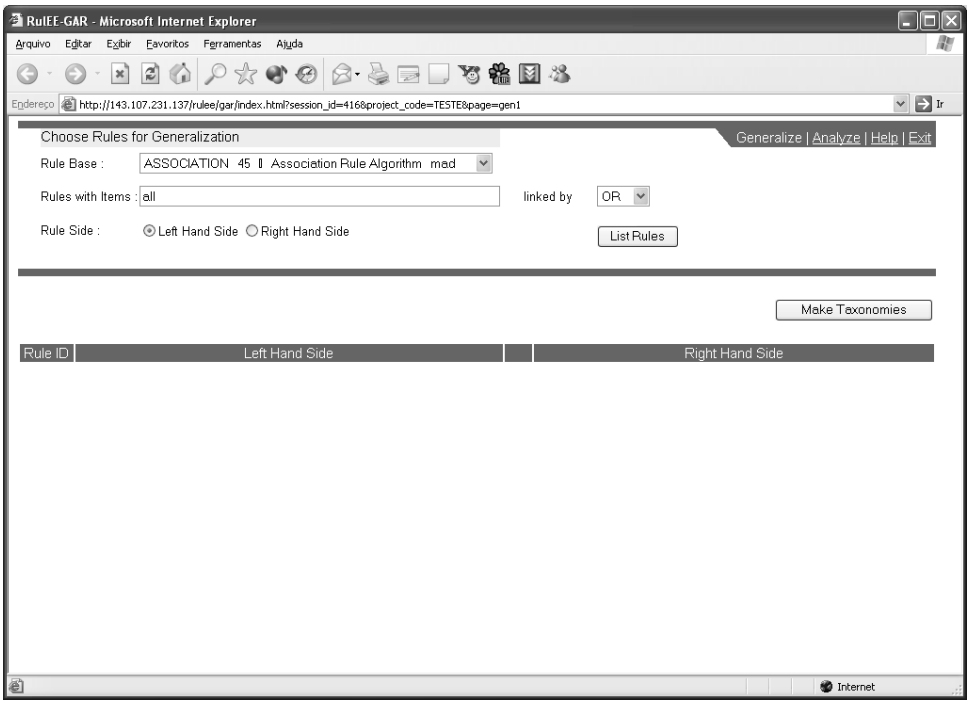

Figura 21: Interface para generalização de Regras de Associação

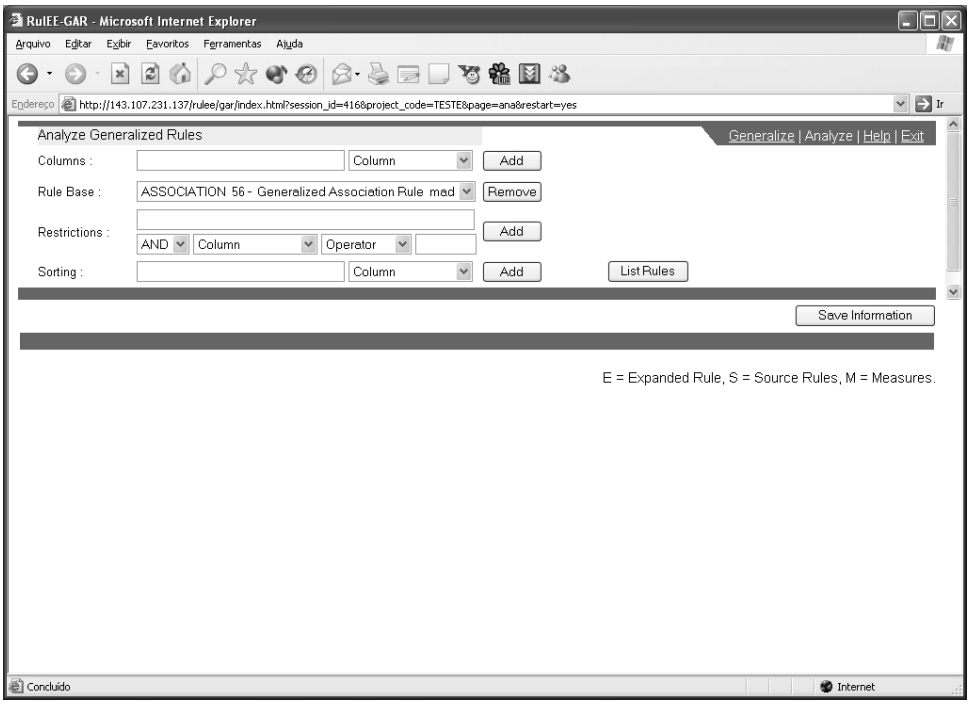

Figura 22: Interface para análise das Regras de Associação generalizadas

Na Figura 21 é ilustrada a interface do módulo *RulEE-GAR* que utiliza o algoritmo $\mathcal{IGART}$ (descrito na Seção 3) para generalizar os conjuntos de Regras de Associação armazenados no ambiente *RulEE*. Já na Figura 22 é ilustrada a interface que permite analisar e explorar os conjuntos de Regras de Associação generalizadas. A navegação pelo módulo *RulEE-GAR* é realizada utilizando o *menu* disponibilizado no canto superior direito da interface, como pode ser visualizado na Figura 22. O *menu* principal do módulo fornece ao usuário as seguinte opções:

Link Generalize Permite o acesso a interface de generalização de Regras de Associação. Essa interface é ilustrada na Figura 21 e descrita na Seção 5.4.

Link Analyze Fornece acesso a interface de análise e exploração da Regras de Associação generalizadas. A interface é ilustrada na Figura 22 e descrita na Seção 5.5.

Link Help Permite o acesso ao sistema de ajuda do módulo. Esse sistema é ilustrado na Figura 23.

Link Exit Finaliza a execução do módulo e fecha o *browser*.

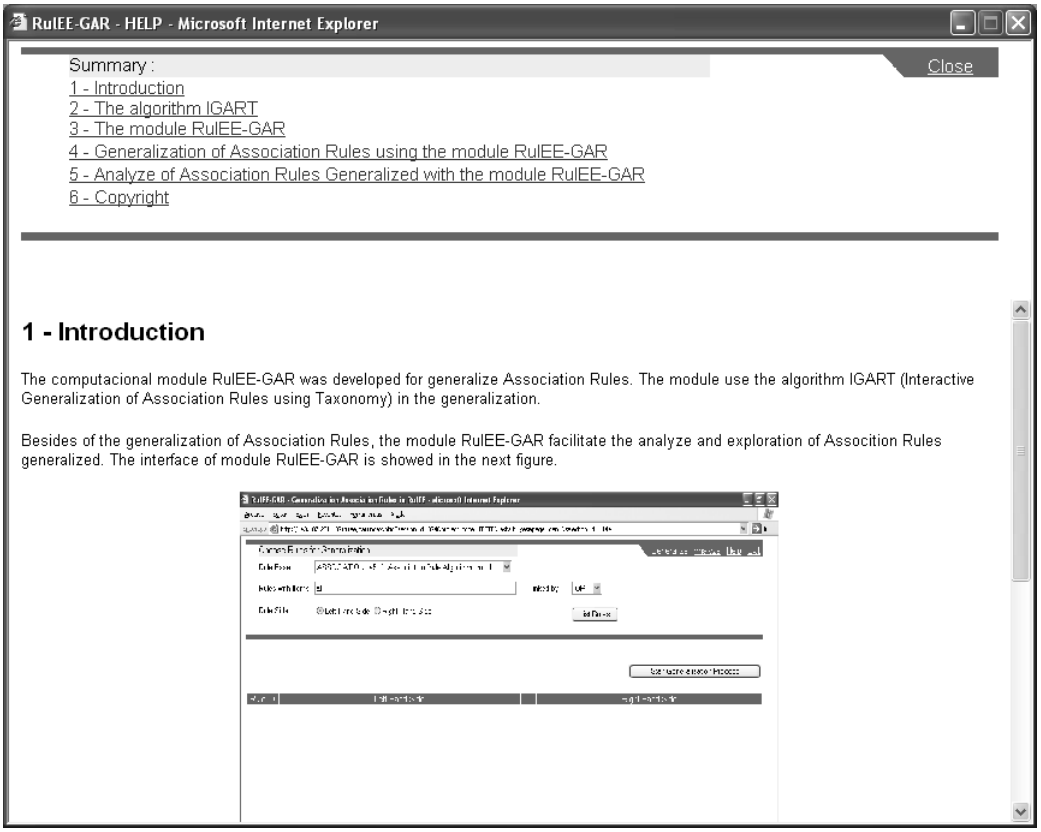

Figura 23: Sistema de ajuda (*Help*) do módulo *RulEE-GAR*

Na próxima seção é exemplificado o uso do módulo computacional *RulEE-GAR* na generalização de um conjunto de Regras de Associação.

5.4 Generalização de Regras de Associação Utilizando o Módulo *RulEE-GAR*

O *link Generalize*, descrito anteriormente, dá acesso a tela da interface de generalização de Regras de Associação ilustrada na Figura 21. Nessa tela há alguns campos para entrada de dados que permitem ao usuário selecionar um conjunto de regras para ser generalizado. Os campos são descritos a seguir:

Rule Base Nesse campo o usuário seleciona a base de regras da qual será selecionado um conjunto de regras para ser generalizado. Visando facilitar a identificação da base de regras, cada elemento desse campo apresenta as seguintes informações: tipo da base de regras, identificador, data de extração das regras, algoritmo utilizado e o usuário responsável pela inserção da base no ambiente *RulEE*.

Rules with Items Nesse campo deve ser inserido os itens (separados por um espaço em branco) que devem estar presente nas regras selecionadas. Se nenhum item for especificado ou se for inserido o item "*all*", a seleção irá considerar que qualquer item pode estar presente nas Regras de Associação a serem generalizadas.

linked by Nesse campo o usuário define a relação entre os itens, definidos no campo *Rules with Items*, que devem estar presente na regra. Existem duas opções de relação: a opção "*OR*" que irá selecionar regras que tenham pelo menos um dos itens especificados no campo *Rules with Items* e a opção "*AND*" que irá selecionar apenas as regras que tenham todos os itens especificados.

Rule Side Nesse campo é definido o lado, das Regras de Associação, que deve ser generalizado. Como apresentado na Seção 3, o algoritmo $\mathcal{IGART}$ generaliza apenas um lado da regra – lado esquerdo (LHS) ou lado direito (RHS). Além disso, a escolha do lado da regra a ser generalizado define:

- o lado na qual as restrições dos campos *Rules with Items* e *linked by* deverão ser aplicadas, uma vez que, no módulo implementado, essas restrições são aplicadas apenas no lado das regras que será generalizado;
- o modo como as regras selecionadas deverão ser ordenadas antes de serem apresentadas para o usuário. Se o usuário escolher generalizar o lado esquerdo

das regras, as mesmas serão ordenadas primeiro pelo lado direito e em seguida pelo lado esquerdo, caso contrário, as regras serão ordenadas primeiro pelo lado esquerdo e em seguida pelo lado direito. A ordenação do conjunto de regras, antes de sua visualização, facilita a identificação de taxonomias pelo usuário.

Para visualizar o conjunto de Regras de Associação selecionado deve-se utilizar a opção *List Rules*. Para cada regra do conjunto é apresentado o seu número de identificação (*Rule ID*), o seu lado esquerdo (*Left Hand Side*) e o seu lado direito (*Right Hand Side*). Na Figura 24 é ilustrada a seleção de um pequeno conjunto de regras para ser generalizado.

Após a seleção do conjunto de regras a ser generalizado, deve-se definir as taxonomias que serão utilizadas na generalização das regras selecionando a opção *Make Taxonomies*. A tela da interface que permite o usuário definir as taxonomias que serão utilizadas na generalização das regras é ilustrada na Figura 25.

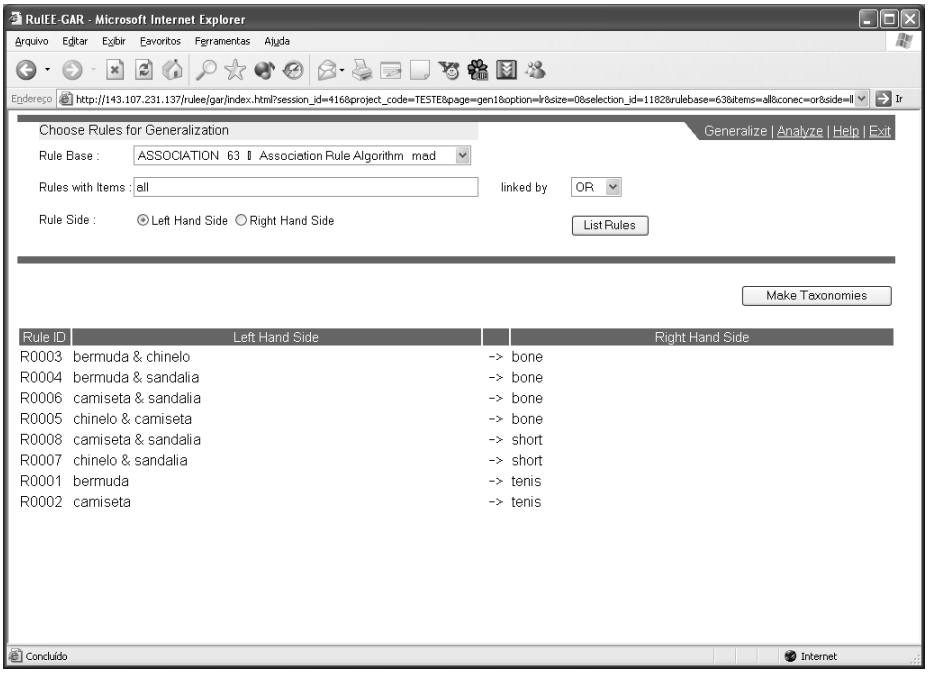

Figura 24: Seleção de um conjunto de Regras de Associação para ser generalizado

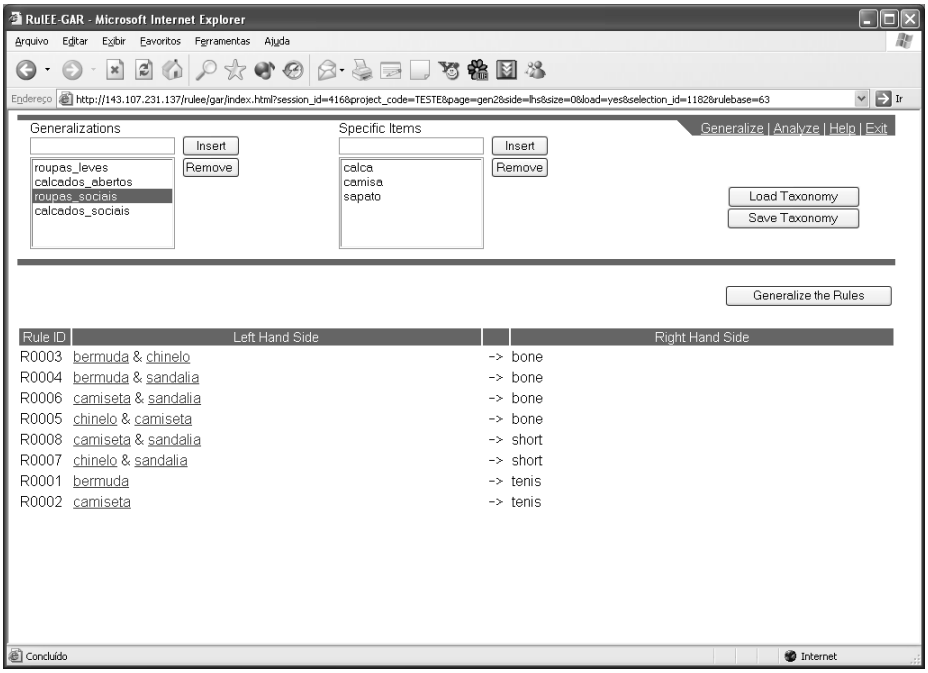

Figura 25: Definição das taxonomias para a generalização das regras

Na tela apresentada na Figura 25, o usuário possui duas maneiras para montar as taxonomias que serão utilizadas na generalização das regras:

Montando as taxonomias manualmente Na montagem manual, o usuário utiliza o campo *Generalizations* para inserir o nome de cada generalização, ou seja, para inserir o item de generalização máxima na taxonomia. No campo *Specific Items*, o usuário informa os itens específicos que devem ser generalizados, ou seja, os itens que são especializações máximas na taxonomia. Assim, os itens específicos podem ser inseridos na taxonomia utilizando o campo *Specific Items* ou clicando nos itens das regras que serão generalizadas. Como pode ser visualizado na Figura 25, os itens das Regras de Associação se tornam *links* para facilitar a montagem das taxonomias.

Carregando um arquivo de taxonomias Com essa opção pode ser utilizado um arquivo de taxonomias (.tax), igual ao definido na Seção 4.2.1, para generalizar as regras. Além das duas maneiras para montar as taxonomias, o usuário pode também salvá-las em um arquivo .tax.

Definidas as taxonomias a serem utilizadas, o usuário pode executar o algoritmo $\mathcal{IGART}$ para generalizar as regras. A execução do algoritmo é realizada na opção *Generalize the Rules*. Ao término da execução do algoritmo é apresentado ao usuário uma mensagem

informando que o processo de generalização foi realizado com sucesso, como é ilustrado na Figura 26.

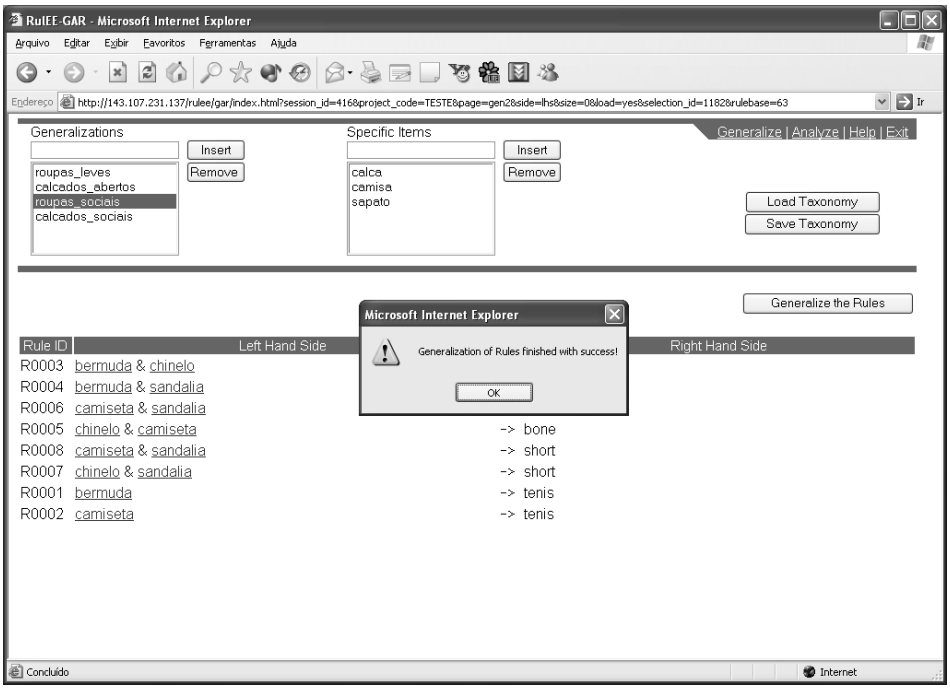

Figura 26: Término do processo de generalização das Regras de Associação

Após a mensagem, o módulo retorna a interface ilustrada na Figura 25, onde o usuário pode realizar um outra generalização ou pode selecionar o *link Analyze* para analisar as regras generalizadas. O uso do módulo *RulEE-GAR* na análise de Regras de Associação generalizadas é apresentado na próxima seção.

5.5 Análise de Regras de Associação Generalizadas Utilizando o Módulo *RulEE-GAR*

O *link Analyze*, apresentado na Seção 5.3, dá acesso a tela da interface de análise de Regras de Associação generalizadas ilustrada na Figura 22. Nessa tela há alguns campos para entrada de dados que permitem ao usuário selecionar um conjunto de regras generalizadas para ser analisado. Os campos são descritos a seguir:

Columns Nesse campo devem ser inseridas as colunas a serem visualizadas nos dados finais referentes à consulta. Dentre as colunas que podem ser utilizadas nesse campo,

encontram-se *RULE*, referenciando a regra completa, *RULE_NUMBER*, referenciando o número da regra dentro do conjunto de regras, e os nomes das medidas disponíveis (que foram cadastradas no ambiente *RulEE*), por exemplo, *Acc* referenciando a precisão da regra. Uma lista contendo as colunas disponíveis para utilização se encontra ao lado do campo *Columns*.

Rule Base Nesse campo deve ser selecionada a base de regras da qual serão selecionadas as regras a serem apresentadas. Visando facilitar a identificação da base de regras, cada elemento desse campo apresenta as seguintes informações: tipo da base de regras, identificador da base, data da generalização das regras, descrição de que a base foi generalizada e o usuário responsável pela generalização da base de regras. O botão *Remove*, ao lado do campo *Rule Base*, permite o usuário remover do ambiente *RulEE* uma base de Regras de Associação generalizadas.

Restrictions Nesse campo devem ser inseridas as restrições a serem aplicadas às regras visando verificar quais regras devem ser selecionadas pela consulta. No campo podem ser utilizadas restrições sobre quaisquer medidas disponíveis para o conjunto de Regras de Associação generalizadas. Abaixo do campo *Restrictions* existe alguns campos para auxiliar o usuário na elaboração das restrições. Por esses campos, o usuário pode selecionar a coluna sobre a qual deseja aplicar a restrição, o operador a ser utilizado e o valor de teste.

Sorting Nesse campo deve ser especificada a lista de colunas para ordenação das regras, podendo ser utilizadas quaisquer colunas disponíveis para a base de Regras de Associação generalizadas.

Para visualizar o conjunto de Regras de Associação generalizadas deve-se selecionar a opção *List Rules*. Na Figura 27 é ilustrada a seleção de um pequeno conjunto de regras generalizadas.

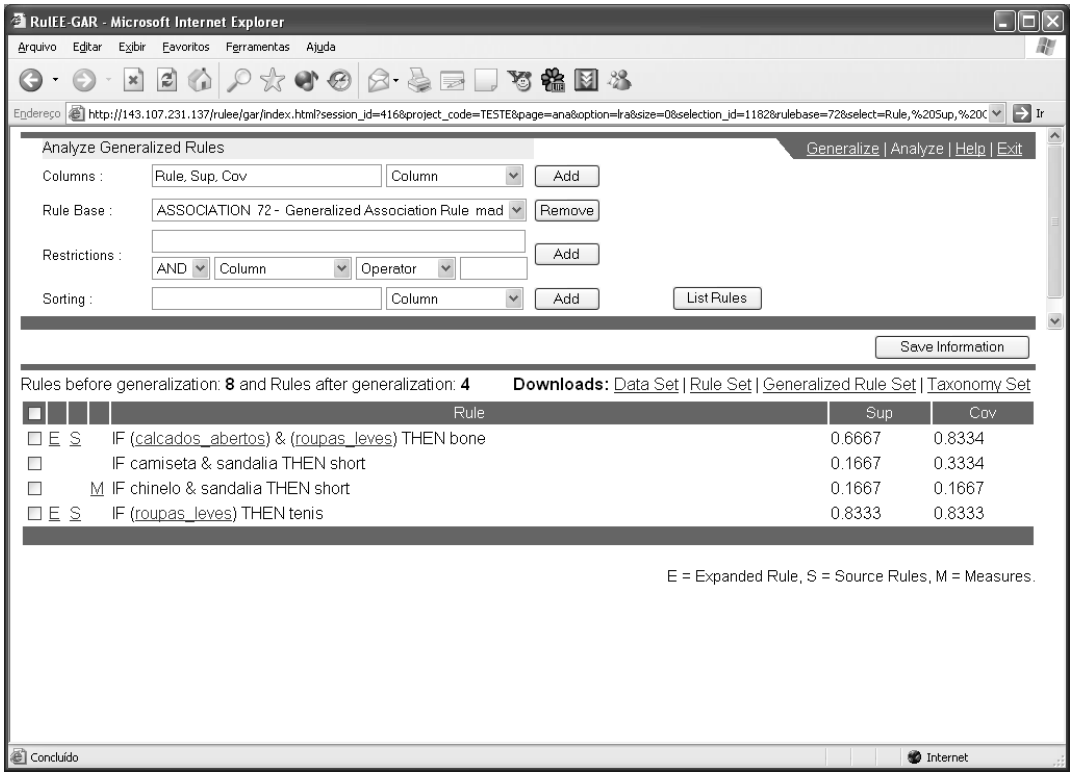

Figura 27: Consulta a uma base de Regras de Associação generalizadas

Na Figura 27 é possível verificar que além das colunas selecionadas pelo usuário, a consulta retorna outras informações como, por exemplo, o número total de regras da base generalizada (*Rules after generalization*) e da base original (*Rules before generalization*). Junto com cada consulta realizada, a interface disponibiliza quatro *links* na seção *Downloads* para o usuário visualizar ou fazer o *download* dos arquivos contendo, respectivamente, o conjunto de dados transacionais (*link Data Set*), o conjunto de regras originais (*link Rule Set*), o conjunto de regras generalizadas (*link Generalized Rule Set*) e o conjunto de taxonomias utilizadas (*link Taxonomy Set*). Com exceção do arquivo contendo o conjunto de regras generalizadas, os demais arquivos estão no formato descrito na Seção 4.2.1. Já o arquivo que contém o conjunto de regras generalizadas, embora esteja representado na sintaxe padrão definida por Melanda & Rezende (2003), o mesmo apresenta os itens generalizados das regras delimitados por "( )" (parênteses). Além disso, ao final de cada regra é listado os identificadores das regras originais (que são disponibilizadas pelo *link Rule Set*) que geraram a regra generalizada. Esses identificadores são separados por ";" (ponto e vírgula) e delimitados por "[ ]" (colchetes). Um exemplo de cada arquivo disponível, para visualização ou *download*, é ilustrado nas Figuras 28, 29, 30 e 31.

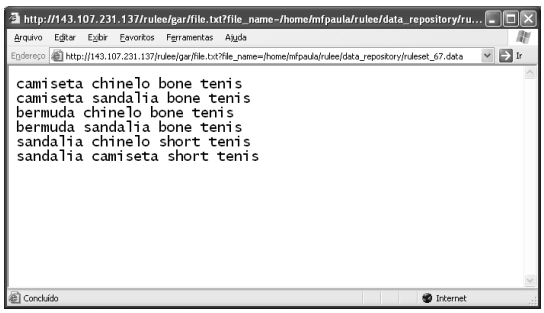

Figura 28: Visualização do arquivo de dados transacionais (*link Data Set*)

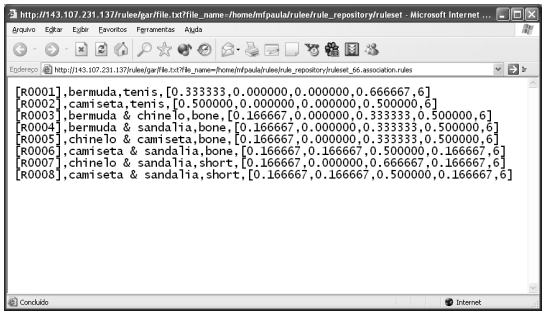

Figura 29: Visualização do arquivo de regras originais (*link Rule Set*)

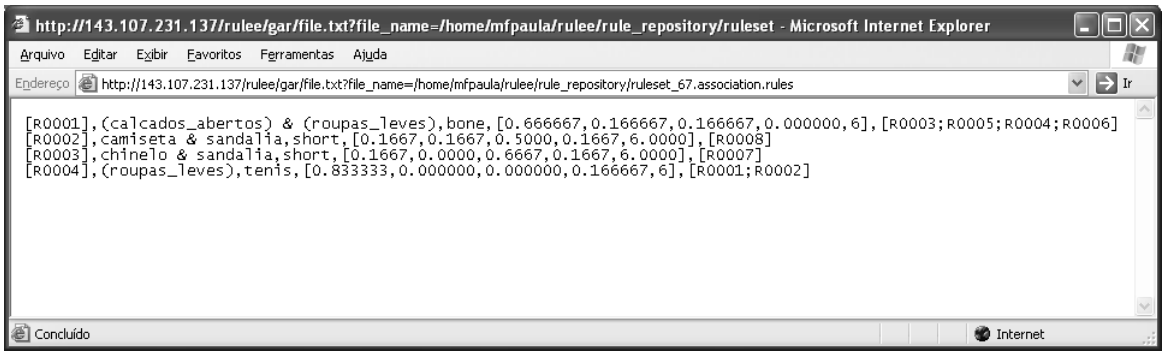

Figura 30: Visualização do arquivo de regras generalizadas (*link Generalized Rule Set*)

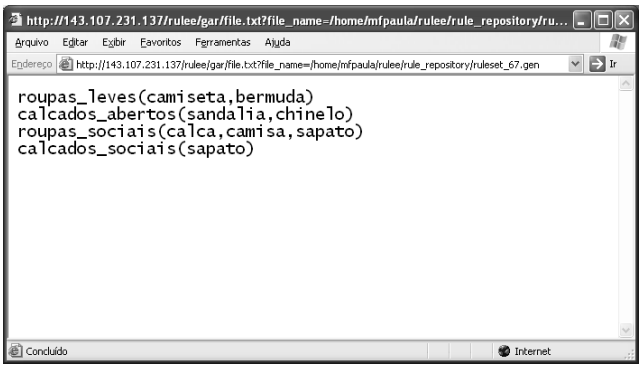

Figura 31: Visualização do arquivo de taxonomias (*link Taxonomy Set*)

Além dos *links* para visualização ou *download* dos arquivos, cada Regra de Associação generalizada, listada pela consulta, apresenta alguns *links* que permitem explorar informações relacionadas a sua generalização. Os *links* que são descritos a seguir, estão posicionados a esquerda das regras (Figura 27).

Link Expanded Rule - E

O *link Expanded Rule*, representado na interface pela letra "E" (Figura 27), permite a visualização da regra generalizada de modo expandida, ou seja, os itens generalizados em uma regra (que são generalizações máximas na taxonomia) são substituídos pelos respectivos itens específicos (que são especializações máximas na taxonomia). Na Figura 32 é ilustrada a visualização de uma Regra de Associação generalizada expandida.

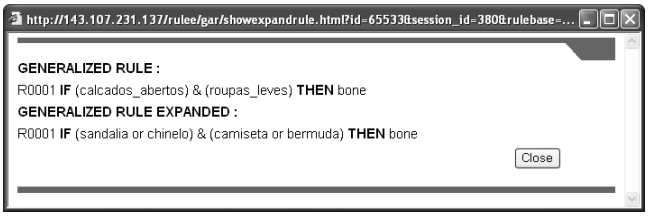

Figura 32: Tela de visualização de uma Regra de Associação generalizada expandida

Link Source Rules - S

O *link Source Rules*, representado na interface pela letra "S" (Figura 27), permite ao usuário visualizar as regras originais que geraram uma Regra de Associação generalizada, como é ilustrado na Figura 33.

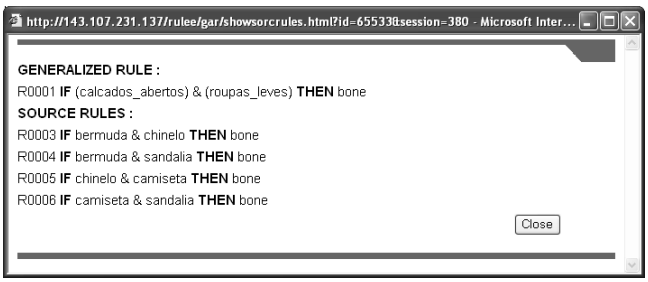

Figura 33: Tela de visualização das Regras de Associação que geraram uma regra generalizada

Link Measures - M

O *link Measures*, representado pela letra "M" (Figura 27), é visualizado na interface apenas se o usuário seleciona as medidas suporte (*Sup*) e/ou confiança (*Cov*) em sua consulta.

Como o ambiente *RulEE* possibilita que o usuário, ao inserir um conjunto de Regras de

Associação, possa também inserir o valores de suporte e confiança mínimos utilizados na mineração desse conjunto, o *Link Measures* permite ao usuário comparar os valores de suporte e confiança mínimos de um conjunto de regras não generalizadas com os valores de suporte e confiança de uma regra generalizada obtida a partir desse conjunto. Essa análise é importante, pois embora o suporte de uma Regra de Associação generalizada seja sempre igual ou maior do que o valor do suporte mínimo definido na mineração das regras não generalizadas, a mesma garantia não ocorre para a medida confiança, ou seja, uma Regra de Associação generalizada pode ter confiança menor do que a confiança mínima definida na mineração das regras. Na Figura 34 é ilustrada a tela que permite ao usuário comparar os valores das medidas suporte e confiança.

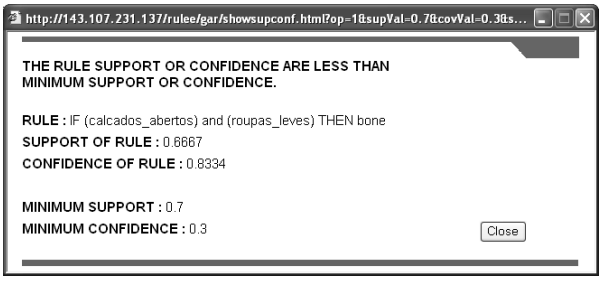

Figura 34: Tela de visualização dos valores das medidas suporte e confiança de uma regra generalizada

Na Figura 27 também é possível verificar que os itens generalizados em uma regra (que estão delimitados por parênteses) são apresentados como *links*, para que o usuário possa visualizar os itens originais que foram generalizados e geraram esse item. A tela que permite a visualização de itens originais a partir de um item generalizado é ilustrada na Figura 35.

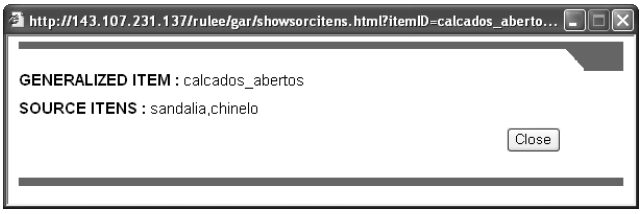

Figura 35: Tela de visualização de itens originais a partir de um item generalizado

Por fim, o usuário também tem a opção de salvar em um arquivo no formato texto, as informações listadas na consulta que ele considerar importante. Para isso basta marcar as informações consideradas importantes (utilizando os *checkboxs*) e selecionar a opção *Save Information*.

6 Considerações Finais

O objetivo deste trabalho foi o de oferecer uma visão detalhada do algoritmo $\mathcal{IGART}$ e do módulo computacional *RulEE-GAR*. O algoritmo $\mathcal{IGART}$ (*Interactive Generalization of Association Rules using Taxonomies* – Generalização Interativa de Regras de Associação usando Taxonomias) utiliza taxonomias para generalizar Regras de Associação. Já o módulo *RulEE-GAR*, além de facilitar o uso do algoritmo $\mathcal{IGART}$ durante a identificação de taxonomias e generalização de regras, provê funcionalidades para analisar as Regras de Associação generalizadas.

Neste relatório foram descritos o algoritmo $\mathcal{IGART}$ e as funções que fazem parte do mesmo, uma exemplificação de uso do algoritmo e a biblioteca de classes que implementa o algoritmo $\mathcal{IGART}$, sendo relatada a funcionalidade dos métodos implementados, os arquivos de entrada e de saída e a interface para uso do algoritmo. Em seguida foi apresentado o módulo computacional *RulEE-GAR*, sendo descrito exemplos de uso do módulo tanto para a generalização de Regras de Associação quanto para a análise das regras generalizadas. Ao final do relatório é apresentada a sintaxe padrão definida para representar conjuntos de dados transacionais.

Cabe ressaltar que o algoritmo $\mathcal{IGART}$ e o módulo computacional *RulEE-GAR* fazem parte do *Discover* (Baranauskas & Batista, 2000; Rezende et al., 2004), um projeto em desenvolvimento pelos pesquisadores do LABIC (Laboratório de Inteligência Computacional - ICMC/USP) que tem como objetivo fornecer um ambiente integrado para apoiar as etapas do processo de Extração de Conhecimento de Dados e Textos.

A A Sintaxe Padrão para Conjuntos de Dados Transacionais

Em Batista & Monard (2003) é proposto, para o projeto *Discover*, uma sintaxe padrão para representação de dados. Essa sintaxe padrão, denominada DSX (*Discover Standard Sintax*), utiliza arquivos no formato texto para declarar os atributos e seus respectivos tipos, e os valores que esses atributos assumem em um conjunto de dados.

A sintaxe proposta em Batista & Monard (2003) facilita o uso de Bases de Dados relacionais, porém dificulta o uso de Bases de Dados transacionais no *Discover*. Por esse motivo, nesta seção é proposta uma sintaxe padrão para a representação de dados transacionais, denominada DTSX (*Discover Transactions Standard Sintax*). Essa sintaxe utiliza um

arquivo no formato texto, com a extensão .trans, para declarar os itens de um conjunto de transações. Na Figura 36 é apresentada a representação genérica de um arquivo no formato texto de uma Base de Dados transacional na sintaxe DTSX.

```
item1 item2 item3 item4 ...
item1 item5 item3 ...
item6 item1 ...
item6 item2 item3 item4 ...
item6 item5 item3 item8 ...
item3 item4 item1 ...
item5 item2 item7 item4 ...
...
```

Figura 36: Representação de um arquivo de dados transacionais .trans

No arquivo .trans cada linha de declaração de dados representa uma transação. Sendo assim, o caracter responsável por identificar o final de uma transação é o caracter de nova linha (representado em muitas linguagens de programação por "`\n`"). Cada transação possui uma seqüência de valores (itens) separados por um espaço em branco, ou seja, o espaço em branco é o caracter responsável por separar os itens de uma transação. Em Bases de Dados transacionais, os itens que fazem parte de uma transação, independente do que representam, são sempre valores nominais e, na sintaxe DTSX, podem ser compostos por qualquer caracter exceto o espaço em branco e a tabulação (representada em muitas linguagens de programação por "`\t`"). Na Figura 37 é apresentado um arquivo de dados contendo 8 transações, no qual cada transação representa as compras realizadas por um determinado cliente de uma loja.

```
camiseta chinelo bone tenis
camiseta sandalia bone tenis
bermuda camiseta
bermuda chinelo bone tenis
bermuda sandalia bone tenis
bone tenis camiseta
sandalia chinelo short tenis
sandalia camiseta short tenis
```

Figura 37: Exemplo de um arquivo de dados transacionais

A sintaxe DTSX para o arquivo de transações .trans é definida pela gramática $G = (V_n, V_t, P, S)$, em que:

S: <transação>

V_n: {<transação>, <item>, <letra>, <número>, <símbolo>}

V_t: {a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z, A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, ~, ‘, #, %, ^, &, *, +, -, :, ;, ’, ’’, <, >, ,, ., ?, /, _, =, (,), {, }, [,], @, $ }

P: Produções da gramática G usando a notação BNF. As produções são apresentadas na Figura 38.

```
S ::=
    <transação>

<transação> ::=
    <item> | <item> <transação>

<item> ::=
    <letra> | <letra><item> | <número> | <número><item> | <símbolo>
  | <símbolo><item>

<letra> ::= a | b | c | d | e | f | g | h | i | j | k | l | m | n | o
          | p | q | r | s | t | u | v | w | x | y | z | A | B | C | D
          | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S
          | T | U | V | W | X | Y | Z

<número> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<símbolo>::= ~ | ‘ | # | $ | ^ | & | * | + | - | \ | : | ; | " | ’ | @
           | < | > | , | . | ? | / | _ | = | ( | ) | { | } | [ | ] | %
```

Figura 38: Produções da gramática G para Bases de Dados transacionais

Referências

Adamo, J.-M. (2001). *Data Mining for Association Rules and Sequential Patterns.* New York, NY: Springer-Verlag. 1, 2, 3

Agrawal, R. & R. Srikant (1994). Fast algorithms for mining association rules. Em J. B. Bocca, M. Jarke, & C. Zaniolo (eds.), *Proceedings 20th International Conference on Very Large Data Bases, VLDB*, pp. 487–499. 2

Baesens, B., S. Viaene, & J. Vanthienen (2000). Post-processing of association rules. Em *Proceedings of the Special Workshop on Post-Processing. The Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 2–8. 1

Baranauskas, J. A. & G. E. A. P. A. Batista (2000). O projeto *Discover*: Idéias iniciais (comunicação pessoal). 23, 59

Batista, G. E. & M. C. Monard (2003). Descrição da arquitetura e do projeto do ambiente computacional *Discover Learning Environment* - DLE. Relatório Técnico do ICMC/USP - Número 187. Disponível em: `http://www.icmc.usp.br/~biblio/BIBLIOTECA/rel_tec/RT_187.PDF`. Acesso em: 15/12/2003. 59

Clementini, E., P. D. Felice, & K. Koperski (2000). Mining multiple-level spatial association rules for objects with a broad boundary. *Data & Knowledge Engineering 34*(3), 251–270. 1

Fayyad, U. M., G. Piatetsky-Shapiro, & P. Smyth (1996). From data mining to knowledge discovery: An overview. Em U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, & R. Uthurusamy (eds.), *Advances in Knowledge Discovery and Data Mining*, Volume 1, pp. 1–30. Menlo Park, CA: American Association for Artificial Intelligence. 1

Lavrač, N., P. Flach, & R. Zupan (1999). Rule evaluation measures: A unifying view. Em S. Dzeroski & P. Flach (eds.), *Proceedings of the Ninth International Workshop on Inductive Logic Programming (ILP-99)*, Volume 1634, pp. 174–185. Springer-Verlag. LNAI. 6, 15

Liu, B., W. Hsu, S. Chen, & Y. Ma (2000). Analyzing the subjective interestingness of association rules. *IEEE Intelligent Systems & their Applications 15*(5), 47–55. 1, 3

Melanda, E. A. & S. O. Rezende (2003). Sintaxe padrão para representar regras de associação. Relatório Técnico do ICMC/USP - Número 206. Disponível em: `http://www.icmc.usp.br/~biblio/BIBLIOTECA/rel_tec/rt_206.pdf`. Acesso em: 15/12/2003. 6, 24, 32, 34, 35, 55

Paula, M. F. (2003). Ambiente para exploração de regras. Dissertação de Mestrado, Instituto de Ciências Matemáticas e de Computação - Universidade de São Paulo, São Carlos, SP - Brasil. ii, 37, 47

Rezende, S. O., M. C. Monard, G. E. A. P. A. Batista, R. C. Prati, J. B. Pugliesi, & E. A. Melanda (2004). Ambiente *Discover*: Sintaxes padrão de dados e regras. Relatório Técnico do ICMC/USP. Prelo. 23, 59

Rezende, S. O., J. B. Pugliesi, E. A. Melanda, & M. F. Paula (2003). Mineração de dados. Em S. O. Rezende (ed.), *Sistemas Inteligentes: Fundamentos e Aplicações*, Volume 1, pp. 307–335. Barueri, SP: Editora Manole. 2

Semenova, T., M. Hegland, W. Graco, & G. Williams (2001). Effectiveness of mining association rules for identifying trends in large health databases. Em *Workshop on Integrating Data Mining and Knowledge Management. ICDM'01: The 2001 IEEE International Conference on Data Mining*. Disponível em `http://cui.unige.ch/~hilario/icdm-01/DM-KM-Final/Semenova.pdf`. Acesso em: 11/12/2002. 1

Silberschatz, A. & A. Tuzhilin (1995). On subjective measures of interestingness in knowledge discovery. Em U. M. Fayyad & R. Uthurusamy (eds.), *Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD-95)*, pp. 275–281. 1

Srikant, R. & R. Agrawal (1997). Mining generalized association rules. *Future Generation Computer Systems 13*(2–3), 161–180. 1

Till, D. (1996). *Teach Yorself Perl 5 in 21 Days*. Indianapolis, USA: Sams Publishing. 23

Weiss, S. M. & N. Indurkhya (1998). *Predictive Data Mining: A Practical Guide*. San Francisco, CA: Morgan Kaufmann. 2