**UNIVERSIDADE FEDERAL DE SÃO CARLOS**

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

# MINERAÇÃO DE REGRAS DE ASSOCIAÇÃO GENERALIZADAS UTILIZANDO ONTOLOGIAS FUZZY E SIMILARIDADE BASEADA EM CONTEXTO

**RODRIGO MOURA JUVENIL AYRES**

**ORIENTADORA: PROF. DRA. MARILDE TEREZINHA PRADO SANTOS**

São Carlos - SP
Agosto/2012

**UNIVERSIDADE FEDERAL DE SÃO CARLOS**

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

# MINERAÇÃO DE REGRAS DE ASSOCIAÇÃO GENERALIZADAS UTILIZANDO ONTOLOGIAS FUZZY E SIMILARIDADE BASEADA EM CONTEXTO

**RODRIGO MOURA JUVENIL AYRES**

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação, área de concentração: Engenharia de Software.
Orientadora: Dra. Marilde Terezinha Prado Santos

São Carlos - SP
Agosto/2012

**Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária da UFSCar**

A985mr

Ayres, Rodrigo Moura Juvenil.
Mineração de regras de associação generalizadas utilizando ontologias fuzzy e similaridade baseada em contexto / Rodrigo Moura Juvenil Ayres. -- São Carlos : UFSCar, 2012.
115 f.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2012.

1. Banco de dados. 2. Data mining (Mineração de dados). 3. Regras de associação. 4. Ontologia difusa. 5. Pós-processamento. 6. Similaridade baseada em contexto. I. Título.

CDD: 005.74 (20ª)

# Universidade Federal de São Carlos

## Centro de Ciências Exatas e de Tecnologia

### Programa de Pós-Graduação em Ciência da Computação

# "Mineração de Regras de Associação Generalizadas utilizando Ontologias Fuzzy e Similaridade baseada em Contexto"

Rodrigo Moura Juvenil Ayres

**Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.**

**Membros da Banca:**

**Profa. Dra. Marilde Terezinha Prado Santos**
**(Orientadora - DC/UFSCar)**

**Profa. Dra. Marcela Xavier Ribeiro**
**( DC/UFSCar)**

**Profa. Dra. Marina Teresa Pires Vieira**
**(UNIMEP)**

**São Carlos**
**Agosto/2012**

*Para meus pais, Edson (in memoriam) e Nielse, a quem, além de Deus, devo tudo o que tenho, o que sou, e o que ainda serei.*

*Em especial, à minha esposa Pâmela, pois sem ela certamente este trabalho não seria o mesmo.*

# AGRADECIMENTOS

Primeiramente agradeço a Deus, que me conhece, me guia e me capacita, dando-me paz, inteligência e, acima de tudo, saúde para poder caminhar.
Agradeço à minha orientadora, Dra. Marilde, pela oportunidade que me deu, pela sua confiança, amizade, paciência, compreensão e pelo conhecimento que me proporcionou durante todo o mestrado. Muito obrigado.
Ao professor Dr. Ricardo Ciferri pela ajuda, e orientações.
Aos colegas do Departamento de Computação e do laboratório LaBDES, pelos momentos de estudos e alegria.
Aos colegas do grupo de banco de dados, pela amizade e ajuda.
À CAPES, pelo auxílio financeiro deste trabalho.

Muito obrigado a todos.

*"Quando a gente acha que tem todas as respostas, vem a vida e muda todas as perguntas..."*

*Luís Fernando Veríssimo*

# RESUMO

Algoritmos tradicionais de associação se caracterizam por utilizar apenas itens contidos na base de dados, proporcionando um conhecimento muito específico. No entanto, essa especificidade nem sempre é vantajosa, pois normalmente os usuários finais necessitam de padrões mais gerais, e de fácil compreensão. Nesse sentido, existem abordagens que não se limitam somente aos itens da base, e trabalham com o objetivo de minerar regras (generalizadas) com itens presentes em qualquer nível de estruturas taxonômicas. Taxonomias podem ser utilizadas em diferentes etapas do processo de mineração. A literatura mostra que, em contextos crisp, essas estruturas são utilizadas tanto em etapa de pré-processamento, quanto em etapa de pós-processamento, e que em domínios fuzzy, a utilização ocorre somente na etapa de pré-processamento, durante a geração de transações estendidas. Além do viés de utilização de transações estendidas, que podem levar a geração de um volume de regras superior ao caso tradicional, é possível notar que, em domínios nebulosos, as pesquisas dão enfoque apenas à mineração de regras fuzzy, deixando de lado a exploração de diferentes graus de especialização/generalização em taxonomias. Nesse sentido, este trabalho propõem o algoritmo FOntGAR, um novo algoritmo para mineração de regras de associação generalizadas com itens presentes em qualquer nível de ontologias compostas por graus de especialização/generalização variando no intervalo [0,1] (ontologias de conceitos fuzzy), em etapa de pós-processamento. Objetivando obter maior enriquecimento semântico, as regras geradas pelo algoritmo também podem possuir relações de similaridade, de acordo com contextos pré-definidos. Outros pontos relevantes são a especificação de uma nova abordagem de generalização (incluindo um novo tratamento de agrupamento das regras), e um novo e eficiente método para calcular o suporte estendido das regras generalizadas durante a etapa mencionada.

**Palavras-chave**: Regras de Associação Generalizadas, Ontologias Difusas, Pós-processamento, similaridade baseada em contexto.

# ABSTRACT

The mining association rules are an important task in data mining. Traditional algorithms of mining association rules are based only on the database items, providing a very specific knowledge. This specificity may not be advantageous, because the users normally need more general, interesting and understandable knowledge. In this sense, there are approaches working in order to obtain association rules with items belonging to any level of a taxonomic structure. In the crisp contexts taxonomies are used in different steps of the mining process. When the objective is the generalization they are used, mainly, in the pre-processing or post-processing stages. On the other hand, in the fuzzy context, fuzzy taxonomies are used, mainly, in the pre-processing step, during the generating extended transactions. A great problem of these transactions is related to the huge amount of candidates and rules. Beyond that, the inclusion of ancestors ends up generating redundancy problems. Besides, it is possible to see that many works have directed efforts for the question of mining fuzzy rules, exploring linguistic terms, but few approaches have been proposed for explore new steps of mining process. In this sense, this paper proposes the *Context* FOntGAR algorithm, a new algorithm for mining generalized association rules under all levels of fuzzy ontologies composed by specialization/generalization degrees varying in the interval [0,1]. In order to obtain more semantic enrichment, the rules may be composed by similarity relations, which are represented at the fuzzy ontologies in different contexts. In this work the generalization is done during the post-processing step. Other relevant points of this paper are the specification of a new approach of generalization; including a new grouping rules treatment, and a new and efficient way for calculating both support and confidence of generalized rules.

**Keywords**: Generalized Association Rules, Fuzzy Ontologies, Post-Processing, Context-Based Similarity.

# LISTA DE FIGURAS

Figura 2.1 - Visão Geral dos Passos do Processo KDD – adaptada de (FAYYAD, *et al.*, 1996b) .....20

Figura 2.2 – Principais Tarefas de Mineração de Dados.....22

Figura 2.3 - Exemplo de Taxonomia – adaptada de (Srikant e Agrawal, 1995). .....26

Figura 2.4 – Abordagem de agrupamento por similaridade. .....31

Figura 2.5 – Taxonomia simples. .....33

Figura 3.1 - Funções de Pertinência para *Jovem, Adulto* e *Idoso* – adaptada de Klir e Yuan (1995).....39

Figura 3.2 - Representação Gráfica dos Conceitos Jovem, Adulto e Idoso – adaptada de (Klir e Yuan, 1995). .....40

Figura 4.1 - Usos de Ontologias – adaptada de Uschold e Gruninger (1996).....44

Figura 4.2 – Exemplo de Ontologia Simples (Yaguinuma, 2007).....46

Figura 4.3 - Tipos de Representações de Ontologias (Yaguinuma, 2007).....47

Figura 4.4 - Tipos de Ontologias – adaptada de (Guarino, 1998). .....48

Figura 4.5 - Exemplo de Ontologia Difusa com Classes e Relacionamentos Difusos. .....53

Figura 4.6 – Representação abstrata de classe nebulosa. .....54

Figura 4.7 – Representação em OWL de uma instância de classe difusa. .....54

Figura 4.8 – Representação abstrata de relacionamento difuso binário. .....55

Figura 4.9 – Representação em OWL de relacionamento difuso binário. .....55

Figura 4.10- Representação abstrata de relacionamento difuso com contexto.....57

Figura 4.11 – Representação em OWL de relacionamentos fuzzy em vários contextos.....57

Figura 5.1 – Diagrama do algoritmo - adaptada de Mohamadlou *et al.* (2009). .....62

Figura 5.2 - Etapas do *NARFO* Miani (2009).....69

Figura 6.1 – Etapas do Algoritmo FOntGAR. .....73

Figura 6.2 – Generalização de Regras de Associação usando Agrupamento. .....74

Figura 6.3 – Generalização de regras utilizando agrupamento por similaridade e ancestrais da estrutura taxonômica.....75

Figura 6.4 – Processo utilizado no agrupamento de regras de associação (generalização no antecedente ou consequente).....77

Figura 6.5 – Pseudocódigo para generalização antecedente/consequente. .....78

Figura 6.6 – Pseudocódigo para generalização antecedente ou consequente. .....79

Figura 6.7 - Exemplo de Figura e Legenda (Yaguinuma, 2007).....80

Figura 6.8 – Taxonomia Simples de Domínio Alimentício. .....81

Figura 6.9 – Base de Dados em Formato Vertical.....81

Figura 6.10 – Redução de uma Ontologia a uma taxonomia relativa onde todos os itens folhas estrutura extraída estejam presentes na base de dados.....83

Figura 6.11 – Ilustração de grupos de regras com uma regra mais geral representante. .....84

Figura 6.12 – Pseudocódigo Tratamento de Generalização. .....85

Figura 6.13 – Formato em que o Resultado é Apresentado.....86

Figura 6.14 – Estrutura utilizada para armazenar itens do banco e ancestrais da ontologia.....89

Figura 6.15 – Estrutura utilizada para mapear as chaves (ancestrais) aos graus de suporte das transações. .....89

Figura 6.16 – Representação da ideia utilizada no cálculo de suporte estendido. ....90

Figura 6.17 – Exemplo Simples de Folhas com Similaridade.....92

Figura 6.18 – Pseudocódigo utilizado na Etapa de Identificação e representação de Similaridade.....93

Figura 7.1 – Taxonomia Crisp de Características Demográficas. .....96

Figura 7.2 – Representação para Anos de Estudo, Sexo e Etnia. .....96

Figura 7.3 – Análise do Tempo de Varredura por Transação gasto pelos Algoritmos. .....99

Figura 7.4 – Tempo Total de Execução dos Algoritmos Variando o *minsup.* .....100

Figura 7.5 – Redução de Regras (%) no Algoritmo FOntGAR usando a Segunda Base. .....101

Figura 7.6 – Variação da Quantidade de Regras de Acordo com o Parâmetro *minGen.*.....102

Figura 7.7 – Parte da ontologia difusa FOnt-2.....103

# LISTA DE TABELAS

Tabela 2.1 - Notação *Apriori*.....24
Tabela 2.2.2 – Relação de itens comprados por transação .....28
Tabela 5.1 – Exemplo de dados e atributos quantitativos. .....60
Tabela 5.2 – Mapeamento de dados e atributos quantitativos. .....61
Tabela 6.1 – Exemplo Simples de Base Transacional. .....80
Tabela 6.2 – Representação dos Graus entre Folhas e Ancestrais. .....80
Tabela 7.1 – Regras Tradicionais Incluindo Associações de Similaridade por Sabor .....102
Tabela 7.2 - Regras Tradicionais Incluindo Associações de Similaridade por Aparência .....103
Tabela 7.3 – Exemplo de Regras Tradicionais Geradas .....104
Tabela 7.4 – Exemplos de Regras Generalizadas no Nível 1 .....104
Tabela 7.5 – Exemplos de Regras Generalizadas no Nível 2 .....105

# LISTA DE ABREVIATURAS E SIGLAS

**FOntGAR** – Fuzzy Ontology-Based Generalized Association Rules Algorithm

**GARPA** – Generalized Association Rule Post-Processing Approach

**IBGE** – Instituto Brasileiro de Geografia e Estatística

**KDD** – Knowledge Discovery from Data

**MD** – Mineração de Dados

**NARFO** – Non-redundant and Generalized Association Rule based on Fuzzy Ontologies

**OWL** – Web Ontology Language

**SSDM** – Semantically Similar Data Mining

**UFOCoRe** - The Upper Fuzzy Ontology With Context Representation

**XSSDM** – Extended Semantically Similar Data Mining

# SUMÁRIO

**CAPÍTULO 1 - INTRODUÇÃO ..... 14**

1.1 Contexto ..... 14
1.2 Motivação e Objetivos ..... 15
1.3 Organização do Trabalho ..... 17

**CAPÍTULO 2 - MINERAÇÃO DE DADOS ..... 18**

2.1 Considerações Iniciais ..... 18
2.2 Mineração de Dados e o KDD ..... 19
2.3 Regras de Associação ..... 22
2.4 Algoritmo *Apriori* ..... 24
2.5 Regras de Associação Generalizadas ..... 26
2.6 Redundância e Medidas de Interesse ..... 32
2.7 Considerações Finais ..... 34

**CAPÍTULO 3 - LÓGICA DIFUSA ..... 36**

3.1 Considerações Iniciais ..... 36
3.2 Conjuntos Clássicos ..... 37
3.3 Conjuntos Difusos ..... 38
3.4 Relações Difusas ..... 40
3.5 Considerações Finais ..... 42

**CAPÍTULO 4 - ONTOLOGIAS ..... 43**

4.1 Considerações Iniciais ..... 43
4.2 Conceitos e Definições Básicas ..... 44
4.3 Tipos, Projetos e Linguagens de Ontologias ..... 47
4.4 Ontologias Difusas ..... 51
4.5 Abordagens de Ontologias Difusas ..... 52
4.6 Considerações Finais ..... 58

**CAPÍTULO 5 - LÓGICA DIFUSA NA TAREFA DE ASSOCIAÇÃO ..... 59**

5.1 Considerações Iniciais ..... 59
5.2 Lógica Difusa em Regras de Associação Quantitativas ..... 60

5.3 Lógica Difusa em Regras de Associação Generalizadas ..... 63
5.4 Considerações Finais ..... 69

**CAPÍTULO 6 - TRABALHOS DESENVOLVIDOS ..... 70**

6.1 Considerações Iniciais ..... 70
6.2 Algoritmo FOntGAR para Mineração de Regras de Associação Generalizadas Utilizando Ontologias de Conceitos Fuzzy e Similaridade Baseada em contexto ..... 71
6.2.1 Ideias Principais ..... 71
6.2.2 Abordagem de Agrupamento Proposta ..... 74
6.2.3 O Algoritmo Passo a Passo ..... 79
6.2.4 Calculando Graus de Suporte e Confiança Estendidos no Pós-Processamento ..... 86
6.2.5 Utilizando Similaridade Baseada em Contexto nas Regras ..... 91
6.3 Considerações Finais ..... 93

**CAPÍTULO 7 - EXPERIMENTOS ..... 94**

7.1 Considerações Iniciais ..... 94
7.2 Experimentos Realizados ..... 95
7.3 Considerações Finais ..... 105

**CAPÍTULO 8 - CONCLUSÕES ..... 106**

8.1 Resultados Obtidos ..... 106
8.2 Contribuições ..... 107
8.2.1 Publicações ..... 107
8.3 Limitações ..... 108
8.4 Trabalhos Futuros ..... 109
8.4.1 Realização de mais testes ..... 109
8.4.2 Desenvolvimento de interface gráfica ..... 110
8.4.3 Automatização de parâmetros providos pelo usuário ..... 110

**REFERÊNCIAS ..... 111**

# Capítulo 1

# INTRODUÇÃO

## 1.1 Contexto

De um modo geral, cada vez mais as pessoas e as corporações têm armazenado dados, tais como atividades diárias, informações sobre clientes, fornecedores ou transações comerciais. Esse aumento é oriundo da atual facilidade e redução dos custos para manter essas informações. Entretanto, o grande acúmulo de dados dificulta a análise dos mesmos, tornando inviável qualquer processo de descoberta de informações relevantes.

Durante anos, métodos predominantemente manuais foram utilizados para transformar dados brutos em conhecimento. Porém, quando aplicado a grandes bases de dados, o uso desses métodos tornou-se dispendioso (em termos financeiros e de tempo), subjetivo e até mesmo inviável. Dentro desse contexto, surgiu a necessidade do desenvolvimento de processos de análise computacional, como a mineração de dados.

A mineração de dados é uma área que atrai a atenção de muitos pesquisadores, sendo a associação uma das técnicas com grande destaque no assunto. A tarefa de associação é realizada por meio de algoritmos, que geram regras que caracterizam o quanto a presença de um conjunto de itens, nos registros de uma base de dados, implica na presença de outro conjunto distinto de itens, nos mesmos registros (AGRAWAL; SRIKANT, 1994). Portanto, trata-se de implicações do tipo antecedente/consequente.

Além de serem muitos, os padrões gerados por algoritmos tradicionais de associação são muito específicos, e retratam o domínio em sua forma mais especializada. Nesse sentido, seria mais interessante apresentar ao usuário padrões em nível de generalização superior, e em menor quantidade. Dentro desse contexto, a obtenção de padrões gerais deve ser assistida pelo uso de conhecimento de domínio, que pode se representado, por exemplo, via taxonomias. Por outro lado, além dos pontos destacados, também seria conveniente considerar a exploração de enriquecimento semântico nas regras.

Sendo assim, sabendo que ontologias podem representar um domínio de forma taxonômica, possuem formalismo bem definido, e grande poder de inferências, é possível dizer que elas são uma alternativa muito interessante em mineração de dados. Ontologias tradicionais são baseadas na lógica clássica, que considera a discriminação do raciocínio em *verdadeiro* ou *falso*. Elas são muito utilizadas em mineração, porém, a restrição imposta pela lógica tradicional (*verdadeiro* ou *falso*) torna esse tipo de estrutura inapropriada para representar conceitos nebulosos, tais como *novo, velho, quente, frio*, ou relacionamentos imprecisos, como o de similaridade, por exemplo.

Conceitos e relacionamentos imprecisos são melhores representados pela lógica fuzzy, ou lógica difusa, que é baseada na teoria dos conjuntos difusos (ZADEH, 1965). De modo geral, a teoria dos conjuntos difusos permite que um elemento pertença a um determinado conjunto com um grau entre 0 e 1. Sendo assim, ontologias difusas permitem que seja possível representar domínios imprecisos, proporcionando grande potencial de inferências, além de permitirem a extração de regras com maior enriquecimento semântico.

## 1.2 Motivação e Objetivos

A crescente disponibilidade de dados tem facilitado o acesso à informação pertencente a diversas áreas do conhecimento. Diante da oferta abundante de dados, cada vez mais são necessárias técnicas eficazes para encontrar informações relevantes em meio a eles. Com base nisso, conforme mencionado na seção 1.1, seria mais interessante que os algoritmos de mineração gerassem menos padrões,

facilitando processo de análise dos mesmos. Nesse sentido, a mineração de regras generalizadas, introduzida em (SRIKANT; AGRAWAL, 1995), surge como uma alternativa muito relevante, pois permite que a quantidade de padrões extraídos seja reduzida, facilitando a compreensão dos mesmos. Entretanto, em muitas situações do mundo real o domínio explorado pode ser *fuzzy* e não *crisp*.

Para lidar com domínios nebulosos, a maioria das abordagens explora regras generalizadas fuzzy, utilizando termos linguísticos nas regras. Nesses trabalhos, os dados utilizados são do tipo categórico ou quantitativo, as estruturas taxonômicas são, em geral, crisp, e os termos linguísticos estão sempre associados à questão de intervalos difusos. Por outro lado, poucos trabalhos foram desenvolvidos para minerar regras generalizadas utilizando taxonomias de conceitos fuzzy, que são estruturas compostas por graus de especialização/generalização variando no intervalo [0,1]. Além disso, essas pesquisas utilizam somente o conceito de transações estendidas, e a obtenção de regras generalizadas ocorre durante a etapa inicial de extração de padrões.

O conceito de transações estendidas foi introduzido em (SRIKANT; AGRAWAL, 1995), e o maior problema em relação ao uso das mesmas, é que elas ocasionam a geração de muitos itemsets candidatos, implicando no aumento do montante final de regras extraídas. Além disso, elas acabam possibilitando a geração de regras generalizadas, e tradicionais com relação hierárquica em comum, ocasionando problema de redundância. Sendo assim, além dos inconvenientes apresentados, também é necessário que se preocupe com a questão da poda de padrões redundantes.

Sendo assim, o objetivo deste trabalho é a especificação e implementação do algoritmo FOntGAR (***F****uzzy* ***On****tology-Based* ***G****eneralized* ***A****ssociation* ***R****ules Algorithm*), para mineração de regras de associação com itens presentes em qualquer nível de ontologias difusas, a fim de se obter resultados não redundantes, semânticos, e em menor quantidade. Diferente de outros trabalhos existentes na literatura, FOntGAR é o primeiro algoritmo desenvolvido para realizar a generalização, utilizando estruturas taxonômicas compostas por graus de especialização/generalização variando em [0,1], durante etapa de pós-processamento. Esse diferencial é muito importante, pois além de permitir que a generalização reduza a quantidade de regras extraídas, o pós-processamento

também impede a ocorrência de padrões redundantes, eliminando a necessidade de medidas utilizadas para podar esse problema.

O algoritmo FOntGAR é capaz de gerar regras compostas por associações de similaridade, contribuindo com a questão de enriquecimento semântico. Por conseguinte, além de diferentes contextos, também são exploradas inferências nas relações mencionadas, aproveitando a propriedade transitiva das mesmas. FOntGAR possui, ainda, o diferencial de não restringir que a relação “itens da base/folhas da ontologia” seja total, portanto, nem todos os itens da base precisam estar representados na estrutura, e vice-versa.

## 1.3 Organização do Trabalho

No Capítulo 2 serão apresentados diversos conceitos envolvendo a descoberta de conhecimento em bancos de dados, tais como mineração de dados, tarefas de mineração de dados, regras de associação, regras de associação generalizadas, redundância em regras de associação e algoritmos de mineração de regras de associação.

No Capítulo 3 serão apresentados alguns conceitos da teoria dos conjuntos clássico, da lógica difusa e da teoria dos conjuntos difusos. Serão apresentadas também a definição de relações difusas e algumas de suas propriedades.

O Capítulo 4 traz a definição de ontologias e ontologias difusas, apresentando também os principais conceitos envolvidos nesse assunto, tais como tipos, projetos e linguagens de ontologias e algumas abordagens de ontologias difusas.

No Capítulo 5 serão apresentados alguns conceitos e trabalhos sobre lógica difusa na mineração de dados. O Capítulo 6 apresenta a proposta de trabalho do mestrado. No Capítulo 7 será apresentado os experimentos realizados, e o Capítulo 8 apresenta a conclusão e as considerações relevantes para trabalhos futuros.

# Capítulo 2

# MINERAÇÃO DE DADOS

## 2.1 Considerações Iniciais

Segundo Elmasri e Navathe (2005), “Mineração de Dados” se refere à descoberta de novas informações em função de padrões ou regras em grandes quantidades de dados. Resumidamente, é possível afirmar que a mesma provê um método computacional para descobrir padrões, sem a tendenciosidade e a limitação de uma análise baseada meramente na intuição humana (BRAGA, 2005).

Sendo assim, o propósito deste capítulo é esclarecer o que é mineração de dados, bem como suas principais tarefas e conceitos, destacando-se a tarefa de associação, que é foco deste trabalho. Neste capítulo também será definido o que são regras de associação generalizadas, e os principais conceitos envolvidos.

No presente trabalho adota-se o conceito de que a mineração de dados é uma das etapas do processo de descoberta de conhecimento em banco de dados, conhecido como KDD. Sendo assim, como se trata de um estudo sobre algoritmos de mineração, que envolve também a compreensão de conceitos relacionados a esse processo, também será apresentada uma visão geral do mesmo.

O presente capítulo está organizado da seguinte maneira:

- Seção 2.2: conceitos básicos sobre KDD, mineração de dados e tarefas de mineração de dados;
- Seção 2.3: definição de regras de associação e os principais conceitos envolvidos. Descreve o que é um algoritmo de mineração e apresentas algumas definições envolvidas em tais algoritmos;

- Seção 2.4: algoritmo *Apriori*;
- Seção 2.5: definição de regras de associação generalizadas e seus principais conceitos;
- Seção 2.7: resumo do conteúdo apresentado.

## 2.2 Mineração de Dados e o KDD

Os sistemas de bancos de dados se evoluíram em virtude da expansão de negócios e da formação de grandes corporações (BRAGA, 2005). Essa evolução se iniciou na década de 1960 com os primitivos sistemas de processamento de arquivos, que atualmente dão lugares a sofisticados sistemas gerenciadores, mais evoluídos em funcionalidades como coleta de dados, gerenciamento (armazenamento e recuperação) e processamento de transações (HAN; KAMBER, 2006). Nesse sentido, o progresso da tecnologia de armazenamento proporcionou o crescimento de dados armazenados, implicando em bases cada vez maiores. Esse crescimento ocorreu em todas as áreas de atividade humana, tais como transações de supermercado, registros de usos de cartões de crédito, detalhes de chamadas telefônicas e estatísticas governamentais (HAND; MANNILA, 2001).

Dessa forma, o conceito de *Knowledge Discovery from Data* (KDD), ou Descoberta de Conhecimento em Banco de Dados surge da preocupação desses seguimentos em conseguir extrair padrões em meio as grandes quantidades de dados armazenados em suas bases. Devido à complexidade envolvida no KDD, é fundamental que se entenda como ele realmente funciona (BRACHMAN; ANAND, 1994).

KDD é o processo não trivial de extração de padrões válidos, novos, potencialmente úteis e compreensíveis a partir de dados (FRAWLEY, et al., 1992), ou seja, sua finalidade é fazer com que dados em baixo nível tornem-se conhecimento em alto nível, por exemplo, transformando-os em relatórios compactos ou então em modelos que permitam estimar casos futuros (FAYYAD, et al., 1996b).

Um ponto muito importante a ser ressaltado, é que muitos pesquisadores consideram KDD sinônimo de mineração de dados. Por outro lado, outros definem KDD como um processo, no qual a mineração de dados é uma de suas etapas

(HAN; KAMBER, 2006). Em Fayyad, *et al.* (1996a), por exemplo, foi adotada a segunda visão. Para os autores a mineração de dados é uma fase, situada dentro de um processo interativo e iterativo, onde são definidos e aplicados algoritmos específicos para extração de padrões.

A Figura 2.1 ilustra o processo, evidenciando o pré-processamento e a transformação como etapas antecedentes à mineração de dados. Segundo os autores, essas etapas antecedentes são essenciais, pois englobam, por exemplo, questões como a compreensão do domínio, o tratamento dos dados e a escolha do algoritmo de mineração de dados. A partir de dados transformados, a fase de mineração irá gerar padrões compactos, que serão utilizados na fase de interpretação como conteúdo para descoberta de novos conhecimentos.

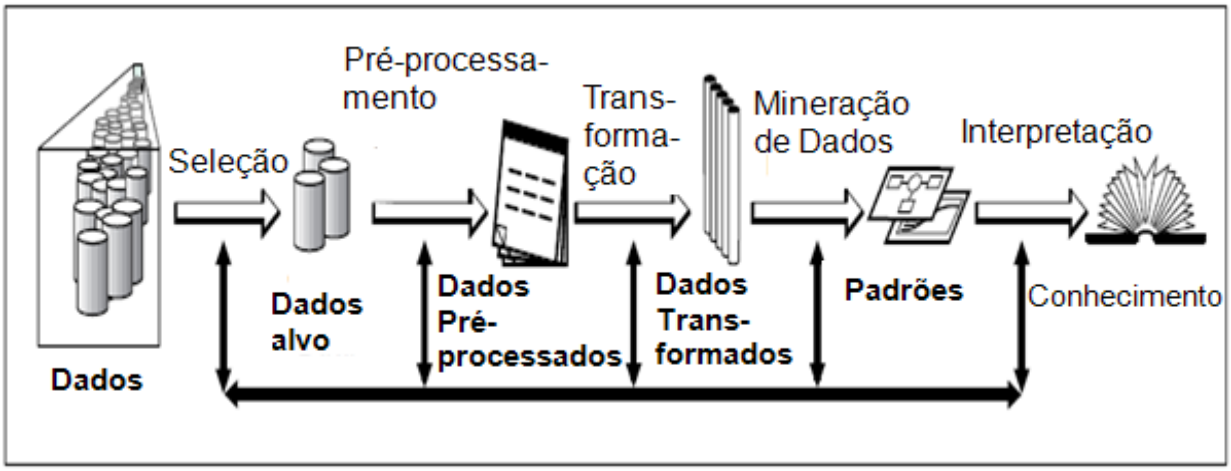


**Figura 2.1 - Visão Geral dos Passos do Processo KDD – adaptada de (FAYYAD, *et al.*, 1996b)**

As etapas ilustradas podem ser sumarizadas em nove passos principais:

1. **Compreensão do Domínio da Aplicação** (compreensão do domínio da aplicação, e identificação dos objetivos do processo KDD do ponto de vista do cliente);
2. **Criação do conjunto de dados destino** (inclui a seleção do conjunto de dados onde será realizada a extração de conhecimento);
3. **Limpeza dos dados e pré-processamento** (inclui operações básicas como remoção de ruídos e inconsistências);
4. **Redução dos dados e projeção** (inclui encontrar recursos úteis para representar os dados, utilizando métodos que reduzam o número efetivo de variáveis em consideração);

5. **Escolha da Função de mineração de dados** (inclui a escolha do método de mineração de dados (por exemplo, classificação, sumarização, etc.) correspondente com os objetivos do processo KDD);
6. **Escolha do algoritmo de mineração** (inclui escolher o algoritmo e selecionar os métodos pra encontrar padrões em dados);
7. **Mineração de Dados** (procura por padrões de interesse em uma forma representacional particular ou um conjunto delas, incluindo regras de classificação, árvores, regressão e agrupamento);
8. **Interpretação** (inclui a interpretação e visualização dos padrões descobertos, possivelmente retornando a algum dos passos anteriores, removendo redundâncias e padrões irrelevantes);
9. **Utilização do conhecimento descoberto** (inclui a incorporação desse conhecimento no desempenho do sistema, ou simplesmente documentá-lo e reportá-lo as partes interessadas).

O conhecimento gerado pelo KDD depende do tipo de informação que se deseja obter, e está diretamente relacionado à etapa de mineração de dados (FAYYAD, *et al.*, 1996). Sendo assim, é necessário que já no início do processo seja decidido qual o tipo de conhecimento que deverá ser extraído, de forma que a tarefa de mineração e o algoritmo necessário sejam adequadamente escolhidos na etapa de mineração.

De acordo com Han e Kamber (2006) as possíveis tarefas de um algoritmo de extração de padrões podem ser agrupadas em atividades descritivas e preditivas:

- **Tarefas Descritivas** caracterizam as propriedades gerais dos dados no banco de dados, elas objetivam derivar padrões de conhecimento que resumam os relacionamentos subjacentes dos dados, tais como, por exemplo, tendências, agrupamentos, trajetórias;
- **Tarefas Preditivas** permitem fazer inferência sobre os dados presentes, possibilitando que seja feita predições sobre os dados futuros. O dado a ser previsto, é comumente conhecido como variável dependente ou alvo, ao passo que os dados utilizados na previsão, são chamados de variáveis independentes ou explicativas.

Existem diversas *tarefas* de mineração de dados, cada uma delas é utilizada de acordo com a natureza dos dados e objetivos da aplicação. Dentre essas tarefas,

as mais conhecidas são: *classificação, associação, agrupamento e regressão.* As tarefas de *classificação* e *regressão* são consideradas preditivas, e as tarefas de *associação* e *agrupamento* são descritivas. A Figura 2.2 ilustra essa divisão.

Conforme já mencionado, nas tarefas de predição o atributo a ser previsto é chamado de variável alvo. Como as tarefas preditivas podem ser subdivididas em tarefas de classificação e de regressão, pode-se dizer que a classificação é utilizada para obtenção de variáveis discretas, e a regressão para obtenção de variáveis contínuas. A tarefa de agrupamento, ou análise de grupo, é utilizada para descobrir grupos de observações relacionadas, de modo que as observações de um mesmo grupo sejam semelhantes, comparativamente as pertencentes a outros grupos (MAGALHÃES JUNIOR, 2010). Tarefas de associação consistem na descoberta de regras de associação, e seus conceitos serão apresentados na seção 2.3.

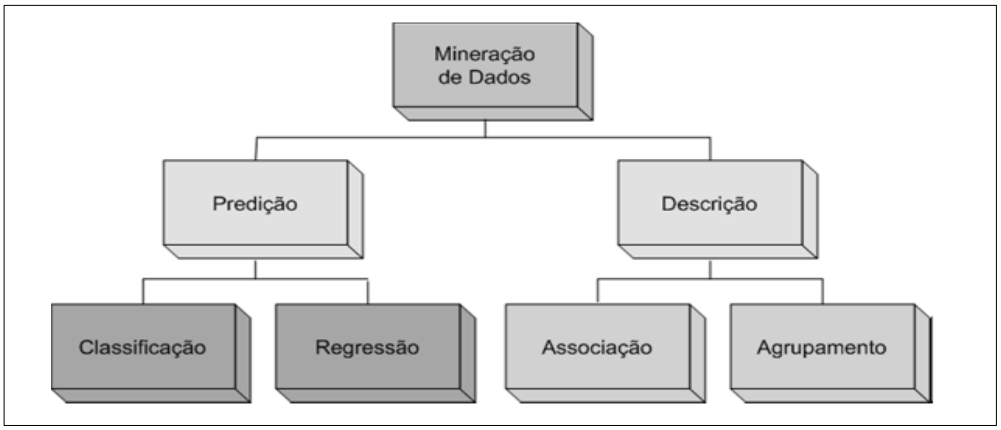


**Figura 2.2 – Principais Tarefas de Mineração de Dados.**

## 2.3 Regras de Associação

Na tarefa de associação, o banco de dados é tratado como uma coleção de transações, sendo cada uma delas um conjunto de itens (ELMASRI; NAVATHE, 2005). A mineração de regras de associação foi proposta inicialmente por Agrawal, *et al.*, (1993). Segundo eles, uma regra de associação é uma implicação *antecedente* → *consequente*, na qual antecedente e consequente são partes disjuntas compostas por um ou mais elementos da base de dados.

Considerando I = (*i1, i2,....,in*) um conjunto de itens de uma base de dados B, uma regra de associação é uma implicação $X \rightarrow Y$, onde $X \subset I$, $Y \subset I$, $X \cap Y = \emptyset$,

sendo que X e Y representam, respectivamente, o lado antecedente e consequente da regra (AGRAWAL, et al., 1993).

Dois conceitos muito importantes relacionados às regras de associação são o de *suporte*, e *confiança*. O suporte de uma regra é o percentual de transações da base de dados em que antecedente e consequente da regra aparecem conjuntamente na mesma transação. Esse percentual pode ser obtido através da seguinte equação:

$$Suporte\ (X \rightarrow Y) = \frac{Total\ de\ Transações\ com\ Ocorrências\ (X \cup Y)}{Total\ de\ Transações}$$

A confiança de uma regra é o percentual de transações, dentre as que possuem o antecedente da regra, em que antecedente e consequente aparecem conjuntamente na mesma transação. Essa medida pode ser obtida através da equação:

$$Confiança\ (X \rightarrow Y) = \frac{Total\ de\ Transações\ com\ Ocorrências\ (X \cup Y)}{Total\ de\ Transações\ com\ Ocorrências\ X}$$

Como exemplo, poderíamos obter uma regra *{alface → presunto, queijo},* com 5% de grau de suporte e 60% de grau de confiança. Isso significa que em 5% de todas as transações da base de dados os itens *alface, presunto e queijo,* aparecem juntos na mesma transação, e em 60% das vezes que *alface* apareceu em uma transação, *presunto e queijo* também apareceu.

Sendo assim, algoritmos de regras de associação trabalham com duas medidas pré-estabelecidas, o suporte mínimo (*minsup)*, que pode ser utilizado para determinar se um conjunto de itens (*itemset*) da base de dados é frequente, e a confiança mínima (*minconf),* que pode ser utilizada para determinar se uma regra gera é valida ou não. Um *itemset* é frequente apenas se seu valor de suporte é maior ou igual ao *minsup*.

Com base nesses conceitos, pode-se dizer que o objetivo de algoritmos tradicionais de associação é gerar regras válidas a partir de um conjunto de *itemsets* frequentes, da seguinte maneira:

1. Encontrar todos os *itemsets* frequentes;
2. Gerar regras de associação válidas a partir dos *itemsets* frequentes.

Agrawal e Srikant (1994) foram os primeiros a apresentar um algoritmo de mineração de regras de associação, denominado *Apriori*. Desde então, diversos

algoritmos foram e ainda continuam sendo desenvolvidos. Considerando que o algoritmo mencionado está inserido no projeto desenvolvido por este trabalho de mestrado, na seção 2.4 será apresentado um detalhamento do mesmo.

## 2.4 Algoritmo *Apriori*

*Apriori* é um algoritmo clássico na mineração de regras de associação. A partir de uma base de dados de transações, ele identifica todos os conjuntos de *itemsets* frequentes e gera regras de associação a partir dos mesmos. É possível resumir o objetivo do *Apriori* da seguinte maneira:

Dados:

- Um conjunto de transações *D, D* = $\{T | T \text{ é } um\ conjunto\ de\ itens\}$;
- Um suporte mínimo *minsup;*
- Uma confiança mínima *minconf;*

Obtenha todas as regras de associação que possuam

- Confiança ≥ *minconf*;
- Suporte ≥ *minsup.*

O algoritmo utiliza algumas notações específicas, que são apresentadas na Tabela 2.1.

**Tabela 2.1 - Notação *Apriori*.**

| | |
|---|---|
| $L_k$ | Conjunto de *large itemset* de tamanho *k* que possuem suporte mínimo, ou seja, são os *itemsets* frequentes. |
| $C_k$ | Conjunto de *itemsets* candidatos de tamanho *k* |
| *k-itemset* | O tamanho de um *itemset* é o número de itens que o mesmo possui. Portanto, trata-se de um *itemset* com *k* itens. |

*Apriori* utiliza duas funções: função *apriori-gen*, responsável por gerar os candidatos, e a função *ap-genrules*, para geração das regras. Além disso, ele pode ser dividido em quatro passos principais:

1. Contagem inicial da ocorrência dos itens da base de dados para determinar os *1-itemsets* frequentes;

2. Geração do conjunto de *itemsets* candidatos $C_k$ (função *apriori-gen*) e contagem do suporte dos candidatos em $C_k$;
3. Geração do conjunto de *itemsets* frequentes $L_k$ (*large itemsets*) a partir dos candidatos $C_k$;
4. Geração das regras a partir dos *itemsets* frequentes (função *ap-genrules*).

Primeiramente, ele realiza uma varredura completa na base de dados, identificando todos os itens, e registrando a quantidade de ocorrências de cada um deles. Os itens que possuírem ocorrências superiores ou iguais ao *minsup* são chamados de itens frequentes, passando a integrar o conjunto $L_1$ de *1-itemsets* frequentes. A função *apriori-gen*, recebe como parâmetro um conjunto $L_k$ e retorna um conjunto de candidatos $C_k$. Ela é dividida em dois passos: *Join* (Junção) e *Prune* (Poda).

No passo *Join*, ocorrem junções de $L_{k-1}$ com $L_{k-1}$, ou seja, são realizadas todas as combinações possíveis entre os elementos do conjunto $L_{k-1}$. Essas junções são efetuadas para gerar os novos *itemsets* de tamanho *k* que serão incluídos em $C_k$. Por exemplo, supondo $L_1 = \{\{A\},\{B\},\{C\}\}$, $L_1 * L_1 = C_2 = \{\{AB\},\{AC\},\{BC\}\}$.

No passo *Prune*, a função *apriori-gen* exclui todos os *itemsets* "***c***" do conjunto $C_k$, que possuem subconjuntos "***s***" de tamanho *k-1*, não frequentes. Para isso, ela elimina todos os *itemsets* "***c***" do conjunto $C_k$ que possuem algum subconjunto de tamanho *k-1* que não pertença ao conjunto $L_{k-1}$.

Por exemplo, dado $L_3 = \{\{ABC\},\{ABD\},\{ACD\},\{ACE\},\{BCD\}\}$, após a fase de junção $C_4$ será $\{\{ABCD\},\{ACDE\}\}$. A fase de poda irá excluir o *itemset* $\{ACDE\}$, pois subconjunto $\{ADE\}$ do mesmo não pertence a $L_3$, portanto $C_4$ ficará somente com o *itemset* $\{ABCD\}$.

Com a geração de $C_k$ concluída, o algoritmo faz uma varredura em toda a base de dados para a contagem do suporte dos *itemsets* desse conjunto. Feita a contagem, ele elimina todos os *itemsets* que não possuem suporte maior ou igual ao *minsup*, gerando um novo $L_k$ de frequentes. Essa rotina continua sucessivamente até que *k itemsets* frequentes não possam mais ser encontrados, ou seja, o processo irá terminar quando um novo $L_k$ não puder ser gerado.

Ao final, a partir do conjunto de *itemsets* frequentes todas as regras são geradas, e aquelas que possuem suporte e confiança maiores ou iguais a *minsup* e

*minconf* são mostradas ao usuário. Um exemplo interessante pode ser encontrado em Pasquier, *et al.*, (1999).

## 2.5 Regras de Associação Generalizadas

Algoritmos tradicionais de regras de associação geram todas as regras possíveis compostas apenas por itens da base de dados. Dessa forma, por incluírem apenas tais itens, as regras não fornecem um conhecimento generalizado relacionado ao conjunto de dados, podendo dificultar o processo de análise das mesmas (CARVALHO; REZENDE; CASTRO 2007).

Nesse sentido, objetivando resolver essa questão, o uso de taxonomias foi introduzido na tarefa de associação. Taxonomias são estruturas com aplicação em diversas áreas do conhecimento. Em geral, elas refletem como um conjunto de itens pode ser organizado hierarquicamente.

A Figura 2.3 ilustra um pequeno exemplo de uma taxonomia, onde se pode verificar que jaqueta é um agasalho, calça é um agasalho, sapato é um calçado, botas é um calçado, agasalho é uma roupa e camisa é uma roupa.

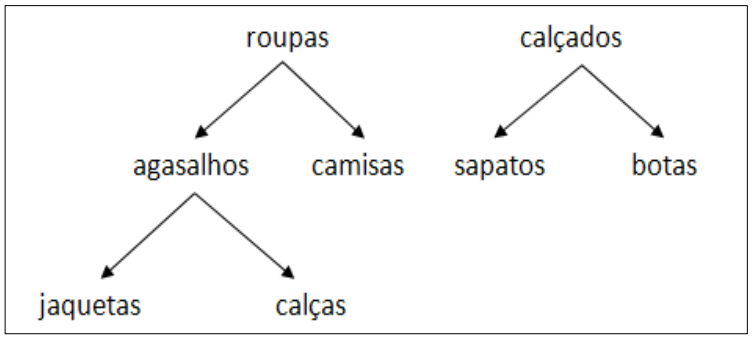


**Figura 2.3 - Exemplo de Taxonomia – adaptada de (Srikant e Agrawal, 1995).**

Srikant e Agrawal (1995) apresentam algumas razões que torna interessante o uso de taxonomias em regras de associação:

- Regras tradicionais podem não ter suporte suficiente para serem incluídas na solução, mas podem representar conhecimento interessante aos serem agrupadas segundo uma taxonomia;
- Mesmo considerando as regras específicas com altos valores de suporte e confiança, as mesmas podem ser agrupadas em regras

mais gerais, melhorando sua compreensão e ainda podendo aumentar os valores de suporte e confiança;

- O uso de informações contidas nas taxonomias pode proporcionar a identificação de regras mais interessantes.

Sendo assim, dado um banco de dados transacional e uma taxonomia de itens, o problema de minerar regras generalizadas consiste em encontrar regras de associação com itens presentes em qualquer nível dessa taxonomia (SRIKANT; AGRAWAL, 1995).

Sendo $\mathrm{I} = \{i_1, i_2, \ldots, i_n\}$ um conjunto de itens da base de dados e *T* uma taxonomia, uma aresta em *T* representa um relacionamento do tipo "*é um*". Portanto, se existe uma aresta em *T* de *p* para *c*, então *p* é chamado *pai* de *c* e *c* é chamado *filho* de *p* (*p* é a generalização de *c*).

Dessa forma, regras generalizadas são implicações da forma $X \rightarrow Y$, $X \subset Y$, $Y \subset I$, $X \cap Y = \emptyset$, onde nenhum item em Y é *ancestral* de nenhum item em X (SRIKANT; AGRAWAL, 1995). O motivo para nenhum item em Y ser *ancestral* de nenhum item em X, é que a regra "x → *ancestral de x*" é trivialmente verdadeira, com 100% de grau de confiança.

Considere a taxonomia ilustrada na Figura 2.3. A partir da mesma, "*jaquetas → camisas"*, *"agasalhos → sapatos"*, ou então, *"agasalhos → camisas"*, são exemplos de regras de associação generalizadas, porém, a regra "*jaquetas → agasalhos, camisas"*, por exemplo, não pode ser incluída nesse contexto, pois o item *agasalhos* é *pai* de jaqueta. É importante salientar que os nós folhas da árvore representam os itens do banco de dados.

Assim como no caso tradicional, apresentado na seção 2.3, os algoritmos de regras generalizadas também utilizam as medidas de suporte e confiança durante o processo de geração dos padrões. Nesse caso, embora o suporte para um item terminal na taxonomia seja definido conforme apresentado na seção 2.3, o suporte para um item *y* não terminal na taxonomia é definido como: $\sup(y) = \left|\cup\, t\left(desc(y)\right)\right| / n$, onde $desc(y)$ representa o conjunto de descendentes de y, |X| a cardinalidade de X, t(X) as transações em que X ocorre e *n* o número de transações. No Exemplo 1, é apresentado como é calculado o suporte de itens terminais e não terminais da taxonomia.

**Exemplo 1** *Considerando a taxonomia apresentada na Figura 2.3 e uma base de dados que contém um conjunto de itens A = {jaquetas, calças, sapatos, botas} e um conjunto de transações T = {1, 2, 3, 4, 5, 6} no qual a relação de itens comprados por cada transação é apresentada na Tabela 2.2.*

**Tabela 2.2.2 – Relação de itens comprados por transação**

| **Transações** | ***Itens comprados*** |
|---|---|
| 1 | *jaquetas, calças, botas* |
| 2 | *sapatos, botas, jaquetas, calças* |
| 3 | *sapatos, jaquetas, camisas* |
| 4 | *calças, botas* |
| 5 | *sapatos, botas, jaquetas* |
| 6 | *botas, jaquetas, camisas* |

Com base nessas informações o suporte dos itens terminais e não terminais são calculados como a seguir:

- *Terminais:*

*Sup*({jaquetas}) = |*t{jaquetas}*|/6 = |{1, 3, 5, 6}|/6 = 4/6 = 0,66 ou 66%

*Sup*({calças}) = |*t{calças}*|/6 = |{1, 2, 4}|/6 = 3/6 = 0,50 ou 50%

*Sup*({botas}) = |*t{botas}*|/6 = |{1, 2, 4, 5, 6}| = 5/6 = 0,83 ou 83%

*Sup*({sapatos}) = |*t{sapatos}*| = |{2, 3, 5}| = 3/6 = 0,50 ou 50%

*Sup*({camisas}) = |*t{camisas}*| = |{3, 6}| = 2/6 = 0,33 ou 33%

- *Não Terminais:*

*Sup({agasalhos})* = |*t{jaquetas}* ∪ *t{calças}*|/6 = |{1, 3, 5, 6} ∪ {1, 2, 4}|/6 =

= |{1, 2, 3, 4, 5, 6}|/6 = 6/6 = 1 ou 100%.

*Sup({roupas})* = |*t{agasalhos}* ∪ *t{camisas}*|/6 =

= |{1, 2, 3, 4, 5, 6} ∪ {3, 6}|/6 = |{1, 2, 3, 4, 5, 6}|/6 = 6/6 = 100%

*Sup({calçados})* = |*t{sapatos}* ∪ *t{botas}*|/6 = |{2, 3, 5} ∪ {1, 2, 4, 5, 6}|/6 =

= |{1, 2, 3, 4, 5, 6}|/6 = 6/6 = 1 ou 100%

O processamento de algoritmos de mineração de regras generalizadas pode ser subdividido em etapas ou fases, até que o conjunto final de padrões seja apresentado ao usuário. Dessa forma, a literatura mostra que taxonomias são utilizadas, para fins de generalização, basicamente em duas etapas, no pré-processamento ou então no pós-processamento.

O pré-processamento ocorre antes da fase de geração de padrões. Nessa etapa, taxonomias são utilizadas para permitir que as transações da base de dados sejam compostas pelos itens originais e pelos ancestrais desses itens na estrutura. Trata-se de transações estendidas, introduzidas em (SRIKANT; AGRAWAL, 1995).

Sendo assim, a partir das transações mencionadas, algoritmos como o de Srikant e Agrawal (1995) utilizam a etapa de extração para realizar as seguintes atividades:

1. Encontrar todos os *k-itemsets* que possuam suporte maior ou igual ao suporte mínimo especificado pelo usuário. Entretanto, os itens que constituem os *k-itemsets* encontram-se presentes em qualquer nível da hierarquia.
2. Utilizar todos os *k-itemsets* frequentes para gerar as regras de associação.

Portanto, transações estendidas fazem com que o resultado da etapa de extração de padrões seja um conjunto formado por regras tradicionais, e generalizadas. Isso ocorre porque os itemsets candidatos gerados a partir dessas transações podem ser compostos também por ancestrais da taxonomia.

É importante enfatizar que a utilização de transações estendidas acaba colaborando com a geração de muitos itemsets candidatos (superior ao caso tradicional), implicando no aumento da quantidade de regras extraídas. Além disso, essa estratégia faz com que o resultado apresentado ao usuário possa contemplar ocorrências simultâneas de regras generalizadas, e específicas correlatas, causando problemas de redundância. Por exemplo, considerando a Figura 2.3, o algoritmo poderia gerar, ao mesmo tempo, as regras "*jaquetas → sapatos*" e "*agasalhos → calçados*", que são redundantes. Regras redundantes não trazem nenhum conhecimento adicional ao usuário, pois proporcionam conhecimento com significado semelhante ou idêntico ao de outras. Dessa forma, na tentativa de reduzir o volume de regras gerado e eliminar padrões desinteressantes, é necessário que após a extração de padrões seja realizada uma etapa onde são aplicadas medidas de interesse para podar redundâncias. P

Por outro lado, taxonomias também são utilizadas em fase de pós-processamento, após a etapa de extração de padrões. No pós-processamento, a generalização é feita baseando-se no conjunto de regras extraído e na informação

contida na estrutura taxonômica. Algoritmos de pós-processamento distinguem-se, principalmente, por não utilizarem bases de dados com transações estendidas.

Em geral, o pós-processamento trabalha em um conjunto de regras substituindo padrões especializados por outros mais gerais, de modo que seja possível obter um conjunto mais reduzido. Adomavicius e Tuzhilin (2001), por exemplo, propõem uma abordagem muito interessante, na qual regras tradicionais geradas são agrupadas em função da similaridade existente entre as mesmas, formando vários grupos de regras idênticas. A partir disso, apenas uma regra de cada grupo é inserida no resultado final. A Figura 2.4 ilustra a ideia proposta.

A similaridade é medida por uma transformação sintática que se faz nas regras, com base na taxonomia do domínio. O processo de agrupamento ocorre basicamente em três etapas. Primeiro, a partir de um conjunto inicial de regras tradicionais, o usuário define em que nível deseja generalizar os itens das regras. Por exemplo, poderia ser definido que os itens A1 e A2 fossem mantidos, e os itens A3, A4 e A5 fossem generalizados para o nó A7 (destacado em cinza).

Após a definição do nível de generalização, cada item dos padrões iniciais é substituído pelo ancestral correspondente ao nível escolhido. Por exemplo, as regras “A2 & A3 → A4” e “A2 & A5 → A3” se tornariam “A2 & A7 → A7” e “A2 & A7 → A7”, respectivamente. É possível que uma regra contenha vários itens que possuam o mesmo ancestral. Nesse caso, havendo itens repetidos após o mapeamento, apenas um permanece na regra mapeada, e os outros são removidos.

Finalmente, após o mapeamento elas são agrupadas em função de sua estrutura sintática, formando-se vários grupos de regras sintaticamente idênticas. No exemplo proposto, seria formado um grupo contendo duas regras, “A2 & A7 → A” e “A2 & A7 → A7”. Após o agrupamento, apenas uma regra de cada grupo formado é inserida no resultado final. Portanto, o objetivo principal do trabalho é a redução do conjunto inicial, proporcionando uma visão mais geral do domínio.

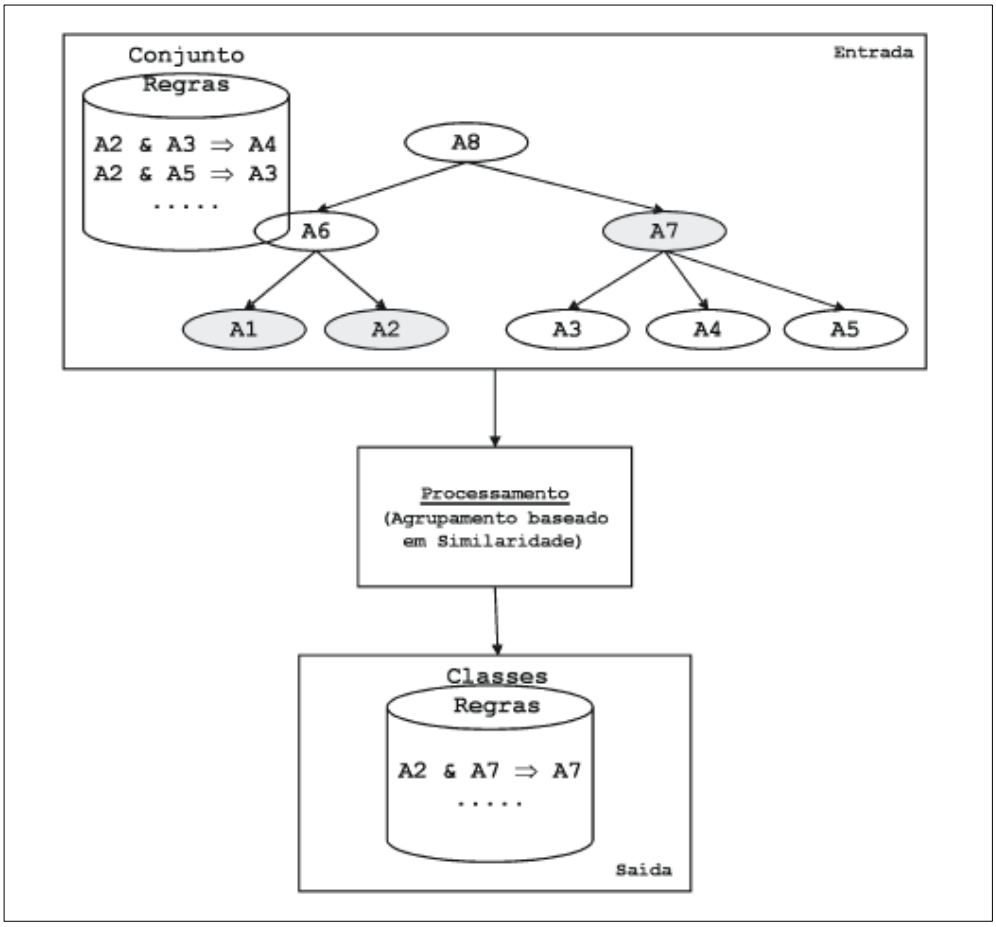


**Figura 2.4 – Abordagem de agrupamento por similaridade.**

O trabalho de Carvalho, Rezende e Castro (2007) segue essa ideia. A abordagem proposta recebeu o nome de GARPA (*Generalized Association Rule Post-Processing Approach*), e consiste basicamente em obter um conjunto final de regras generalizadas a partir de alguns parâmetros de entrada: a base de dados, o conjunto de padrões tradicionais gerado e a taxonomia do domínio.

No algoritmo GARPA, regras generalizadas podem ser obtidas considerando-se todos ou apenas alguns dos itens contidos na taxonomia, ou seja, é possível transformar regras específicas em regras gerais mesmo que um item geral da regra generalizada não represente todos os itens específicos contidos na taxonomia.

Supondo que a regra *leite → pão,* por exemplo*,* represente uma regra generalizada e que leite esteja representado na taxonomia por *leite-a, leite-b, leite-c, leite-d, leite-e*. A regra *leite → pão* irá existir mesmo que não exista uma regra específica para cada tipo de leite. Nesse caso, para evitar generalizações excessivas, um subconjunto de regras específicas só pode ser substituído por uma regra mais geral se o *suporte* ou a *confiança* da regra geral for *t*% maior do que o maior valor da mesma medida em suas regras específicas.

## 2.6 Redundância e Medidas de Interesse

A utilização de taxonomias permite a remoção de regras consideradas desinteressantes ou redundantes. Isso ocorre porque elas possibilitam derivar medidas que podem ser utilizadas para calcular o quão interessante cada regra é. Portanto, pode-se dizer que medidas de interesse são utilizadas para podar regras redundantes, em relação a um conjunto de regras gerado. A noção de "interesse", utilizando taxonomias, é apresentada a seguir.

Considerando que a regra *"agasalhos → camisas"* (suporte = 8%, confiança = 70%) fosse gerada, se agasalhos é pai de jaquetas, e um quarto das vendas de agasalhos é de jaquetas, então poderia ser esperada a presença da regra "*jaquetas → camisas"* (suporte = 2%, confiança = 70%).

Se o suporte e a confiança da regra "*jaquetas → camisas"* são respectivamente, 2% e 70%, a regra poderia ser considerada redundante, pois não apresenta nenhuma informação adicional e é menos geral que a primeira. Portanto, um padrão é interessante se possuir suporte maior ou igual a *R* vezes o valor de um suporte esperado, ou confiança maior ou igual a *R* vezes o valor de uma confiança esperada, para algum valor *R* especificado pelo usuário (Srikant e Agrawal, 1995).

Adamo (2001) apresenta a seguinte definição em relação à noção de interesse, proposta por Srikant e Agrawal (1995):

Seja A = {*a1, a2, ..., an*} um conjunto de itens, *r* uma regra válida e *r*↓ uma regra especializada de *r* . A regra *r*↓ é considerada redundante em relação a *r* (podendo assim ser removida do conjunto final de regras) se *r*↓ possui o mesmo comportamento que *r*. Nesse caso, a sentença "*r*↓ possui o mesmo comportamento que *r"* será substituída pela medida *desvio*.

O *desvio* pode ser definido em relação a duas situações, quando *r*↓ difere de *r* em apenas um item *aj* ∈ A, ou quando *r*↓ difere de *r* em mais de um item *aj* ∈ A. Na primeira situação, o *desvio* é definido como:

$$desvio(r\downarrow, r) = max\left\{\frac{sup(r\downarrow)/sup(r)}{sup(a_j\downarrow)/sup(a_j)}, \frac{conf(r\downarrow)/conf(r)}{sup(a_j\downarrow)/sup(a_j)}\right\}$$

Portanto, *r*↓ será considerada uma regra interessante em relação a *r*, quando $desvio(r\downarrow, r) \geq R$, sendo que $R \geq 1$. O número *R*, especificado pelo usuário, é chamado de interesse mínimo (SRIKANT; AGRAWAL, 1995), ele especifica qual

deve ser o desvio do comportamento da regra *r*↓ para que a mesma possa ser considerada suficientemente diferente de *r*.

Na segunda situação, quando *r*↓ difere de *r* em mais de um item *aj* ∈ A, o *desvio* de *r*↓ em relação à *r* é definido como a seguir:

$$desvio(r\downarrow, r) = max\{sup\_desvio(r\downarrow, r), conf\_desvio(r\downarrow, r)\}$$

Onde

$$sup\_desvio(r\downarrow, r) = \frac{sup(r\downarrow)/sup(r)}{\prod_{j=1}^{m} sup(a_j\downarrow)/sup(a_j)}$$

$$conf\_desv(r\downarrow, r) = \frac{conf(r\downarrow)/conf(r)}{\prod_{j=1}^{m} sup(a_j\downarrow)/sup(a_j)}$$

A seguir, será apresentado um exemplo, extraído de Adamo (2001), no qual é demonstrado como a medida *desvio* é utilizada para encontrar regras interessantes.

**Exemplo 2** *Considere D como uma tabela relacional, e a taxonomia ilustrada na* Figura 2.5*.*

| | **a1** | **a2** | **b** | **A** |
|---|---|---|---|---|
| **1** | 0 | 1 | 1 | 1 |
| **2** | 1 | 0 | 1 | 1 |
| **3** | 0 | 0 | 1 | 0 |
| **4** | 0 | 1 | 0 | 1 |
| **5** | 0 | 0 | 1 | 0 |
| **6** | 0 | 1 | 0 | 1 |

*Então*,

*sup(A) = 4/6* *sup(a1) = 1/6* *sup(a2) = 3/6*

*sup(A → b) = 2/6* *sup(a1 → b) = 1/6* *sup(a2 → b) = 1/6*

*conf(A → b) = 2/4* *conf(a1 → b) = 1* *conf(a2 → b) = 1/3*

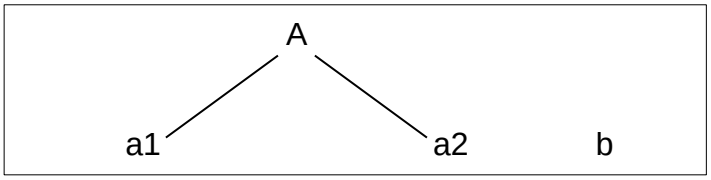


**Figura 2.5 – Taxonomia simples.**

*Portanto*,

*(sup(a1 → b) / sup(A → b)) = 1/2 ≥ (2 . (sup(a1) / sup(A) = 1/4) = 1/2,*

*(conf(a1 → b) / conf(A → b)) = 2 ≥ (2 . (sup(a1) / sup(A) = 1/4) = 1/2,*

*(sup(a2 → b) / sup(A → b)) = 1/2 < (sup(a2) / sup(A)) = 3/4,*

*(conf(a2 → b) / conf(A → b)) = 2/3 < (sup(a2) / sup(A)) = 3/4.*

*A partir das duas primeiras expressões, pode-se concluir que a1 → b é 2-interessante (duas vezes mais interessante) em relação a A → b. As duas últimas expressões mostram que não existe R ≥ 1 para que a2 → b possa ser R-interessante em relação a A → b.*

## 2.7 Considerações Finais

Um dos problemas de algoritmos tradicionais de associação é que eles geram muitas regras, dificultando a análise das mesmas pelo usuário. Além disso, o conhecimento fornecido por padrões tradicionais tende a ser muito específico, pois eles são compostos apenas por itens da base de dados.

Nesse sentido, Srikant e Agrawal (1995) introduziram a mineração de regras generalizadas, utilizando o conceito de transações estendidas. O principal diferencial de padrões generalizados, em relação aos tradicionais, é que eles são compostos por itens presentes em qualquer nível de uma estrutura taxonômica.

Entretanto, o fato de algoritmos como de Srikant e Agrawal (1995) utilizarem transações estendidas permite que o conjunto de regras apresentadas ao usuário seja até mais volumoso, e contenha problemas de redundância. Dessa forma, é necessário que se utilize medidas de interesse, para podar o conjunto extraído. Entretanto, além de serem subjetivas, pois dependem da escolha do usuário, essas medidas não garantem uma grande redução do volume de regras, dependendo do valor que é escolhido.

Dentro desse contexto, trabalhos relacionados ao pós-processamento possuem uma vantagem quantitativa, pois mostram que menos padrões podem ser gerados. Além disso, os trabalhos mostram que quando regras são agrupadas, elas podem ser substituídas por padrões generalizados correspondentes. Nesse caso, apenas a regra geral é apresentada ao usuário, evitando a ocorrência de regras generalizadas e específicas correlatas no mesmo resultado. Isso é muito importante, pois elimina a necessidade de utilização de medidas de poda.

Sendo assim, no capítulo seguinte será descrito que, além de representarem hierarquias de itens, ontologias podem proporcionar grande poder de inferências, por meio de reasoners que se baseiam em axiomas presentes na mesma. Além disso, quando se trata de enriquecimento semântico dos padrões, as inferências que podem ser realizadas são muito relevantes, inclusive em situações onde ocorram imprecisões e incertezas. Em domínios imprecisos, ontologias devem ser assistidas pela lógica difusa, que será mais bem detalhada no capítulo seguinte.

# Capítulo 3

# LÓGICA DIFUSA

## 3.1 Considerações Iniciais

A lógica difusa preocupa-se com os princípios formais do raciocínio aproximado (ZADEH, 1988), ou seja, ela é uma lógica precisa do raciocínio impreciso (ZADEH, 2008). É possível dizer que através da teoria dos conjuntos difusos, a lógica difusa captura informações imprecisas e as converte em formato numérico, fundamentando os modos de raciocínio que são aproximados ao contrário de exatos (ORTEGA, 2001).

Neste capítulo serão sumarizados alguns conceitos relacionados à lógica difusa, mais precisamente à teoria dos conjuntos difusos. A lógica difusa pode ser incorporada em tarefas de mineração de dados, e trabalhos recentes demonstram a viabilidade de se utilizar os conceitos dessa teoria em algoritmos de mineração, permitindo que os mesmos possam ser utilizados em domínios imprecisos.

A organização do capítulo está disposta da seguinte maneira:

- Seção 3.2: definições básicas sobre a teoria dos conjuntos clássicos;
- Seção 3.3: principais conceitos da teoria dos conjuntos difusos;
- Seção 3.4: será introduzido o conceito de relações difusas, e algumas de suas propriedades;
- Seção 3.5: sumarização do conteúdo apresentado no capítulo.

## 3.2 Conjuntos Clássicos

A definição de conjunto é muito geral, entretanto, sabe-se que um conjunto pode ser formado por membros que atendam a critérios de pertinência pré-definidos, ou então, caso ele não contenha elementos, deverá ser chamado de conjunto vazio.

Para Klir e Yuan (1995), Bojadziev e Bojadziev (1996), os três principais métodos matemáticos utilizados na representação de conjuntos são:

- Listagem dos elementos;
- Regra de pertinência;
- Função característica.

O método da listagem dos elementos consiste, por exemplo, em descrever o conjunto da seguinte maneira:

$$A = \{5, 6, 7, 8\}.$$

Uma regra de pertinência representa os conjuntos por propriedades que satisfazem somente seus membros (KLIR; YUAN, 1995); portanto, através desse método o conjunto *A* seria descrito como:

$$A = \{x \mid 5 \leq x \leq 8\}.$$

A *função característica,* conhecida também como *função de pertinência,* normalmente é representada da seguinte maneira:

$$\mu_{\mathrm{A}}(x) = \begin{cases} 1 & para \quad x \in A \\ 0 & para \quad x \in A \end{cases}$$

Tomando somente dois valores, 1 e 0, a função característica indica se um elemento é ou não membro de um conjunto, ou seja, para qualquer elemento *x* existem somente duas possibilidades em relação a um conjunto *A*: *x* pertence ou *x* não pertence a ele. Por exemplo, considerando o conjunto universo $U = \{2, 6, 7, 8, 9, 10\}$ e o subconjunto *A* de *U*, $A = \{6, 7, 8\}$, existem somente três elementos em *U* membros de *A.* Portanto, pela função de pertinência:

$$\mu_{\mathrm{A}}(2) = 0, \quad \mu_{\mathrm{A}}(2) = 0, \quad \mu_{\mathrm{A}}(7) = 1$$
$$\mu_{\mathrm{A}}(8) = 1, \quad \mu_{\mathrm{A}}(9) = 0, \quad \mu_{\mathrm{A}}(10) = 0$$

No entanto, existem situações em que a tarefa de classificar elementos como pertencentes ou não a um determinado conjunto se torna difícil. Isso ocorre quando o mesmo é ambíguo.

A ambiguidade de um conjunto, muitas vezes, está ligada a termos de caráter qualitativos relacionados à linguagem natural, por exemplo, “conjunto dos homens altos”, “conjunto dos indivíduos ricos”, “conjunto dos indivíduos pobres”.

Para lidar com esse tipo de situação, Zadeh (1965) propôs a teoria dos conjuntos difusos, que pode ser vista como uma generalização dos conjuntos clássicos, mas com um escopo de aplicabilidade potencialmente maior.

## 3.3 Conjuntos Difusos

Zadeh (1965), afirma que um conjunto difuso é definido por uma *função de pertinência,* representada por:

$$\mu_A : \mathrm{X} \rightarrow [0,1]$$

Essa função atribui a cada elemento $x$ de um conjunto X um valor $\mu_{\mathrm{A}}(x)$ no intervalo [0,1] que representa o *grau de pertinência* de $x$ em relação ao conjunto *A*.

Esse grau é uma medida que expressa a possibilidade de um elemento ser membro do conjunto. Sendo assim, pode-se dizer que essa função característica, diferentemente da função clássica, implica em um intervalo de valores, ou seja, ela torna flexível a relação de pertinência para conjuntos ambíguos (ORTEGA, 2001).

Um conjunto difuso geralmente é representado por uma equação ou então por gráficos matemáticos. Dubois e Prade (1980), por exemplo, afirmam que um conjunto difuso *A* pode ser representado através da seguinte equação matemática:

$$A = \mu_{\mathrm{A}}(x_1)/x_1 + \cdots + \mu_{\mathrm{A}}(x_n)/x_n = \sum_1^n \mu_{\mathrm{A}}(x_i)/x_i.$$

Nessa notação, a somatória representa a operação de união dos elementos do conjunto difuso *A* e $\mu_{\mathrm{A}}(x_i)/x_i$ se refere ao elemento $x_i$ que pertence ao conjunto difuso *A* com grau $\mu_{\mathrm{A}}(x_i)$ (Ortega, 2001). Por exemplo, se *A* = (0,8/$x_1$) então $x_1$ pertence ao conjunto *A* com grau 0,8.

Paralelamente, Dubois e Prade (1980), apresentam outra notação:

$$A = \{(x, \mu_A(x)), \forall\, x \in X\}$$

Embora elas sejam aparentemente distintas, em qualquer situação os conjuntos obtidos por elas serão os mesmos.

Klir e Yuan (1995) apresentam um exemplo clássico de conjuntos difusos, onde são definidos três conjuntos que representam os conceitos jovem, adulto e idoso, através de três funções de pertinência definidas no intervalo de idades [0,80]. O exemplo desses autores é ilustrado nas Figuras 3.1 e 3.2, sendo que a Figura 3.1 exibe as funções de pertinência, $\mu_J(x)$, $\mu_A(x)$ e $\mu_I(x)$ para os conjuntos e a Figura 3.2 ilustra suas representações gráficas.

Segundo Zadeh (1965), Dubois e Prade (1980), operações envolvendo a teoria clássica, como: c*onjunto vazio, igualdade, subconjunto, complemento, união e intersecção* podem ser estendidas para a teoria nebulosa através das seguintes definições:

- *Conjunto vazio*: $\forall\, x \in X,\ \mu_{\emptyset}(x) = 0$;
- *Igualdade*: $\forall\, x \in X,\ A = B \Leftrightarrow \mu_A(x) = \mu_B(x)$;
- *Subconjunto difuso*: $\forall\, x \in X,\ A \subseteq B \Leftrightarrow \mu_A(x) \leq \mu_B(x)$;
- *Complemento*: $\forall\, x \in X,\ \mu_{\bar{A}}(x) = 1 - \mu_A(x)$;
- *União*: $\forall\, x \in X,\ \mu_{A \cup B}(x) = max(\mu_A(x),\ \mu_B(x))$;
- *Intersecção*: $\forall\, x \in X,\ \mu_{A \cap B}(x) = min\big(\mu_A(x),\ \mu_B(x)\big)$.

| | |
|---|---|
| $\mu_J(x) = \begin{cases} 1 \\ (35-x)/15 \\ 0 \end{cases}$ | $\begin{array}{l} \text{Quando } x \leq 20 \\ \text{Quando } 20 < x < 35 \\ \text{Quando } x \geq 35 \end{array}$ |
| $\mu_A(x) = \begin{cases} 0 \\ (x-20)/15 \\ (60-x)/15 \\ 1 \end{cases}$ | $\begin{array}{l} \text{Quando } x \leq 20 \text{ ou } x \geq 60 \\ \text{Quando } 20 < x < 35 \\ \text{Quando } 45 < x < 60 \\ \text{Quando } 35 \leq x \leq 45 \end{array}$ |
| $\mu_I(x) = \begin{cases} 0 \\ (x-45)/15 \\ 1 \end{cases}$ | $\begin{array}{l} \text{Quando } x \leq 45 \\ \text{Quando } 45 < x < 60 \\ \text{Quando } x \geq 60 \end{array}$ |

**Figura 3.1 - Funções de Pertinência para *Jovem, Adulto* e *Idoso* – adaptada de Klir e Yuan (1995).**

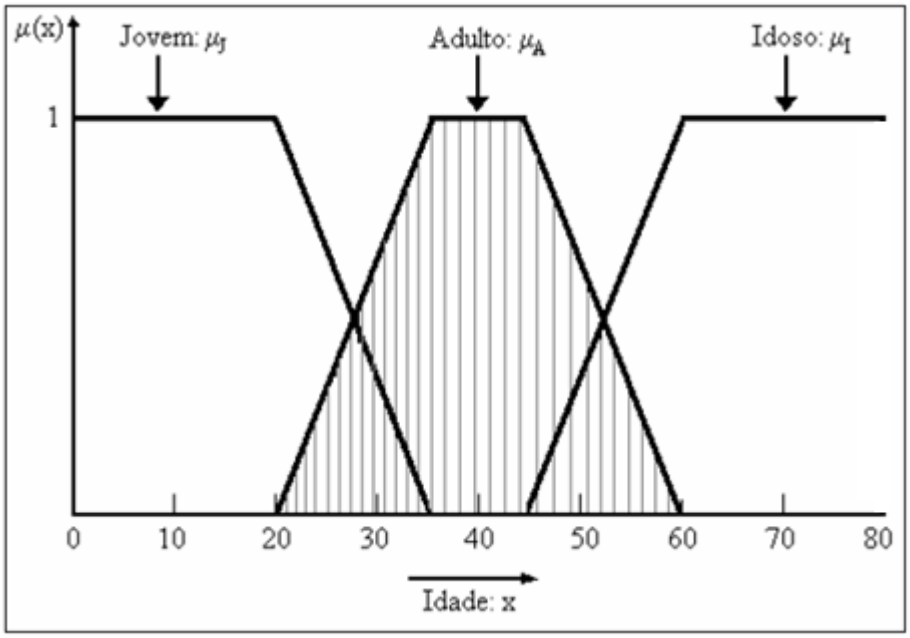


**Figura 3.2 - Representação Gráfica dos Conceitos Jovem, Adulto e Idoso – adaptada de (Klir e Yuan, 1995).**

Além das definições apresentadas, o conceito de relações difusas é muito importante dentro da teoria dos conjuntos difusos. Essas relações possuem uma gama de aplicações muito ampla, algumas delas são inferências e semântica, e podem ser consideradas extensões das relações comuns (TERANO, ASAI *et al.*, 1992).

## 3.4 Relações Difusas

O conceito de relação é muito geral, mas está baseado nos conceitos de par ordenado $(a, b)$ e produto cartesiano $A \times B$ de dois conjuntos *A* e *B* (Bojadziev e Bojadziev, 1996).

O produto cartesiano $A \times B$ é o conjunto de pares ordenados:

$$A \times B = \{(a, b) | a \in A, b \in B\}$$

Sendo assim, uma *relação* entre A e B, é qualquer subconjunto *R* de $A \times B$ (Bojadziev e Bojadziev, 1996). Por exemplo, considerando *A* = $\{a_1, a_2\}$ e *B* = $\{b_1, b_2\}$, $A \times B$ geraria um conjunto de pares ordenados $P = \{(a_1, b_1), (a_1, b_2), (a_2, b_1), (a_2, b_2)\}$, onde cada um de seus subconjuntos é uma relação binária.

Nesse sentido, uma relação pode ser considerada como uma interação entre dois ou mais elementos, podendo ser binária (dois elementos), ternária (três elementos) e assim sucessivamente.

Relações tais como “x e y são quase iguais”, “x e y são muito similares” ou então “x é mais bonito do que y”, são corriqueiras, mas devido à ambiguidade dos termos “quase iguais”, “muito similar” e “mais bonito”, elas são difíceis de serem representadas pela teoria clássica (TERANO; ASAI; SUGENO, 1992). Sendo assim, uma alternativa para representar tais situações seria utilizar o conceito de relações difusas.

Uma relação difusa *R* entre dois conjuntos *A* e *B* é qualquer subconjunto do produto cartesiano $A \times B$, caracterizado por possuir uma função $\mu_R(x, y)$ (Bojadziev e Bojadziev, 1996). Essa função denota a intensidade com que *x* e *y* se associam, e pode ser representada da seguinte maneira:

$$\mu_R : X \times Y \rightarrow [0,1].$$

Portanto, a relação difusa *R* pode ser representada, por exemplo, da seguinte maneira:

$$R = \{(a,b),\ \mu_R(a,b) | (a,b) \in A \times B,\ \mu_R(a,b) \in [0,1]\}.$$

Segundo (Klir e Yuan, 1995), algumas propriedades das relações clássicas também foram estendidas para as relações difusas:

- *Propriedade reflexiva*: $\forall\ x \in X,\ \mu_R(x,x) = 1$;
- *Propriedade simétrica*: $\forall\ (x,y) \in X \times X,\ \mu_R(x,y) = \mu_R(y,x)$;
- *Propriedade transitiva (transitividade max-min*):

$$\forall\ (x,z) \in X \times X,\ \mu_R(x,z) \geq max_{y \in Y}\ min[\mu_R(x,y),\ \mu_R(y,z)].$$

A *transitividade max-min* não é a única maneira de se expressar transitividade difusa, alternativamente, outras definições são possíveis e úteis em algumas aplicações (KLIR; YUAN, 1995), (DUBOIS; PRADE, 1980).

Transitividade é uma propriedade da teoria clássica que permite a execução de inferências entre dois elementos a partir de um elemento em comum entre eles. Por exemplo, se x = y e y = z, através da propriedade transitiva podemos inferir que x = z. Além disso, ela está associada ao conceito de composição, portanto, a *transitividade max-min* se relacionada à *composição max-min*. Uma composição $R \circ S$ é uma relação que associa transitivamente dois conjuntos *R* e *S* a partir de um conjunto *Z* em comum.

Nesse sentido, Terano, Asai *et al.* (1992) afirmam que se *R* é uma relação difusa em $X \times Y$ e *S* é uma relação difusa em $Y \times Z$, uma composição *max-min* $R \circ S$ é uma relação difusa em $X \times Z$, com grau $\mu_R(x,z)$, definida como a seguir:

$$R \circ S = \left\{\left((x,z), \max_y \left(min\left(\mu_R(x,y),\ \mu_S(y,z)\right)\right)\right)\right\}, (x,z) \in A \times C, y \in B$$

Sendo assim, a transitividade *max-min* presente nessa composição, permite que através de um elemento *S* em comum, X e *Y* possam se associar com um determinado grau de relacionamento. Em composições clássicas o grau $\mu_R(x,z)$ é sempre 1, portanto não necessita ser incluído na relação.

Além desses conceitos, a propriedade transitiva, juntamente com as propriedades reflexiva e simétrica, permite a geração de um tipo de relação muito importante, conhecida como relação de similaridade, ou então como relação de equivalência difusa. Uma relação de similaridade é uma relação difusa reflexiva, simétrica e transitiva (DUBOIS, 1980). A relação de similaridade não é a única possível de se obter, além dela outras podem ser encontradas (TERANO; ASAI; SUGENO, 1992).

## 3.5 Considerações Finais

O conhecimento obtido na mineração de dados depende da qualidade dos padrões gerados, ou seja, quanto mais próximos da realidade forem esses padrões melhor será o aproveitamento dos mesmos. Com base nisso, tendo em vista as diversas situações com pouca ou nenhuma precisão existentes no cotidiano, a lógica difusa pode ser utilizada para apoiar algoritmos de associação, na obtenção de padrões mais consistentes e mais claros ao usuário.

No capítulo seguinte serão apresentados os principais conceitos de ontologias. Além disso, considerando que o trabalho proposto está inserido no contexto de ontologias difusas, também serão detalhados alguns conceitos desse tipo de estrutura, valendo-se do conteúdo apresentado no capítulo atual.

# Capítulo 4

# ONTOLOGIAS

## 4.1 Considerações Iniciais

Conceitualização é um termo muito utilizado em ontologias. Em termos gerais, uma conceitualização é uma abstração de uma visão simplificada do mundo em relação a algum domínio. Portanto, pode-se dizer que o papel de uma ontologia é assimilar uma conceitualização, e representá-la através de formalismos codificados por uma linguagem de programação específica.

Conforme descrito no Capítulo 2, os itens de uma base de dados podem ser organizados hierarquicamente através de taxonomias de itens. Complementando esse conceito com o conteúdo apresentado no Capítulo 3, no Capítulo 4, dentre outras coisas, será dito que uma das possíveis abordagens de ontologias é a de taxonomias difusas.

Sendo assim, espera-se que o conteúdo descrito no decorrer deste capítulo possibilite à leitura dos capítulos posteriores o entendimento de que ontologias podem ser utilizadas com diferentes finalidades em tarefas de mineração, inclusive como um meio de auxiliar os algoritmos de mineração a raciocinar sobre os itens de uma base de dados.

O capítulo está organizado da seguinte maneira*:*

- Seção 4.2: procura sumarizar os principais conceitos envolvidos no tema Ontologias, além da definição das mesmas;

- Seção 4.3: serão introduzidos alguns tipos de ontologias em relação aos níveis de generalidade das mesmas, algumas definições relacionadas ao projeto de ontologias e as principais linguagens utilizadas;
- Seção 4.4: será definido o que são ontologias difusas;
- Seção 4.5: serão descritos alguns tipos de representações de ontologias;
- Seção 4.6: sumarização do conteúdo apresentado.

## 4.2 Conceitos e Definições Básicas

Conforme mencionado, o termo conceitualização é muito utilizado e está diretamente relacionado às ontologias. Uma conceitualização é uma abstração de uma visão simplificada de algum domínio.

Ontologias podem ser utilizadas em diversos contextos, e para várias finalidades. Em Uschold e Gruninger (1996) o espaço de utilização das mesmas é subdividido em três categorias, conforme ilustrado pela Figura 4.1.

Uma das principais motivações para ênfase dada à construção de ontologias deve-se a possibilidade de partilha e reutilização de conhecimentos em diferentes aplicações que elas proporcionam (GUARINO, 1997).

No contexto filosófico, Ontologia se refere a um sistema particular de categorias que descreve uma determinada visão do mundo (GUARINO, 1998), ou seja, trata-se de uma teoria sobre a existência da natureza (BERNERS-LEE, HENDLER; LASSILA, 2001).

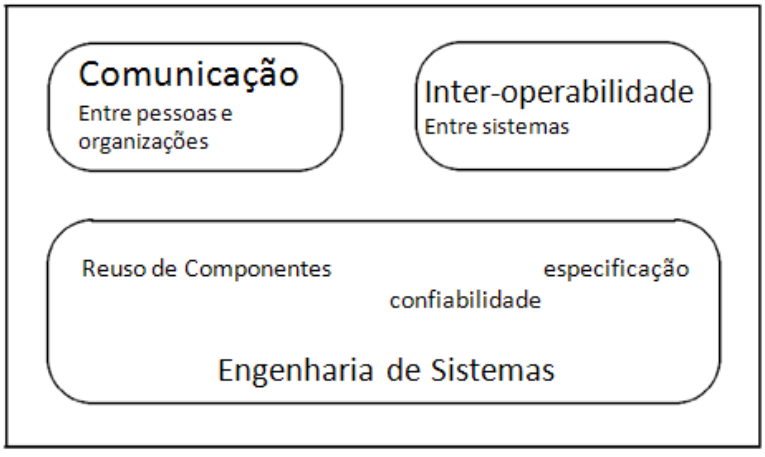


**Figura 4.1 - Usos de Ontologias – adaptada de Uschold e Gruninger (1996).**

No âmbito computacional, ontologia é a especificação formal e explícita de uma conceitualização compartilhada (GRUBER, 1993). Com o objetivo de facilitar o entendimento dessa definição, Uschold e Gruninger (2004) apresentaram uma análise do significado dos termos envolvidos:

- *Conceitualização*: refere-se a um modelo abstrato que representa como as pessoas pensam sobre as coisas do mundo, normalmente restrito a alguma área particular;
- *Especificação Explícita*: significa que são atribuídos *nomes* e *definições* explícitas aos conceitos e relacionamentos do modelo abstrato. O *nome* é um termo e a *definição* é a especificação do significado do conceito ou relacionamento;
- *Formal*: significa que a especificação da conceitualização é codificada através de uma linguagem cujas propriedades formais são bem compreendidas. Essa formalização é importante para remover a ambiguidade prevalente na linguagem natural e também em outras notações informais. Além disso, a formalização possibilita o uso de mecanismos de inferência automática para derivar novas informações a partir das especificações;
- *Compartilhada*: significa que uma ontologia pode ser reutilizada em diferentes aplicações e comunidades, sendo que essa reutilização é um dos principais propósitos de uma ontologia.

Ontologias são modelos compostos por classes, atributos, relacionamentos, axiomas e instâncias.

As classes (conhecidas também como conceitos) são organizadas em uma taxonomia de classes e subclasses, representando entidades pertencentes a um domínio (USCHOLD; GRUNINGER, 2004).

Cada classe está tipicamente associada a várias propriedades, que são os atributos e relacionamentos. Os atributos descrevem características das classes e os relacionamentos expressam como elas se relacionam.

Axiomas são restrições sobre: classes, atributos e relacionamentos, com objetivo de permitir que máquinas possam interpretá-los através de mecanismos de raciocínio automático.

Por fim, instâncias são os indivíduos pertencentes à conceitualização especificada pela ontologia. A Figura 4.2 apresenta um exemplo de uma ontologia

simples, e a Figura 4.3, apresenta um espectro evolutivo que ilustra algumas representações consideradas ontologias.

Nela, as classes estão representadas por elipses em branco, as propriedades por setas contínuas e os valores de atributos por retângulos. As setas tracejadas representam a instanciação e as instâncias possuem tom acinzentado. O exemplo possui um axioma com a propriedade de transitividade associado ao relacionamento "*parte_de"*, possibilitando inferir que instâncias de *Cidade* também fazem parte de instâncias de *País.*

Existem muitas discussões sobre quais representações devem realmente ser consideradas ontologias, principalmente quando elas especificam algum vocabulário de termos (USCHOLD; GRUNINGER, 2004). Porém, em meio a esses debates, Uschold e Gruninger consideram a existência de um núcleo de afirmações em comum que deve ser respeitado:

- Ontologias possibilitam um vocabulário de termos que se refere a coisas de interesse em um determinado domínio;
- Ontologias fornecem alguma especificação do significado dos termos, baseada preferencialmente em algum tipo de lógica.

Essas duas afirmações sugerem que ao analisar uma determinada representação, além de verificar se a mesma provê um vocabulário de termos, deve-se aferir se ela especifica o significado dos mesmos, ou seja, para ser uma ontologia não basta ela especificar somente algum tipo de vocabulário, pois ela também deve possuir axiomas que especificam o significado dos termos.

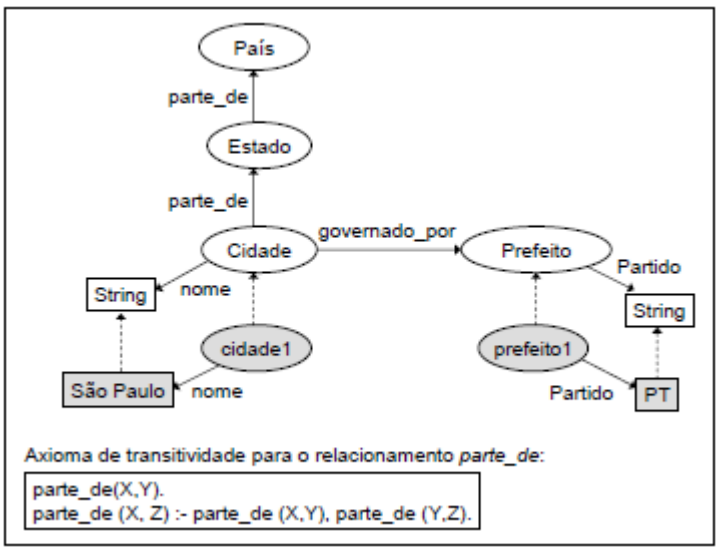


**Figura 4.2 – Exemplo de Ontologia Simples (Yaguinuma, 2007).**

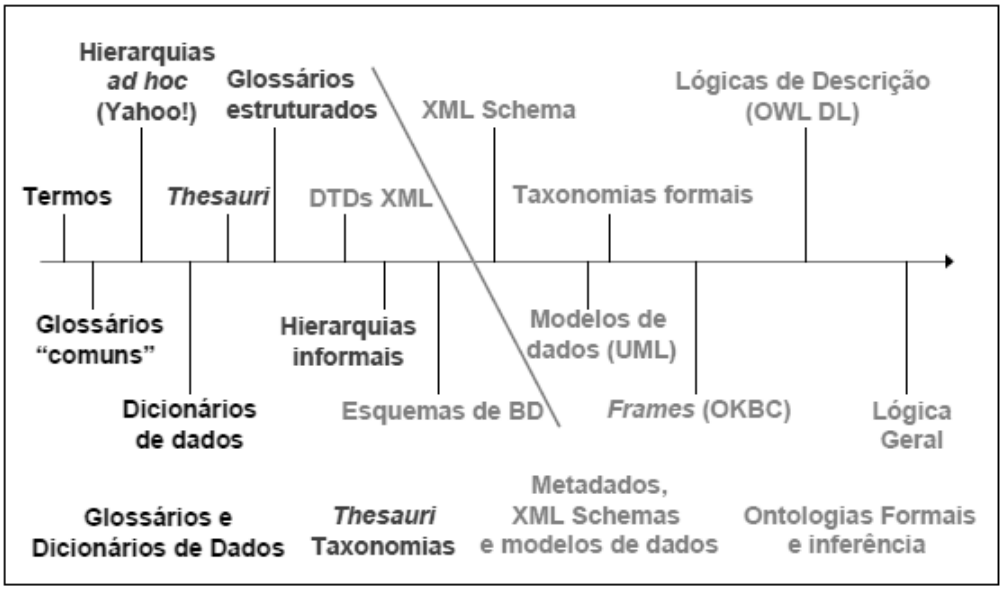


**Figura 4.3 - Tipos de Representações de Ontologias (Yaguinuma, 2007).**

No extremo esquerdo da Figura 4.3 estão às representações que contêm apenas termos com pouco ou nenhuma especificação de significado, chamadas de "ontologias leves". À medida que se segue da esquerda para direita, o grau de formalização aumenta e a especificação de significado se torna mais precisa, reduzindo a ambiguidade. Finalmente, no extremo direito estão as teorias rigorosamente formalizadas, que compõe as ontologias "autenticas".

Além do exposto, é possível desenvolver diferentes tipos de ontologias de acordo com o seu nível de generalidade (GUARINO, 1998). Esses tipos de ontologias, algumas linguagens específicas e alguns conceitos relacionados ao projeto de ontologias, serão apresentados na seção 4.3.

## 4.3 Tipos, Projetos e Linguagens de Ontologias

De acordo com Uschold e Gruninger (1996), o principal assunto relacionado à utilização de ontologias em domínios tais como modelagem empresarial e arquiteturas multiagentes, é a criação de um ambiente de integração para diferentes softwares. Nesses casos, ontologias estão associadas a questões de interoperabilidade entre sistemas, podendo ser utilizadas como apoio à interpretação das diferentes linguagens e representações dos sistemas envolvidos.

Guarino (1998) salienta que a capacidade de proporcionar a interoperabilidade está associada ao nível de generalidade da ontologia, ou seja, o quão geral ela é. Por exemplo, seria inviável construir um sistema de integração de utilizando uma ontologia muito específica, sendo mais interessante desenvolver uma ontologia genérica, que englobe uma conceitualização mais ampla. Esses níveis de generalidade são utilizados para classificar os possíveis tipos de ontologias. A Figura 4.4 ilustra alguns desses tipos em relação a três níveis de generalidade.

As o*ntologias de alto nível* descrevem conceitos muito gerais como espaço, tempo, matéria, objeto, evento, ações, etc., que são independentes de um problema particular ou domínio.

Por outro lado, o*ntologias de domínio e ontologias de tarefa* descrevem, respectivamente, o vocabulário relacionado a um domínio genérico (como medicina ou automobilismo, por exemplo), ou uma tarefa ou atividade genérica (como diagnosticar ou vender, por exemplo), através da especialização dos termos introduzidos nas ontologias de alto nível.

*Ontologias de aplicação* representam conceitos em função tanto de um domínio particular quanto de uma tarefa, que frequentemente são especializações de ambas as ontologias relacionadas. Estes conceitos geralmente correspondem a papéis desempenhados pelas entidades do domínio enquanto executam uma determinada atividade.

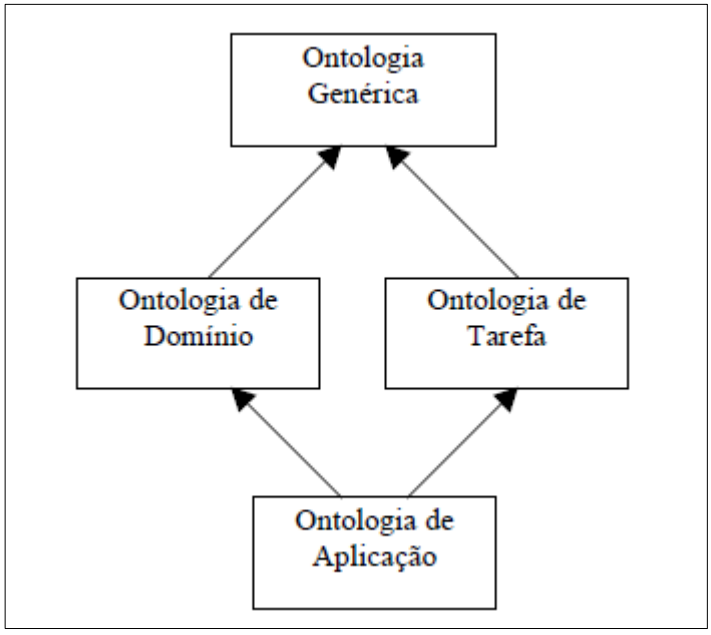


**Figura 4.4 - Tipos de Ontologias – adaptada de (Guarino, 1998).**

Segundo Gruber (1995), o projeto de ontologias deve seguir alguns critérios que objetivam assegurar a qualidade do modelo em desenvolvimento, sendo eles:

- *Clareza*: Uma ontologia deve transmitir efetivamente o significado pretendido dos termos definidos. As definições devem ser objetivas e devem ser documentadas em linguagem natural;
- *Coerência:* Uma ontologia deve ser coerente, ou seja, ela deve permitir inferências que sejam consistentes em relação às definições. Se alguma sentença inferida a partir de axiomas contradiz uma definição então a ontologia é incoerente;
- *Extensibilidade*: Uma ontologia deve oferecer uma base conceitual para uma série de tarefas previstas, e a representação deve ser trabalhada para que seja possível estendê-la e especializá-la. Em outras palavras, deve ser possível definir novos termos para uma utilização especial baseando-se no vocabulário disponível, de forma que não seja necessária a revisão das definições existentes;
- *Mínimo de tendências (viés) de codificação*: A conceitualização deve ser especificada em nível de conhecimento sem depender de uma codificação com símbolos particulares. A codificação deve ser feita exclusivamente por conveniência de notação ou execução. O viés de codificação deve ser minimizado, pois os agentes de conhecimento compartilhado podem estar implementados em diferentes estilos de representação;
- *Mínimo comprometimento ontológico*: Ontologias devem ser produzidas com o mínimo de afirmações possível sobre o mundo modelado, definindo apenas termos que são essenciais à transmissão de conhecimento consistente, permitindo, quando necessário, uma maior liberdade para especialização e instanciação da ontologia.

Além desses critérios qualitativos, Noy e Mcguinness (2001) sugerem sete passos a serem considerados durante a criação de uma ontologia, porém, salientam que não existe um padrão definido para a criação das mesmas. Os passos apresentados pelos autores são:

1. Determinar o domínio e escopo da ontologia;

2. Reutilizar ontologias existentes: considerar a existência de uma ontologia semelhante, e verificar a possibilidade de refiná-la para uma tarefa específica;
3. Levantar termos importantes a serem utilizados na ontologia: Quais as propriedades dos termos? O que é possível dizer sobre estes termos?
4. Definir classes, sua hierarquia e especializações (subclasses);
5. Definir as propriedades das classes (slots): uma classe de vinhos pode ter os seguintes slots: cor, sabor, quantidade de açúcar, ano de safra e produtor;
6. Restrições das propriedades (facetas): os slots podem possuir diferentes facetas, por exemplo, tipos de valores, valores permitidos, cardinalidade, e outros. O nome de um vinho (ou slot nome) pode receber uma string de até vinte caracteres. Esta restrição é uma faceta do slot nome;
7. Criação de instâncias individuais para as classes na hierarquia: escolhe-se a classe, cria-se uma instância individual para a classe escolhida e preenchem-se os valores dos slots.

Outro aspecto importante no desenvolvimento de ontologias é a escolha da linguagem que permita capturar de forma efetiva a conceitualização do domínio. Em geral, pode-se dizer que uma ontologia é um conjunto de axiomas lógicos (formais) que descreve o significado dos termos de um vocabulário, permitindo que máquinas e humanos possam interpretá-los sem ambigüidade.

Nesse sentido, a linguagem utilizada deve estar baseada em conceitos de representação formal do conhecimento (YAGUINUMA, 2007). Segundo a autora, as lógicas de descrição são consideradas uma das famílias mais importantes de representação formal do conhecimento, constituindo a base para linguagens de representação de ontologias como, por exemplo, a *Web Ontology Language* (OWL) (SMITH; WELT; MCGUINNESS, 2004).

OWL é uma das linguagens mais difundidas dentre as que são baseadas nas lógicas de descrição, pois ela é uma recomendação oficial do consórcio *World Wide Web Consortium* (W3C) para a criação de ontologias na Web Semântica (BERNERS-LEE, HENDLER; LASSILA, 2001). Além disso, a OWL é subdividida em três sublinguagens (dialetos), de acordo com o nível de complexidade e expressividade dos mesmos:

- *OWL Lite:* é o dialeto de OWL menos expressivo e complexo, pois provê elementos básicos para a representação de conceitos, relacionamentos e restrições simples de propriedades. Devido à facilidade de suas construções, ele facilita o desenvolvimento de ferramentas e máquinas de inferência que manipulam as ontologias;
- *OWL DL:* dentre as linguagens utilizadas para a criação de ontologias na *Web* Semântica, é uma das mais utilizadas, pois provê maior expressividade que a OWL Lite, garantindo que todas as suas inferências sejam computáveis.
- *OWL Full:* é o dialeto que provê máxima expressividade, mas sem garantias computacionais. Sendo assim, ainda não foram desenvolvidas máquinas de inferência capazes de tratar todas as construções possíveis deste dialeto.

Existem situações em que a lógica clássica não permite a representação de determinados conceitos (STRACCIA, 2006), portanto, a utilização de ontologias tradicionais (*crisp*) nessas situações torna-se inviável. Por exemplo, é difícil representar em ontologias *crisp* conceitos como “cremoso”, “escuro” ou “quente”, para os quais não é possível obter uma definição precisa. A seção 4.4 apresenta alguns conceitos sobre esse tipo de situação.

## 4.4 Ontologias Difusas

De acordo com a seção 4.3, ontologias representam conceitos, propriedades, relacionamentos, instâncias e axiomas de um domínio de aplicação. Em domínios imprecisos é possível incorporar conceitos da lógica nebulosa em ontologias tradicionais, resultando em ontologias difusas.

Nesse sentido, Calegari e Ciucci (2007) descrevem uma ontologia difusa como uma quíntupla $O_F = \{C, I, R, F, A\}$, onde:

- **I** é o conjunto de instâncias dos conceitos;
- **C** é o conjunto de conceitos. Cada conceito $C \in C$ é um conjunto difuso no domínio de instâncias $C : I \rightarrow [0,1]$. O conjunto de entidades de uma ontologia difusa será indicado por **E**, ou seja, $E = C \cup I$.

- **R** é o conjunto de relações. Cada $R \in \boldsymbol{R}$ é uma relação difusa *n-aria* no domínio de entidades $R : \boldsymbol{E^n} \rightarrow [0,1]$. A relação taxonômica $T : \boldsymbol{E^2} \rightarrow [0,1]$, por exemplo, identifica a relação de subordinação difusa entre as entidades da ontologia.
- **F** é o conjunto de relações difusas associadas a um conjunto de entidades **E**, e a um domínio específico contido em $D = \{inteiro, string, \ldots\}$. Elas são funções *n-arias* tais que cada elemento $F \in \boldsymbol{F}$ é uma relação $F : \boldsymbol{E^{(n-1)}} \times P \rightarrow [0,1]$, onde *P* ∈ *D.*
- **A** é o conjunto de axiomas expressados em uma linguagem própria, ou seja, são predicados que restringem o significado dos conceitos, indivíduos, relacionamentos e funções.

Algumas das principais abordagens sobre ontologias difusas existentes na literatura são: taxonomias difusas, ontologias contendo classes e relacionamentos difusos e ontologias contendo hierarquia difusa de classes e relacionamentos (YAGUINUMA, 2007). Essas abordagens serão apresentadas na seção 4.5.

## 4.5 Abordagens de Ontologias Difusas

A abordagem mais simples para ontologias difusas considera taxonomias de termos relacionados por generalização/especialização com graus de pertinência entre os mesmos. Portanto, um termo pode ser mais genérico ou mais específico do que outro com um determinado valor.

Ontologias com classe e relacionamentos difusos são mais completas do que ontologias baseadas em taxonomias difusas. Essa abordagem considera estruturas compostas por classes, instâncias, propriedades e axiomas para representar a semântica do domínio. Nelas, o conceito de classe foi redefinido para classe difusa, correspondente ao conjunto difuso, e o de relacionamento para relacionamento difuso, equivalente à relação difusa (YAGUINUMA, 2007).

Assim como na abordagem baseada em taxonomias difusas, nas ontologias com classes e relacionamentos difusos as instâncias podem pertencer a uma classe com um grau no intervalo [0,1], porém, diferentemente da primeira, a segunda

abordagem possibilita que essas instâncias também se relacionem. Isso permite, por exemplo, que possam ser identificadas relações de similaridade entre as mesmas.

A Figura 4.5 ilustra uma ontologia com classes e relacionamentos difusos. As instâncias são as elipses acinzentadas relacionadas às elipses em branco, que são as classes. A generalização/especialização entre as classes é representada por setas contínuas, e a instanciação por setas tracejadas.

Além das duas abordagens mencionadas, Straccia (2006) propõe outra que estende as ontologias com classe e relacionamentos difusos, portanto, pode-se dizer que a abordagem de Straccia é mais completa que as duas abordagens anteriormente apresentadas. Trata-se de ontologias com hierarquia difusa de classes e relacionamentos. A principal diferença em relação às ontologias com classe e relacionamentos difusos está na forma como a generalização/especialização das classes é tratada, ou seja, a abordagem de Straccia permite que haja graus de pertinência entre classes e subclasses. Além disso, um relacionamento pode ser sub-relacionamento de outro, portanto, uma classe *A* pode ser subclasse de *B* com $\mu_{A \subseteq B}$ e um relacionamento *R* pode ser sub-relacionamento de *S* com $\mu_{R \subseteq S}$.

Yaguinuma, Santos e Biajiz (2007) propõem uma metaontologia que permite modelar classe e relacionamentos difusos para serem herdados e/ou instanciados pelas ontologias específicas crisp de domínio, de modo que estas sejam capazes de representar e realizar inferências sobre informações imprecisas. A metaontologia desenvolvida baseia-se na linguagem OWL DL, por ser uma linguagem recomendada pelo consórcio *World Wide Web Consortium* (W3C), que conta com diversas máquinas de inferência e *Application Programming Interface* (API) disponíveis para a implementação de aplicações baseadas em ontologias. A Figura 4.6 ilustra como a metaontologia representa classes imprecisas.

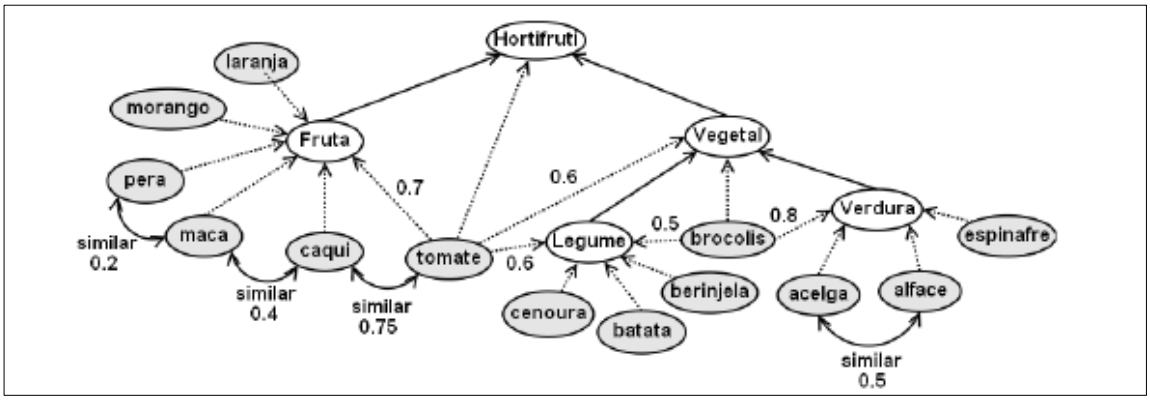


**Figura 4.5 - Exemplo de Ontologia Difusa com Classes e Relacionamentos Difusos.**

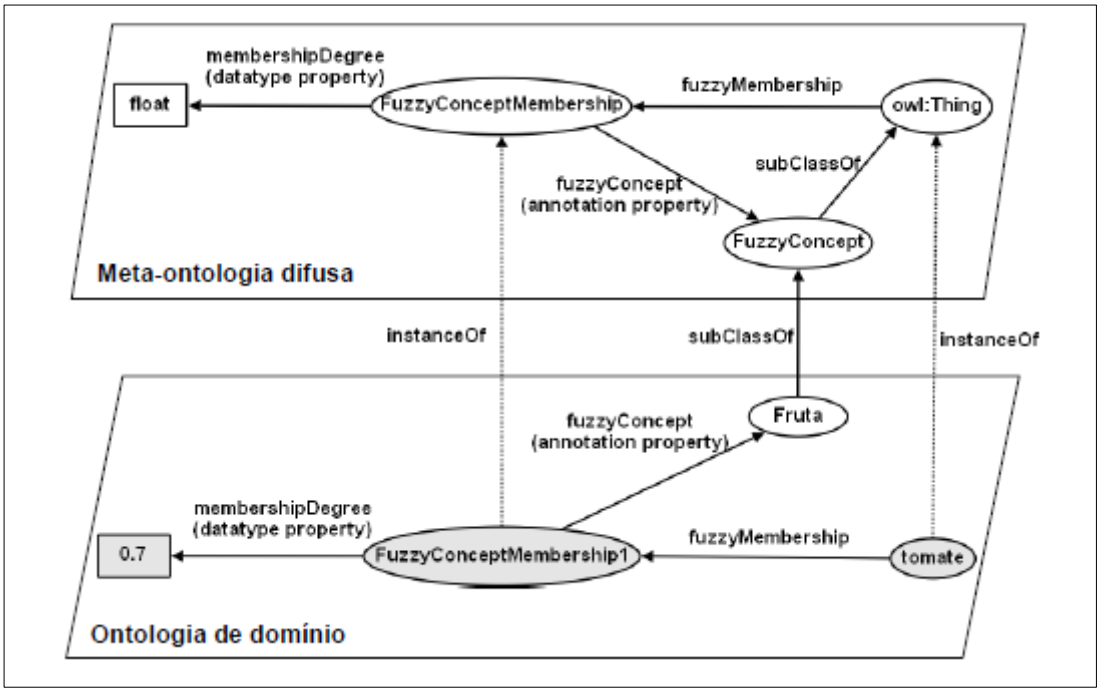


**Figura 4.6 – Representação abstrata de classe nebulosa.**

```
<Fruta rdf:ID="tomate">
  <rdfs:label>tomate</rdfs:label>
  <fuz:fuzzyMembership>
      <fuzz:FuzzyConceptMembership>
         <fuzz:fuzzyConcept rdf:resource="#Fruta"/>
          <fuzz:membershipDegree rdf:datatype="&xsd;float">0.7</membershipDegree>
      </fuzz:FuzzyConceptMembership>
 </fuzz:fuzzyMembership>
</Fruta>
```

**Figura 4.7 – Representação em OWL de uma instância de classe difusa.**

Na ilustração, as classes são as elipses em branco e as instâncias são as elipses acinzentadas. A classe *FuzzyConceptMembership* reúne as seguintes informações para representar classes difusas: a instância da ontologia que pertence a uma classe difusa (relacionamento *fuzzyMembership*), a classe difusa propriamente dita (relacionamento *fuzzyConcept* para classe *fuzzyConcept*) e o grau de pertinência equivalente (propriedade *membershipDegree*). No exemplo, *tomate* pertence à classe *Fruta* com grau de pertinência 0,7.

A metaontologia também permite representar relacionamentos difusos, através da classe *FuzzyRelationMembership* (Figura 4.8), que é responsável por atribuir um grau (*membershipDegree*) ao relacionamento difuso (*fuzzyRelationProp*) entre duas instâncias (*fuzzyRelationDomain* e *fuzzyRelationRange*).

Os relacionamentos difusos são modelados como sub-propriedades de *FuzzyRelation* e as instâncias de *FuzzyRelationMembership* associam as instâncias da ontologia de domínio ao relacionamento difuso e ao grau de pertinência

correspondente. Na Figura 4.8, ilustra-se o relacionamento *similarTo* entre *tomate* e *caqui* com grau 0,7. É importante salientar que a estratégia apresentada não estende a linguagem OWL, como FOWL e Fuzzy OWL, pois a sintaxe da linguagem não é modificada. Assim, a introdução de conceitos fuzzy não impede o uso de reasoners existentes, compatíveis com a OWL DL.

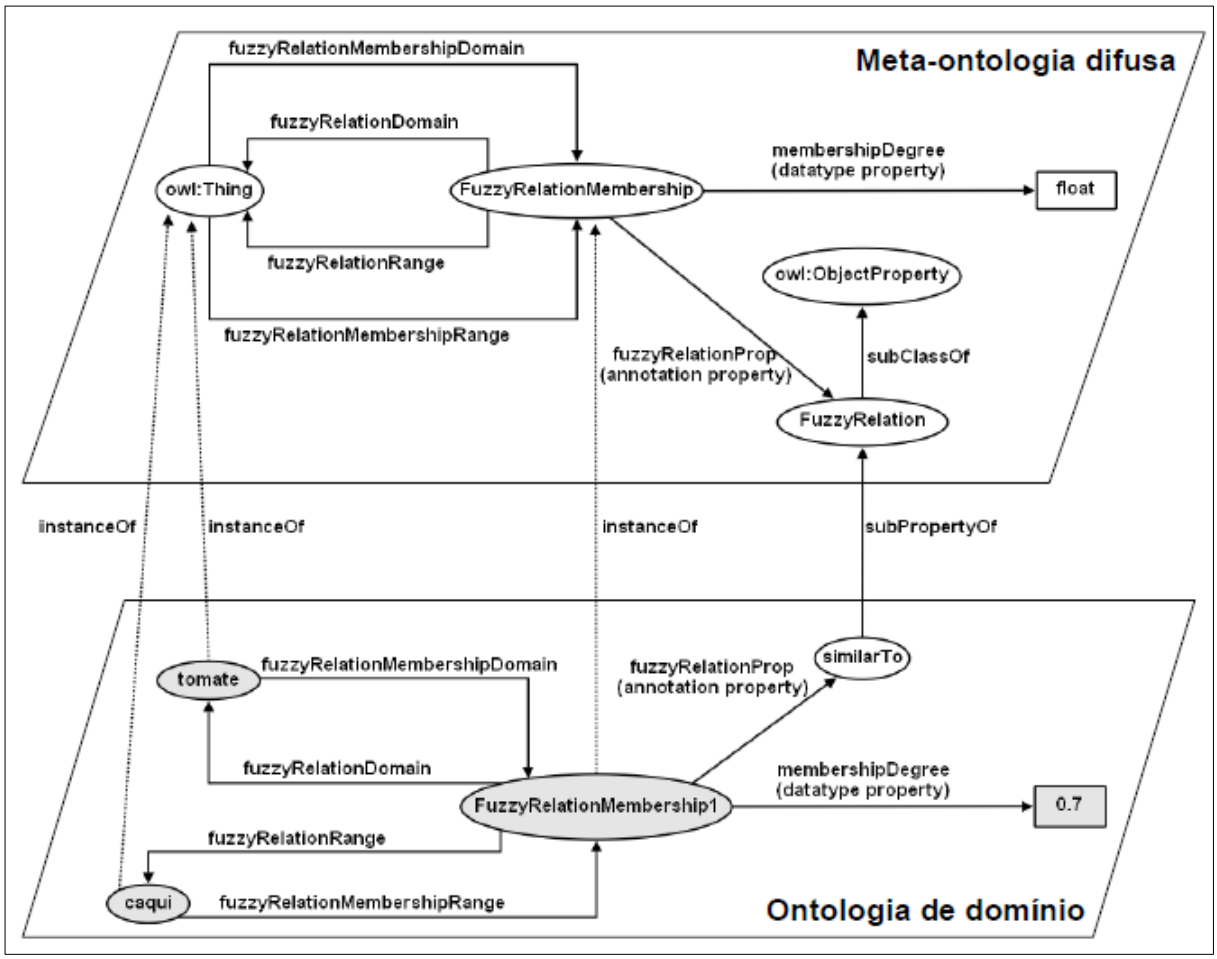


**Figura 4.8 – Representação abstrata de relacionamento difuso binário.**

```
<fuzz:FuzzyRelationMembership>
  <fuzz:fuzzyRelationProp rdf:resource="&fuz;similarTo"/>
  <fuzz:fuzzyRelationDomain rdf:resource="#tomate"/>
  <fuzz:fuzzyRelationRange rdf:resource="#caqui"/>
  <fuzz:membershipDegree rdf:datatype="&xsd;float">0.7</membershipDegree>
</fuzz:FuzzyRelationMembership>
```

**Figura 4.9 – Representação em OWL de relacionamento difuso binário.**

No entanto, em algumas situações, para obter uma representação mais apropriada de um determinado domínio é necessário que se considere distintos pontos de vistas, ou situações. Por exemplo, considere o problema de comparar dois vegetais, tomate e caqui, sob dois pontos de vista diferentes, aparência e classificação biológica. Sob o ponto de vista da aparência seria possível verificar que

tomate é muito similar ao caqui, mas no quesito classificação biológica os dois poderiam ser classificados como sendo pouco similares. Sendo assim, uma ontologia difusa com relação de similaridade entre esses itens deveria considerar dois graus distintos para a relação, um representando o grau de similaridade em relação aparência e outro em relação à classificação biológica, permitindo que a representação do domínio fosse mais bem efetuada.

Considerando essa questão, pode-se dizer que a metaontologia proposta por Yaguinuma, Santos e Biajiz (2007) modela somente um único grau de similaridade para cada relacionamento, de forma que o valor do relacionamento permanece independente do ponto de vista que está sendo analisado. Da mesma forma, os relacionamentos de especialização/generalização também permitem somente um único grau.

Nesse sentido, objetivando aprimorar o processo de representação de relações imprecisas modelada pela metaontologia proposta, Cerri, Yaguinuma e Santos (2010) propõem um novo modelo, denominado UFOCoRe (the upper fuzzy ontology with context representation), ou metaontologia difusa com representação de contexto. Nessa abordagem, o “ponto de vista” recebe o nome de contexto, portanto, o modelo permite a representação de múltiplos relacionamentos com mais de um grau em uma única ontologia, dependendo do contexto escolhido.

De acordo com os autores, UFOCoRe fornece ganho de produtividade na fase de construção de ontologias. Como os especialistas do domínio podem definir diversos contextos de uma maneira direta, editando uma única ontologia, a metodologia apresentada impede a criação de diversas ontologias para representar distintos graus de relacionamentos entre os indivíduos. As Figuras 4.10 ilustra a extensão (identificada pelo prefixo *ctx:*) incorporada à Fuzzy OWL e explicitam que a informação de contexto é incorporada ao representar relacionamentos difusos.

Como ilustrado na Figura 4.10, foi introduzida a classe *ctx:ContextFuzzyRelationMembership* nos relacionamentos difusos, que é responsável por associar os mesmos em diversos contextos. A propriedade *ctx:hasContext* faz a ligação da relação (*ctx:*FuzzyRelation) e seu grau (*ctx:ContextFuzzyRelationMembership*) com o contexto *ctx:contexto*.

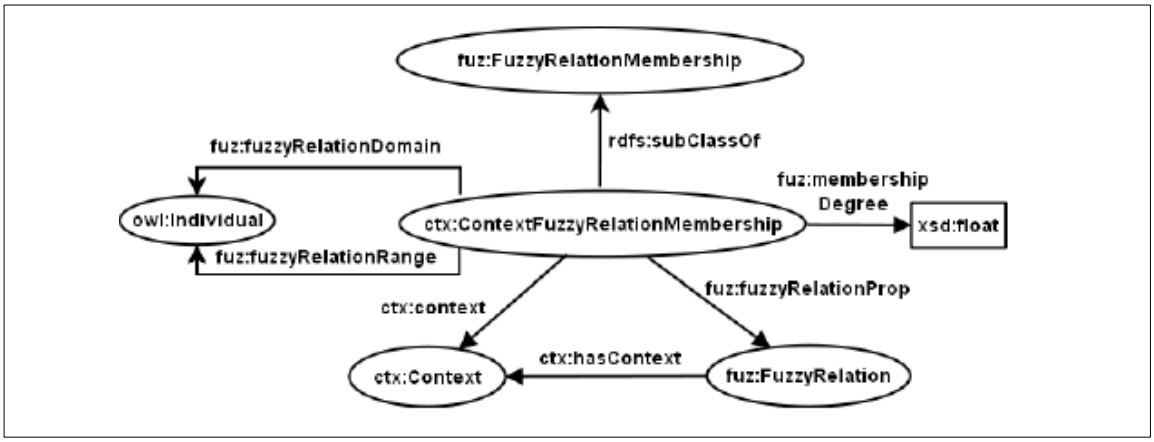


**Figura 4.10- Representação abstrata de relacionamento difuso com contexto.**

```
<!--contextos-->
<ctx:Context rdf:ID="aparencia"/>
<ctx:Context rdf:ID="sabor"/>

<!--classes-->
<owl:Class rdf:ID="Vegetais"/>

<!--instâncias-->
<Vegetais rdf:ID="caqui"/>

<Vegetais rdf:ID="tomate">
    <fuzz:similarTo_directly rdf:resource="#caqui"/>
</Vegetais>

<!--relacionamentos fuzzy em diferentes contextos-->
<ctx:ContextFuzzyRelationMembership>
    <fuzz:fuzzyRelationProp rdf:resource="&fuz;similarTo_directly"/>
    <ctx:context rdf:resource="#aparencia"/>
    <fuzz:fuzzyRelationDomain rdf:resource="#tomate"/>
    <fuzz:fuzzyRelationRange rdf:resource="#caqui"/>
    <fuzz:membershipDegree rdf:datatype="&xsd;float">0.7</fuzz:membershipDegree>
</ctx:ContextFuzzyRelationMembership>

<ctx:ContextFuzzyRelationMembership>
    <fuzz:fuzzyRelationProp rdf:resource="&fuz;similarTo_directly"/>
    <ctx:context rdf:resource="#sabor"/>
    <fuzz:fuzzyRelationDomain rdf:resource="#tomate"/>
    <fuzz:fuzzyRelationRange rdf:resource="#caqui"/>
    <fuzz:membershipDegree rdf:datatype="&xsd;float">0.3</fuzz:membershipDegree>
</ctx:ContextFuzzyRelationMembership>
```

**Figura 4.11 – Representação em OWL de relacionamentos fuzzy em vários contextos.**

## 4.6 Considerações Finais

Neste capítulo foram introduzidos os principais conceitos envolvendo ontologias e ontologias difusas. Englobando aspectos relacionados ao projeto, definições e características de ontologias crisp, alguns conceitos e abordagens de ontologias difusas e a questão da inserção de contexto em suas relações.

Como pontos principais pode-se destacar que ontologias possuem grande poder de inferências, proporcionam uma especificação formal do domínio, e ótima capacidade de reutilização. Portanto, elas são uma excelente alternativa para algoritmos de generalização.

# Capítulo 5

# LÓGICA DIFUSA NA TAREFA DE ASSOCIAÇÃO

## 5.1 Considerações Iniciais

A descoberta de conhecimento em banco de dados está associada a dois pontos principais: identificação de padrões interessantes e descrição desses padrões de maneira concisa e significativa (PEDRYCZ, 1998).

De acordo com Mitra e Acharya (2003), o uso de conjuntos difusos, que são naturalmente inclinados a lidar com o conhecimento linguístico do domínio, pode produzir soluções mais interpretáveis.

Nesse sentido, diversas aplicações utilizam conceitos da lógica difusa na mineração de dados. Algumas dessas abordagens serão apresentadas no decorrer deste capítulo.

O capítulo está disposto da seguinte maneira:

- A seção 5.2 trata sobre alguns conceitos relacionados à lógica difusa na mineração de regras de associação quantitativas. Além disso, serão apresentados alguns trabalhos envolvendo lógica difusa na mineração dessas regras, e na mineração de regras de associação generalizadas.
- Na seção 5.3 serão apresentadas algumas abordagens que utilizam ontologias difusas para trabalhar com a semântica dos itens na mineração de regras de associação generalizadas.

## 5.2 Lógica Difusa em Regras de Associação Quantitativas

Segundo Srikant e Agrawal (1996), o processo de mineração de dados por regras de associação em tabelas de domínio não binário é chamado de Mineração de Regras de Associação Quantitativas.

Em geral, o domínio não binário é aplicado em situações reais, nas quais os atributos podem ser quantitativos como no caso de “idade”, “número de dependentes”, ou categorizados como no caso de “sexo” e “tipo sanguíneo”. A Tabela 5.1 exibe um banco de dados que contém dados de atributos quantitativos, Idade e Nº de dependentes, e também dados do atributo categórico Sexo.

Para aplicar o processo de mineração de dados binários em um domínio quantitativo, basta mapear os atributos quantitativos e/ou categóricos para o domínio binário, onde cada atributo e seus respectivos valores geram novas colunas (Srikant e Agrawal, 1996). De acordo com essa abordagem, a partir do atributo Sexo, duas novas colunas seriam obtidas, (Sexo: M) e (Sexo: F).

Nos casos em que o domínio de valores é muito amplo essa solução não pode ser empregada, pois a grande quantidade de novas colunas geradas seria inviável. Uma solução, portanto, seria dividir esses valores em intervalos, formando faixas de valores. Sendo assim, o atributo Idade, por exemplo, poderia gerar as colunas (Idade: 0..29), (Idade: 30..49) e (Idade: 50..65). A Tabela 5.2 exibe o resultado do mapeamento da Tabela 5.1 para o domínio binário.

Nesse sentido, pode-se dizer que em regras de associação quantitativas os registros de uma tabela são considerados como transações, onde o par <atributo, valor> desempenha a função de um item na mineração de regras de associação.

**Tabela 5.1 – Exemplo de dados e atributos quantitativos.**

| TID | Idade | Nº de dependentes | Sexo |
|---|---|---|---|
| 1 | 25 | 0 | M |
| 2 | 30 | 1 | F |
| 3 | 50 | 2 | F |
| 4 | 20 | 3 | M |
| 5 | 65 | 2 | M |

**Tabela 5.2 – Mapeamento de dados e atributos quantitativos.**

| TID | Idade: 0..29 | Idade: 30..49 | Idade: 50..65 | Dependentes: 0..2 | Dependentes: 3..+ | Sexo: M | Sexo: F |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 2 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| 3 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| 4 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| 5 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |

Portanto, considerando a Tabela 5.2, poderiam ser extraídas, por exemplo, as seguintes regras de associação: <Idade: 0..29>, <Dependentes: 0..2> → <Sexo: M>; <Idade: 30..49> → <Sexo: M>.

Em regras quantitativas, a especificação de intervalos é um fator crítico, pois muitas vezes eles não são suficientemente concisos. Considerando isso, ao invés de intervalos, alguns algoritmos fazem uso de expressões linguísticas (ou *termos linguísticos*), que representam valores nebulosos, tais como *muito, pouco, demais, bastante*. Basicamente, esses algoritmos utilizam funções de pertinência para transformar cada valor quantitativo em conjuntos difusos, associando-os aos termos linguísticos.

Nesse sentido, Wai-Ho e Chan (1999), por exemplo, criaram o algoritmo *FARM* (*Fuzzy Association Rule Mining*). O algoritmo proposto permite que dois ou mais termos linguísticos possam ser associados, formando expressões linguísticas. Os autores utilizam um cálculo, denominado *diferença ajustada*, que permite identificar, dentre um conjunto de associações, quais são realmente interessantes, de modo que a formação das regras compostas por expressões ocorra logo após essa identificação.

Em Lee e Lee-Kwang (1997) também foi proposto um método para mineração de regras quantitativas, no qual regras de associações estendidas descrevem associações entre valores reais utilizando conjuntos difusos. Nessa abordagem, tuplas com dados quantitativos são convertidas em tuplas com dados binários com base em conjuntos difusos estabelecidos pelo usuário.

Após a conversão, um algoritmo convencional de associação é então aplicado para geração das regras, denominadas *regras estendidas*. Os autores afirmam que a utilização de conjuntos difusos permite reduzir o número de pares atributo/valor, além de possibilitarem uma descrição mais concisa e generalizada das regras de associação.

Por exemplo, caso existissem as regras:

(Hambúrguer, $5) → (Refrigerante, $2)

(Hambúrguer, $6) → (Refrigerante, $3)

(Hambúrguer, $4) → (Refrigerante, $1,5)

Elas poderiam ser escritas como:

(Hambúrguer, médio) → (Refrigerante, pequeno)

Shuhong, Jianxun e Pengcheng (2007) propuseram uma abordagem na qual os atributos quantitativos da base de dados, e os valores dos mesmos são transformados, respectivamente, em conjuntos difusos correspondentes, e graus de pertinência. Dessa forma, o banco de dados é transformado em uma matriz funcional composta por membros difusos, e a partir dessa transformação, regras de associação são geradas através do algoritmo *Apriori.*

Em Mohamadlou *et al.* (2009), os autores propuseram um algoritmo baseado em agrupamento difuso, usando a combinação de dados *crisp* e quantitativos. O algoritmo consiste em diversas etapas. Primeiramente, é aplicado um algoritmo de agrupamento difuso nos dados para extrair a distribuição do conhecimento. O segundo passo consiste em gerar intervalos difusos em cada atributo e eliminar dados inconsistentes. Na terceira etapa, através da projeção dos dados quantitativos em partições difusas, os dados quantitativos são transformados em *"transações discretas difusas"*. Finalmente, a partir dessas transações, as regras de associação são mineradas pelo algoritmo de mineração. A Figura 5.1 demonstra esse algoritmo.

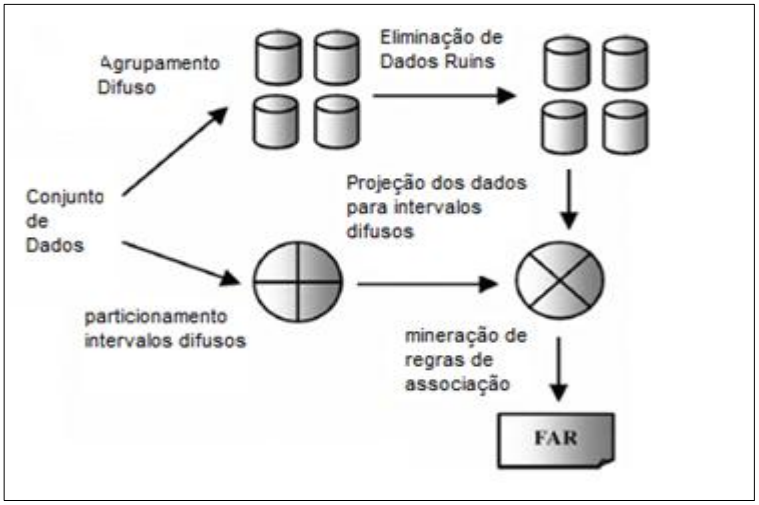


**Figura 5.1 – Diagrama do algoritmo - adaptada de Mohamadlou *et al.* (2009).**

## 5.3 Lógica Difusa em Regras de Associação Generalizadas

Os trabalhos que utilizam a lógica difusa na mineração de regras generalizadas, em sua maioria, estão focados na obtenção de regras de associação generalizadas difusas, que em geral são regras compostas por termos linguísticos fuzzy. É importante ressaltar que nesses trabalhos não são utilizadas taxonomias fuzzy, mas sim estruturas crisp, e que os termos linguísticos são formados com base em intervalos difusos, obtidos com base em abordagens de clusterização.

Por outro lado, são poucos os estudos (na mineração de regras generalizadas) que exploram a questão de diferentes graus de especialização/generalização em taxonomias difusas, conforme introduzido por Wei e Chen (1999). Os autores mencionados incluíram a possibilidade de relacionamento parcial em taxonomias, ou seja, enquanto em taxonomias crisp um filho pertence ao seu ancestral com grau 1, em taxonomias difusas esse grau pode variar no intervalo [0,1]. Por exemplo, um *tomate* pode ser considerado uma *fruta* ou um *vegetal*, porém com diferentes graus cada um.

Entretanto, quando se considera a variação de graus, algumas considerações devem ser feitas, principalmente em relação ao cálculo de suporte e confiança. Nesse sentido, além de definirem o conceito das estruturas, Wei e Chen também introduziram o conceito de graus de suporte e confiança estendidos, denominados *Dsupport* e *Dconfidence*. Portanto, diferentemente do que ocorre em domínios tradicionais, o grau de generalização de um filho em relação ao seu pai foi considerado no cálculo de suporte.

De acordo com os autores, o grau de qualquer nó *x* em relação ao seu ancestral *y* é calculado usando a combinação *max-min*, através da Equação 1,

$$\mu_{xy} = \max_{\forall l:\, x \to y} \left( \min_{\forall e \text{ on } l} \mu_{le} \right) \qquad (1)$$

Nessa fórmula $l: x \to y$ representa um dos caminhos entre *x* e *y*, $e$ em $l$ é uma das arestas do acesso $l$, $\mu_{le}$ é o grau na aresta $e$ em $l$. Se não houver nenhum acesso entre *x* e *y*, $\mu_{xy} = 0$.

Toda regra de associação possui um formato itemset correspondente. Por exemplo, o formato itemset da regra *A, B* → C é {A, B, C}. Supondo que estes sejam itens ancestrais da taxonomia, o suporte dessa regra é calculado somando-se todas

as ocorrências simultâneas de {A,B,C} nas transações da base de dados. Da mesma forma, é possível dizer que ele é calculado pela soma dos os graus que cada transação suporta o itemset {A,B,C}.

Como em taxonomias crisp o grau de especialização/generalização entre os itens é sempre 1, toda ocorrência de um elemento na base de dados corresponde a uma ocorrência de qualquer um de seus antepassados. Portanto, cada transação suporta o itemset {A,B,C} sempre com grau 1, desde que a mesma contenha, simultaneamente, qualquer item folha que pertença a A, B e C.

Entretanto, de acordo com as taxonomias fuzzy apresentadas por Wei e Chen, o grau entre os itens pode variar em [0,1] e, consequentemente, diferente dos casos crisp, o grau que cada transação suporta um itemset generalizado nem sempre será 1. Pensando nisso, os autores propuseram um grau de suporte estendido (*Dsupport*), calculado baseando-se no grau $\mu_{xy}$ (Equação 1) apresentado anteriormente.

Assim, se *a* é um valor em certa transação $t \in T$, *T* é o conjunto de transações e *x* é um atributo em certo itemset *X*, então, $\mu_{xa}$ é visto como o grau que a transação *{a}* suporta *x.* Portanto, o grau que *t* suporta *X* pode ser obtido como a seguir:

$$\mu_{tX} = support_{tX} = \min_{\forall\, x \in X} (\max_{\forall\, a \in t}(\mu_{xa})) \quad (2)$$

O resultado da Equação 2 permite verificar o grau em relação a uma única transação, nesse caso, um operador $\sum count$ foi usado para somar o grau associado com cada uma das transações em *T,* como a seguir:

$$\sum_{\forall\, t \in T} count\ (support_{tX}) = \sum_{\forall\, t \in T} count\ (\mu_{tX}) \quad (3)$$

Dessa forma, o suporte de uma regra generalizada $X \rightarrow Y$, onde $X \cup Y = Z \subseteq I$, pode ser obtido conforme a Equação 4, onde $|T|$ é o total de transações da base de dados:

$$\sum_{\forall\, t \in T} count\ (\mu_{tZ}) / \ |T| \quad (4)$$

Similarmente, a confiança estendida (*Dconfidence)* de $X \rightarrow Y$ pode ser obtida conforme a seguir:

$$\sum_{\forall\, t \in T} count\ (\mu_{tZ}) / \sum_{\forall\, t \in T} count\ (\mu_{tX}) \quad (5)$$

É importante salientar que em Wei e Chen (1999) apenas as definições foram apresentadas, e somente em Chen e Wei (2002) os autores propuseram o algoritmo

para realizar a mineração. No entanto, Chen e Wei apresentam dois algoritmos, um para trabalhar com as taxonomias mencionadas anteriormente, e outro para trabalhar com *hedges linguísticos* nas regras.

De acordo com os autores, *Hedges linguísticos* são expressões linguísticas utilizadas para representar valores nebulosos, como “muito”, “mais ou menos”, “tipo de”, modificando o significado do termo linguístico que o segue. Segundo eles, o uso dessas expressões torna o conhecimento mais compreensível, enriquecendo a semântica das regras obtidas. Por exemplo, *“roupas muito caras → frutas tropicais”*, com termos linguísticos (“roupa cara” e “fruta tropical”) e o *hedge linguístico* (“muito”). Para que a utilização desses termos fosse possível, eles foram incluídos na taxonomia. Além disso, as estruturas também consideravam diferentes graus de especialização/generalização entre os elementos.

O primeiro algoritmo recebeu o nome de *FGAR*, e é responsável por realizar a mineração utilizando taxonomias difusas com graus de pertinência variando em [0,1]. Trata-se de uma extensão do algoritmo clássico proposto Srikant e Agrawal (1995). O segundo, denominado *HFGAR*, foi proposto para realizar a mineração utilizando taxonomias com os *hedges linguísticos*. Em ambos, os autores consideraram a inclusão de medidas de interesse, pois como as regras são geradas a partir de transações estendidas, os padrões redundantes devem ser podados.

Keon-Myung (2001) apresenta um trabalho semelhante ao de Chen e Wei, (2002), porém relacionado à mineração de regras de associação quantitativas. O autor destaca a utilização de dois tipos diferentes de taxonomias fuzzy: hierarquias de conceitos fuzzy e hierarquias de generalização de termos linguísticos *fuzzy*. Na primeira, um conceito possui relacionamento de generalização parcial com outros conceitos mais gerais, incluindo graus de especialização/generalização variando em [0,1], ou seja, trata-se das estruturas introduzidas por Wei e Chen (1999). Na segunda, nós de nível mais alto representam termos linguísticos fuzzy mais gerais, semelhante ao que foi apresentado por Chen e Wei (2002), incluindo também graus variando em [0,1].

Portanto, Keon-Myung (2001) introduz um algoritmo para mineração de regras generalizadas difusas quantitativas, utilizando as estruturas mencionadas. A diferença desse trabalho, em relação aos algoritmos descritos na seção 5.2, é que as regras podem ser compostas por itens presentes em qualquer nível de uma estrutura taxonômica. Dessa forma, as taxonomias de conceitos fuzzy são utilizadas

para generalizar atributos categóricos, e as hierarquias de termos linguísticos difusos são utilizadas para generalizar atributos quantitativos.

A representação de atributos categóricos, nas taxonomias de conceitos fuzzy, é feita conforme apresentado na seção 2.5. No caso de atributos quantitativos, eles são particionados em vários intervalos (conjuntos) e associados a termos linguísticos difusos (como descrito na seção 5.2), e cada um desses termos é então representado na estrutura, de forma que os nós mais altos sejam termos linguísticos mais gerais.

Para dois termos linguísticos A e B, o autor diz que A é mais geral que B quando sua função de pertinência $\mu_A(x)$ é maior ou igual a função de pertinência $\mu_B(x)$ de B. Portanto, a disposição hierárquica dos termos na taxonomia é feita com base nesse critério. É importante ressaltar que, nesse algoritmo, ambas as estruturas utilizadas possuem graus de especialização/generalização variando em [0,1].

Assim como em Chen e Wei (2002), a técnica utilizada na geração das regras é feita como proposto em Srikant e Agrawal (1995), utilizando transações estendidas. Além disso, Keon-Myung (2001) também utiliza medida de interesse para podar regras redundantes. Entretanto, vale ressaltar que em Chen e Wei (2002) os algoritmos foram desenvolvidos para trabalhar com atributos binários, e não com atributos categóricos ou quantitativos.

De acordo com Wen-Yang, *et al.* (2010), a maioria dos trabalhos envolvendo o problema de minerar regras generalizadas em hierarquias fuzzy requer que as mesmas sejam estáticas, ignorando o fato de que elas não necessitam ser mantidas necessariamente inalteradas. Por exemplo, alguns itens podem ser reclassificados, sendo retirados da taxonomia, ou então adicionados como itens novos. Consequentemente, os graus de pertinência também necessitam ser reajustados.

Com base nessa circunstância, os autores propuseram uma abordagem na qual o conjunto final de regras geradas pode ser atualizado de acordo com a evolução das estruturas, ou seja, à medida que as mesmas são alteradas, novos resultados podem ser obtidos. A atualização do resultado é feita de modo que o algoritmo não precise refazer todo o seu processamento. A evolução das taxonomias difusas pode ocorrer devido a quatro causas básicas: inserção, deleção, renomeação e reclassificação de itens na estrutura.

Sendo assim, tratando-se da obtenção de regras generalizadas utilizando taxonomias difusas, os trabalhos descritos até o momento são os mais relevantes. Entretanto, a maioria deles está focada na extração de regras generalizadas compostas por termos linguísticos fuzzy, chamadas de regras generalizadas difusas.

Outro ponto a ser destacado é que esses trabalhos se inserem no contexto de regras generalizadas quantitativas. De acordo com a seção 5.2, o processo de mineração de regras de associação em tabelas de domínio não binário, denominado mineração de regras de associação quantitativas, é aplicado em situações nas quais os atributos podem ser quantitativos ou categorizados.

Portanto, a diferença das pesquisas apresentadas nesta seção em relação às que foram descritas na seção 5.2, está na utilização de taxonomias para generalização das regras. Além disso, ao invés de utilizarem intervalos crisp, eles utilizam intervalos difusos, definidos por funções de pertinência. Os intervalos difusos são rotulados com termos linguísticos fuzzy, tais como *jovem, alto,* e outros, que são utilizados nas regras. Isso torna possível produzir padrões mais descritivos.

Hung-Pin, *et al.* (2006), por exemplo, propõem um algoritmo baseando-se no relacionamento hierárquico e em intervalos fuzzy, e apresentam um método denominado CBFAR (cluster-based fuzzy association rules) que gera intervalos difusos através de clusterização e varre o banco de dados uma única vez para geração de candidatos. Nesse trabalho, itens ancestrais da taxonomia crisp são inseridos nas transações da base de dados, para que o algoritmo Apriori possa gerar as regras.

Em Mahmoudi, *et al.* (2011) foi proposto outro algoritmo para mineração de regras generalizadas em atributos quantitativos. O artigo propõe, em etapa de pré-processamento, um método para gerar o suporte mínimo automaticamente, de acordo com as necessidades do usuário. O processo de mineração é baseado na abordagem de Jiawei Han e Fu (1995), e um conjunto de funções de pertinência é utilizado para transformar as transações quantitativas em intervalos difusos. Além disso, para que seja possível obter conhecimento generalizado, taxonomias crisp são utilizadas.

Ainda nesse contexto, trabalhos semelhantes podem ser encontrados em Cai, *et al.*, (1998), Hong, *et al.*, (2003) e Lee, *et al.*, (2008). Nos dois primeiros, o algoritmo para geração das regras é baseado na abordagem de Srikant e Agrawal, (1995); no segundo, os autores se baseiam no método proposto em Jiawei Han e Fu

(1995). Em todos eles, são utilizadas função de pertinência para transformar os atributos quantitativos em valores difusos, e as estruturas utilizadas também são crisp.

Além do exposto, existem propostas focadas em semântica, utilizando ontologias para extrair associações de similaridade existentes entre os itens da base de dados. Nesses trabalhos, as relações mencionadas são representadas nas folhas da ontologia. Em Escovar, Yaguinuma e Biajiz (2006), por exemplo, foi proposto o algoritmo *XSSDM*, que gera regras generalizadas compostas por itens difusos (associações de similaridade). Durante o processo de mineração, caso o grau de similaridade entre os itens seja maior ou igual a um valor de similaridade mínima definida, é detectada uma associação de similaridade, significando que os itens nessa associação são suficientemente similares, podendo representar um conhecimento interessante ao usuário.

O algoritmo *NARFO* (*Non-redundant and Generalized Association Rules Based on Fuzzy Ontologies*) proposto por Miani (2009) também é um algoritmo que utiliza ontologias para gerar regras de associação generalizadas com similaridade. Trata-se de uma extensão do algoritmo *XSSDM*, incluindo alguns tratamentos de redundância, um parâmetro *mingen* e um processo de generalização de *itemsets* não frequentes. O algoritmo é dividido em oito fases, conforme apresentado na Figura 5.2, em algumas dessas fases é utilizado o raciocinador da ontologia difusa, que tem como principal objetivo buscar relações, conceitos e inferências dos itens presentes na ontologia difusa. As contribuições do *NARFO* em relação em relação ao *XSSDM* podem ser vistas nos pontos acinzentados da Figura 5.2.

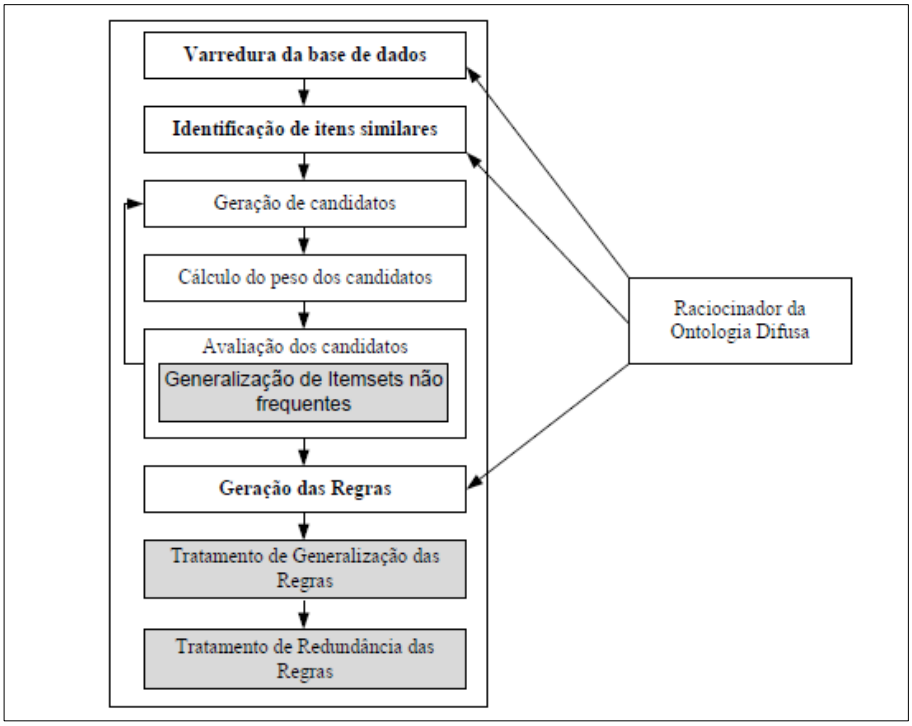


**Figura 5.2 - Etapas do *NARFO* Miani (2009).**

## 5.4 Considerações Finais

De acordo com a seção anterior, são poucos os trabalhos que realizam a mineração de regras generalizadas utilizando taxonomias fuzzy com graus de especialização/generalização variando em [0,1]. A maioria dos trabalhos está inserida na linha de mineração de regras generalizadas difusas, que é um conceito diferente, uma vez que, nesse caso, o foco está na obtenção de regras compostas por termos linguísticos difusos, e as transações exploradas são compostas por atributos quantitativos.

Na seção 5.2 foram apresentados alguns trabalhos que utilizam a lógica fuzzy na obtenção de regras de associação quantitativas. Nesses casos, não são utilizadas estruturas taxonômicas, pois o objetivo não é a geração de regras generalizadas, e sim regras compostas por termos imprecisos.

# Capítulo 6

# TRABALHOS DESENVOLVIDOS

## 6.1 Considerações Iniciais

No Capítulo 5 enfatizou-se que existem poucos trabalhos relacionados ao de Wei e Chen (1999), no qual são explorados diferentes graus em relacionamentos de especialização/generalização na estrutura taxonômica. Além disso, é possível perceber que a maioria dos algoritmos existentes está incluída no contexto de regras generalizadas compostas por termos linguísticos fuzzy. Nesses casos, as taxonomias utilizadas não são fuzzy, mas crisp.

Além disso, também é possível verificar um viés nas abordagens, que é a exploração das estruturas taxonômicas no estágio de pré-processamento do algoritmo, durante a geração de transações estendidas. Nesse sentido, considerando o uso de estruturas nebulosas introduzidas por Wei e Chen (1999), pode-se afirmar que, até o momento, nenhum trabalho foi desenvolvido para realizar a generalização em fase de pós-processamento.

No entanto, um grande desafio de obter regras generalizadas nessa etapa é a realização de cálculos de suporte e confiança das regras, estendidos ao contexto fuzzy, evitando varreduras desnecessárias no banco de dados durante o cálculo, de de forma a contribuir com a redução do tempo de execução do algoritmo.

Por conseguinte, existem duas outras motivações para generalizar na etapa mencionada: Primeira: é a redução do montante de regras geradas, uma vez que trabalhos crisp anteriores mostram que isso é possível. Segunda: é a eliminação de redundância sem a necessidade de utilizar medidas de poda, que em geral são

muito subjetivas, pois dependem de valores escolhidos pelo usuário e, dependendo do valor escolhido, não garantem uma grande redução do volume de regras geradas.

Sendo assim, considerando o que foi descrito nos capítulos anteriores, e os pontos destacados nesta seção, na seção 6.2 serão apresentados os trabalhos desenvolvidos por esta pesquisa.

## 6.2 Algoritmo FOntGAR para Mineração de Regras de Associação Generalizadas Utilizando Ontologias de Conceitos Fuzzy e Similaridade Baseada em contexto

O principal objetivo do algoritmo FOntGAR (*Fuzzy Ontology-based Generalized Association Rules*) é realizar a generalização de regras de associação em etapa de pós-processamento, de modo que seja possível obter um conjunto final de regras mais reduzido, não redundante e com maior semântica, facilitando a compreensão do usuário. Além disso, as ontologias exploradas são fuzzy, possuindo graus de especialização/generalização variando em [0,1], e relações de similaridade baseadas em contexto.

### 6.2.1 Ideias Principais

O processo de geração de regras tradicionais é baseado no algoritmo Apriori (AGRAWAL; SRIKANT, 1994) e, portanto, requer parâmetros de suporte mínimo (minsup) e confiança mínima (minconf). Além disso, o algoritmo também utiliza os parâmetros *(mingen)* de generalização mínima, *(minsim)* de similaridade mínima, (*context*) indicando o contexto explorado, e o parâmetro *side*. O último indica o lado em que ocorrerá a generalização. O uso do *minsin* e do *mingen* foi introduzido por Escovar (2004) e Miani (2009), respectivamente.

Todos os parâmetros mencionados são providos pelo usuário, e o minsup, minconf, mingen e minsin são expressos por um valor real no intervalo [0,1]. O parâmetro *side* é expresso por uma string *left, rigth,* ou *lr*, indicando, respectivamente, se a generalização ocorrerá no lado esquerdo, direito ou em

ambos. Em relação ao lado de generalização explorado, foi levada em consideração a definição apresentada por Srikant e Agrawal (1995), que diz que uma regra generalizada pode conter itens ancestrais em qualquer um dos lados mencionados. O lado *left* indica relação entre classes e itens especializados, e o lado *right* indica relação entre itens especializados e classes de itens. O parâmetro *lr* indica relação entre classes.

As relações de similaridade são representadas nas folhas da ontologia difusa, e relações com grau de similaridade maior ou igual ao minsim podem ser inseridas nas regras geradas, aumentando o enriquecimento semântico das mesmas. Neste trabalho, diferente do que foi proposto por Escovar (2004) e Miani (2009), não ocorre à utilização de itens difusos, nem tampouco a inclusão dos mesmos no conjunto de itemsets candidatos. O algoritmo apenas utiliza o reasoner da ontologia para inferir essas relações e, após isso, tenta inserir as mesmas em regras, adicionando mais semântica a elas. Além disso, também é utilizada a questão da representação de contexto, conforme introduzido por Cerri, e*t al.* (2010).

A generalização é feita com base em uma metodologia de agrupamento e substituição, onde são gerados vários grupos contendo duas ou mais regras com características similares, de modo que os padrões desses grupos possam ser substituídos por um único padrão generalizado, reduzindo-se o montante inicial utilizado no processo. Como vários grupos podem ser gerados, várias regras generalizadas podem ser obtidas. Um ponto a ser destacado é que a generalização pode ocorrer sem que todos os descendentes de um ancestral estejam presentes em um grupo, mas somente se o parâmetro minGen for satisfeito.

Por exemplo, considere o valor do mingen sendo 0.6 (60%), e suponha que um ancestral “pão” é representado em uma taxonomia por três pães diferentes $pão_1$, $pão_2$, $pão_3$, e um ancestral “leite” é representado por cinco tipos de leites diferentes, $leite_1$, $leite_2$, $leite_3$, $leite_4$ e $leite_5$. Sendo assim, uma regra generalizada “leite → pão” poderá ser obtida mesmo que nem todos os tipos de “leite” estejam presentes no grupo analisado, mas somente se 60% dos descendentes desses ancestrais estiverem presentes no mesmo.

Dessa forma, deve-se considerar que (para valores inferiores a 100%) o uso do minGen afeta a semântica dos padrões generalizados, pois eles não representam todos os seus descendentes. Sendo assim, para contornar a situação, e também auxiliar a compreensão do usuário, o algoritmo mostra os itens que não participaram

do processo de generalização. Por exemplo, supondo que o item $leite_5$ não está presente no grupo em questão, a regra generalizada será apresentada ao usuário da seguinte maneira “leite → pão (-$leite_5$)”, indicando que o item $leite_5$ não contribuiu para a generalização.

Voltando ao detalhamento do algoritmo FOntGAR, em relação as ontologias fuzzy exploradas, a inclusão dos graus de especialização foi realizada de forma manual, com base nas metaontologias descritas na seção 4.5 do Capítulo 4, que permitem representar graus de especialização/generalização variando em [0,1], e diferentes contextos na relação de similaridade. A Figura 6.1 ilustra todos os passos do algoritmo proposto, sendo que os pontos coloridos em cinza são as principais contribuições exploradas.

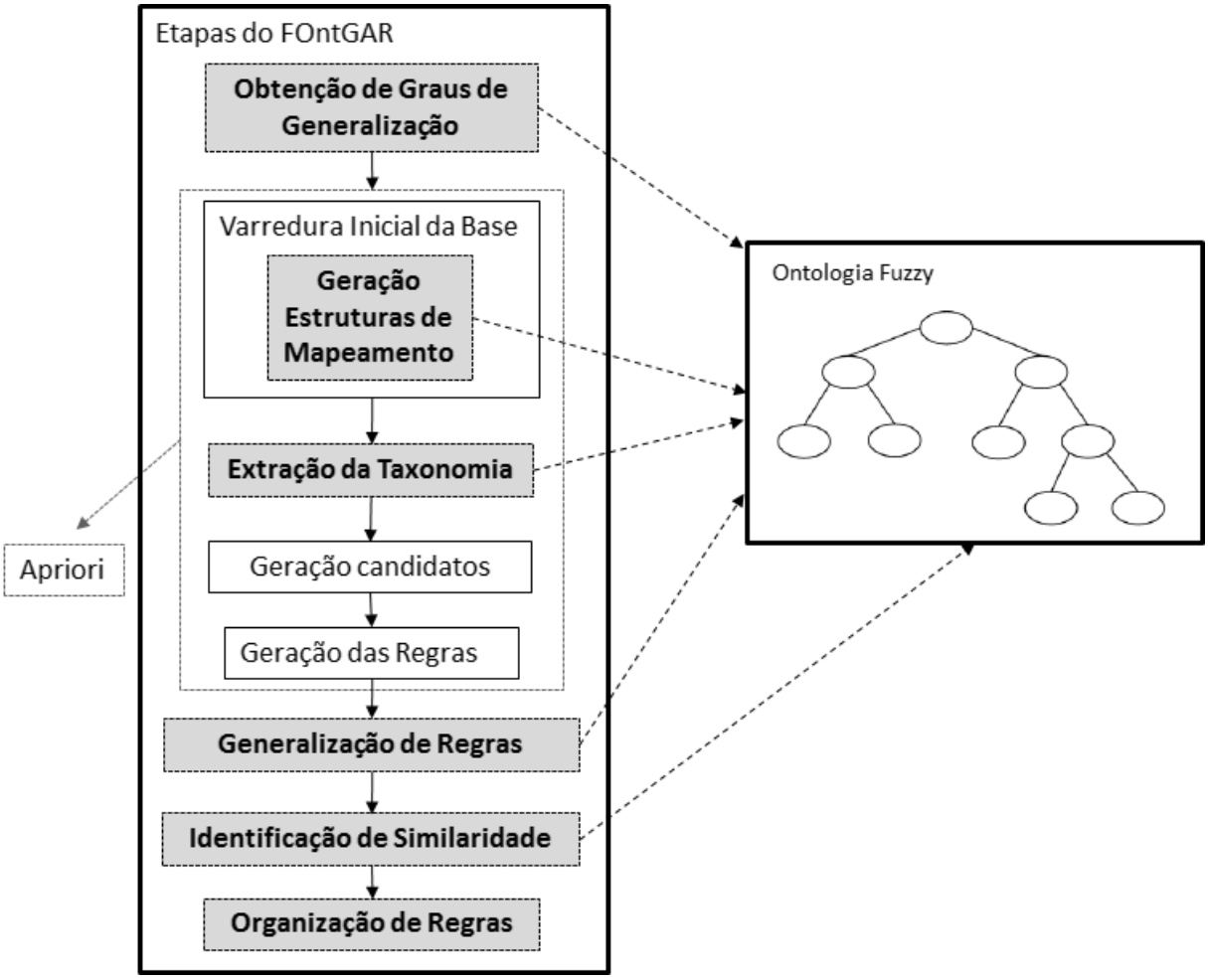


**Figura 6.1 – Etapas do Algoritmo FOntGAR.**

### 6.2.2 Abordagem de Agrupamento Proposta

A abordagem de agrupamento utilizada pelo algoritmo FOntGAR é utilizada durante o tratamento de generalização, portanto, antes das regras serem generalizadas, é necessário que elas sejam agrupadas.

A Figura 6.2 ilustra a generalização de “camiseta → boné” e “bermuda → boné”, com base na taxonomia *T1*, onde os itens “camiseta” e “bermuda” são substituídos por “roupas leves”, gerando-se a regra generalizada “roupas leves → boné”. A partir dessa ilustração, dois pontos importantes podem ser destacados. Primeiro: em relação as suas estruturas sintáticas, as duas regras analisadas possuem o mesmo item consequente. Segundo: em relação à taxonomia *T1,* os dois padrões possuem o mesmo ancestral (roupas leves) no lado onde ocorre a generalização. Portanto, os dois padrões analisados são similares.

Semelhantemente, a Figura 6.3 ilustra um caso onde um conjunto de quatro regras é substituído por uma única regra geral. Em relação às características compartilhadas, todas possuem o mesmo consequente (boné) e, além disso, os ancestrais relativos ao lado da generalização (roupas leves e calçados) são idênticos em todos os padrões.

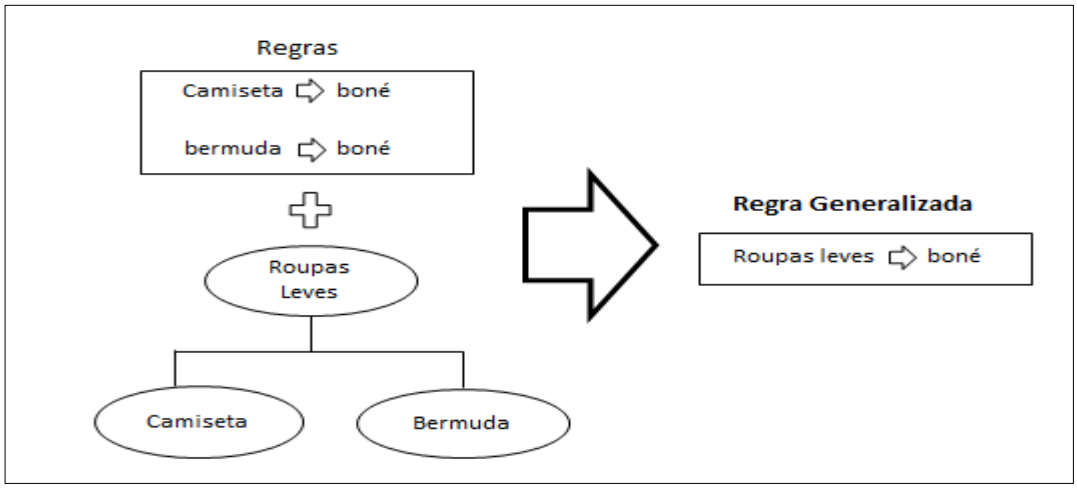


**Figura 6.2 – Generalização de Regras de Associação usando Agrupamento.**

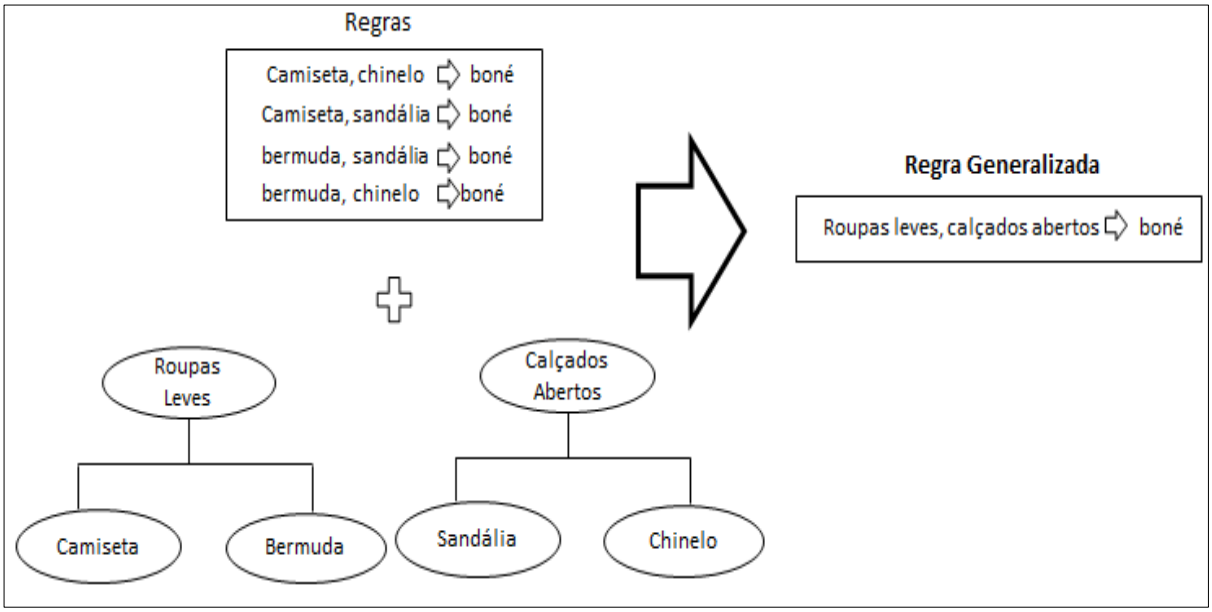


**Figura 6.3 – Generalização de regras utilizando agrupamento por similaridade e ancestrais da estrutura taxonômica.**

O objetivo das ilustrações é demonstrar que regras de associação similares podem ser agrupadas e posteriormente generalizadas. Em cada um dos exemplos, é possível observar um único grupo de padrões com características em comum. Nos casos apresentados, elas possuem o mesmo consequente, e os ancestrais dos itens no lado antecedente, que é onde ocorre a generalização, são os mesmos em todos os padrões. Embora essas ilustrações incluam poucos padrões, sabe-se que algoritmos de associação normalmente geram grandes quantidades. Portanto, considerando a ideia de agrupamento, vários conjuntos de regras com características em comum podem ser obtidas, e consequentemente diversas regras generalizadas podem ser geradas.

No algoritmo FOntGAR, quando o usuário define o antecedente ou o consequente (*lef* ou *right*) como lado a ser generalizado, o agrupamento ocorre da seguinte maneira: Compara-se o conjunto total de regras extraído, verificando quais possuem os mesmos ancestrais no lado da generalização e, ao mesmo tempo, tenham itens idênticos no lado oposto ao que será generalizado.

Em relação a isso, como podem existir muitas variações, diversos grupos de regras com as características mencionadas podem ser formados. Entretanto, um grupo só existirá se puder abrigar mais de uma regra, pois como o resultado posterior ao agrupamento é a generalização, e esta visa à diminuição do montante de padrões, em um grupo com apenas um padrão não existe redução de quantidade.

Por exemplo, considerando a situação de pães e leites, apresentada na seção anterior, onde pão é um $pão_1$, $pão_2$, $pão_3$, $pão_4$ e leite é um $leite_1$, $leite_2$, $leite_3$, $leite_4$ e $leite_5$. Suponha que, o parâmetro "*side"* tenha sido definido como *"right"* (generalização no consequente), e que durante a etapa de extração o algoritmo tenha gerado os seguintes padrões:

- $leite_1, leite_4 \rightarrow pão_1, pão_4;$
- $leite_1, leite_4 \rightarrow pão_2;$
- $leite_1, leite_4 \rightarrow pão_3;$

Comparando as mesmas, é possível notar que elas possuem, simultaneamente, o ancestral (pão) no lado da generalização, e itens idênticos no antecedente, que é o lado oposto ao que será generalizado. Portanto, como mais de uma regra compartilha essas características, todas elas são inseridas em um único grupo. O agrupamento é feito para facilitar a generalização, pois seria difícil encontrar a similaridade mencionada em meio a um conjunto desorganizado de padrões gerados. A Figura 6.4 ilustra o processo utilizado no agrupamento das regras, usando tanto a generalização no antecedente quanto no consequente.

Por outro lado, se o usuário define o parâmetro "*side"* como sendo *lr,* então a generalização deverá ocorrer nos lados antecedente e consequente simultaneamente. A ideia utilizada é muito parecida com a descrita anteriormente, entretanto, apenas os ancestrais dos itens das regras devem ser analisados durante o agrupamento.

O agrupamento ocorre da seguinte forma: comparam-se todas as regras geradas, verificando se existem dois ou mais padrões com os mesmos ancestrais no lado antecedente e, ao mesmo tempo, ancestrais idênticos no lado consequente. Por exemplo, supondo que o parâmetro tenha sido definido como *"lr"* (generalização antecedente/consequente), e que durante a etapa de extração o algoritmo tenha gerado, dentre outras regras, os seguintes padrões:

- $leite_1, leite_4 \rightarrow pão_1, pão_4;$
- $leite_1, leite_2 \rightarrow pão_2;$
- $leite_1, leite_3 \rightarrow pão_3$

Ao realizar a comparação, verifica-se que todas elas possuem o mesmo ancestral (pão) no lado da generalização e, ao mesmo tempo, possuem o ancestral (leite) no antecedente. Portanto, como existem três regras com essas características, elas são agrupadas em um único grupo.

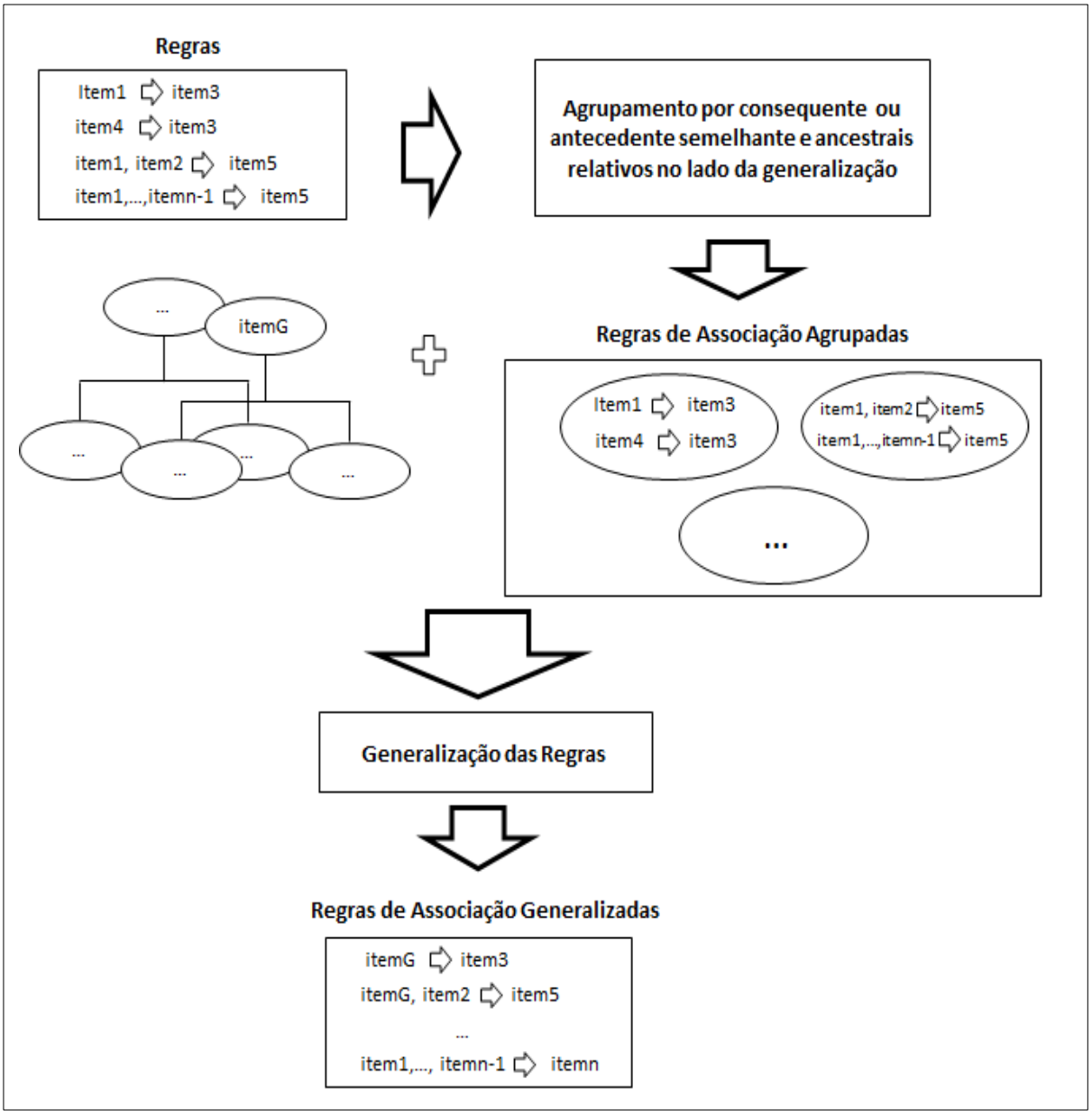


**Figura 6.4 – Processo utilizado no agrupamento de regras de associação (generalização no antecedente ou consequente).**

Entretanto, considerando a possibilidade de relação parcial entre itens da base e estrutura taxonômica, o agrupamento também permite que os grupos possuam regras compostas por itens ausentes na estrutura, mas presentes no banco.

Por exemplo, considerando as regras:

- $leite_1, leite_4, \boldsymbol{tomate} \rightarrow pão_1, pão_4$;
- $leite_1, leite_2 \rightarrow pão_2, \boldsymbol{queijo}$;
- $leite_1, leite_3 \rightarrow pão_3$

Supondo que os itens “tomate” e “queijo” não fazem parte da ontologia, e que o parâmetro *side* seja *“lr”* (generalização nos dois lados), as regras acima não

podem estar em um mesmo grupo, pois como “tomate” aparece apenas na primeira, e “queijo” apenas na segunda, esse grupo não poderá derivar uma generalização $leite$, $\boldsymbol{tomate} \rightarrow pão$, $\boldsymbol{queijo}$. Sendo assim, elas só poderiam ser agrupadas se o item “tomate” ocorresse em todos os antecedentes, e “queijo” em todos os consequentes.

Considere agora o parâmetro de generalização sendo “*left*” (generalização antecedente), e as seguintes regras:

- $leite_1, leite_4, \boldsymbol{tomate} \rightarrow pão_1$;
- $leite_1, leite_2, \boldsymbol{tomate} \rightarrow pão_1$;
- $leite_1, leite_3, \boldsymbol{tomate} \rightarrow pão_1$;

Mesmo que “tomate” não esteja presente na ontologia e, portanto, não possa ser generalizado, as regras acima poderão ser agrupadas, pois esse item aparece em todos os antecedentes. Portanto, após o agrupamento, as regras poderiam ser substituídas por uma única generalizada: “$leite$, $\boldsymbol{tomate} \rightarrow pão_1$”. As Figuras 6.5 e 6.6 apresentam os pseudocódigos utilizados no agrupamento.

```
1   conjuntoRegras:= regras a serem agrupadas;
2   ladoGeneralizacao:= valor do parâmetro side;
3   se (ladoGeneralização = "lr") então
4     para cada regra em conjuntoRegras faça
5        receba uma regra de conjuntoRegras (regra comparada);
6        se ela possui itens sem ancestral então
7           receba os itens sem ancestral no lado antecedente;
8           receba itens sem ancestral no lado consequente;
9           para cada regra seguinte de conjuntoRegras faça
10            Verifique se ela possui os mesmos itens sem ancestral;
11            Se ela possui esses itens então
12              Verifique os ancestrais no antecedente;
13              Verifique os mesmos ancestrais no consequente;
14              Se estes são iguais ao da regra comparada então
15                Agrupe essa regra com a comparada;
16              Fim se;
17            Fim se;
18          Fim para;
19       Fim se;
20       Se ela não possui itens sem ancestral então
21         Para cada regra seguinte de conjuntoRegras faça
22           Verifique os ancestrais no antecedente;
23           Verifique os ancestrais no consequente;
24           Se estes são iguais ao da regra comparada então
25             Agrupe essa regra com a comparada;
26           Fim se;
27         Fim para;
28       Fim se;
29  Fim se;
```

**Figura 6.5 – Pseudocódigo para generalização antecedente/consequente.**

```
1   conjuntoRegras:= regras a serem agrupadas;
2   ladoGeneralizacao:= valor do parâmetro side;
3   se (ladoGeneralização = "left" ou "right") então
4      para cada regra em conjuntoRegras faça
5         receba uma regra de conjuntoRegras (regra comparada);
6         se ela possui itens sem ancestral então
7            receba os itens sem ancestral no lado antecedente;
8            receba itens sem ancestral no lado consequente;
9            para cada regra seguinte de conjuntoRegras faça
10            se ela possui os mesmos itens sem ancestral então;
11              se ela possui os mesmos ancestrais em ladoGeneralização;
12               se ela possui o mesmo lado oposto ao ladoGeneralização;
13                  Agrupe essa regra com a comparada;
14            fim se;
15           fim para;
16        fim se;
17        Se ela não possui itens sem ancestral então
18          para cada regra seguinte em conjuntoRegras faça
19            se ela possui os mesmos ancestrais em ladoGeneralização;
20             se ela possui o mesmo lado oposto ao ladoGeneralização;
21               Agrupe essa regra com a comparada;
22            fim se;
23           fim para;
24        fim se;
25     fim para;
26  fim se;
```

**Figura 6.6 – Pseudocódigo para generalização antecedente ou consequente.**

### 6.2.3 O Algoritmo Passo a Passo

Conforme mencionado anteriormente, o algoritmo FOntGAR utiliza estruturas taxonômicas onde os graus de especialização/generalização podem variar no intervalo [0,1]. Dessa forma, de acordo com Wei e Chen (1999), os cálculos de suporte e confiança devem ser estendidos, de forma a considerar os valores dos graus mencionados. Conforme apresentado no Capítulo 5, o grau de um termo *x* em relação ao seu ancestral *y* pode ser calculado usando a combinação *max-min*, através da Equação 1 (seção 5.3):

$$\mu_{xy} = \max_{\forall l:\, x \to y} \left( \min_{\forall e \text{ on } l} \mu_{le} \right)$$

Onde $l: x \to y$ representa um dos caminhos entre *x* e *y*, $e$ em $l$ é uma das arestas do acesso $l$, $\mu_{le}$ é o grau na aresta $e$ em $l$. Se não houver nenhum acesso entre *x* e *y*, $\mu_{xy} = 0$. Assim, pela Figura 6.7, existem dois caminhos possíveis entre tomate e Hortifrúti: tomate → Fruta → Hortifrúti, com grau mínimo do caminho = 0.7, e tomate → Legume → Vegetal → Hortifrúti, com grau mínimo do caminho = 0.6, portanto, tomate é um Hortifrúti com grau 0.7, pois max(0.7, 0.6) = 0.7.

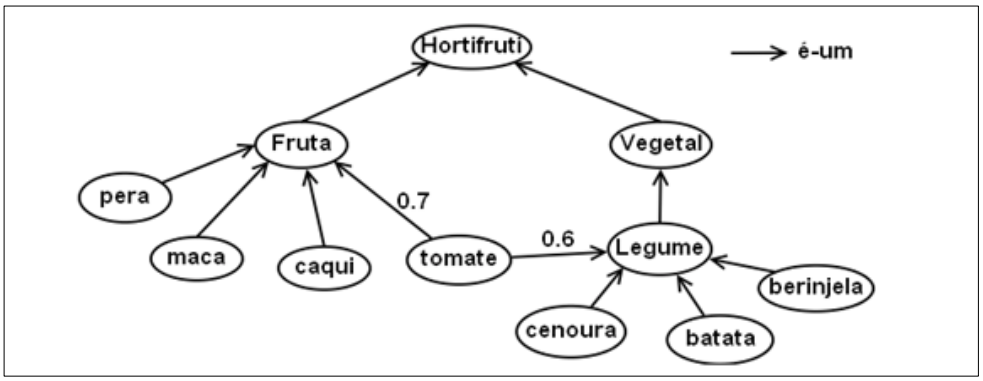


**Figura 6.7 - Exemplo de Figura e Legenda (Yaguinuma, 2007).**

Considere um exemplo, extraído de Wei e Chen (1999). Suponha um banco de dados de um supermercado, como mostrado na Tabela 6.1, o suporte mínimo 30% (duas transações), a confiança mínima é 60%, e uma taxonomia fuzzy, conforme apresentado na Figura 6.8. De acordo com a Equação 1, são encontrados os graus dos nós folhas em relação aos seus ancestrais, que são os valores da tabela 6.2. Por exemplo, o grau de Tomate em relação a Pratos Vegetais é calculado como a seguir:

$$\mu(Tomate \in Pratos\ vegetais) = max(min(1, 0.7), min(1, 0.3)) = 0.7.$$

**Tabela 6.1 – Exemplo Simples de Base Transacional.**

| Transação | Itens |
|---|---|
| #100 | Maçã |
| #200 | Tomate, carneiro |
| #300 | Repolho, carneiro |
| #400 | Tomate, porco |
| #500 | Porco |
| #600 | Repolho, porco |

**Tabela 6.2 – Representação dos Graus entre Folhas e Ancestrais.**

| Nós folhas | Graus entre eles e os ancestrais |
|---|---|
| Maçã | 1/maçã, 1/fruta, 1/pratos vegetais |
| Tomate | 1/tomate, 0.3/vegetal, 0.7/fruta, 0.7/pratos vegetais |
| Repolho | 1/repolho, 1/vegetal, 1/pratos vegetais |
| Porco | 1/porco, 1/carne |
| Carneiro | 1/carneiro, 1/carne |

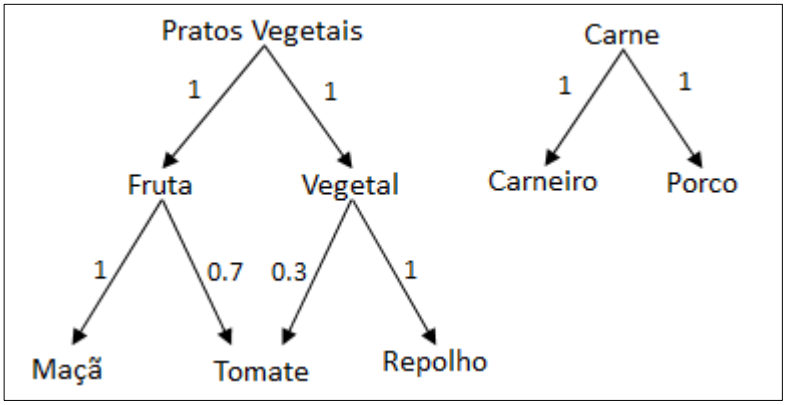


**Figura 6.8 – Taxonomia Simples de Domínio Alimentício.**

Utilizando os conceitos apresentados, antes de iniciar a geração de candidatos, FOntGAR utiliza o reasoner da ontologia difusa para obter os graus dos nós folha em relação a todos os seus ancestrais. Esse processamento é realizado em etapa inicial para que na etapa subsequente seja possível obter o grau em que cada transação do banco de dados suporta um ancestral da ontologia, e posteriormente o suporte e confiança estendidos das regras generalizadas possam ser calculados.

As etapas de geração de candidatos, e geração das regras tradicionais são efetuados como no algoritmo *Apriori*, entretanto, durante a varredura inicial da base (que ocorre na etapa de geração de candidatos), o algoritmo converte a mesma do formato horizontal para um formato vertical, incluindo também os ancestrais da ontologia nessa conversão. Essa estratégia permite acesso indexado às ocorrências de um elemento. A Figura 6.9 ilustra uma base de dados em formato vertical.

Formato Horizontal

| Transação | Itens |
|---|---|
| #100 | Maçã |
| #200 | Tomate, carneiro |
| #300 | Repolho, carneiro, maçã |
| #400 | Tomate, porco |
| #500 | Porco, maçã |
| #600 | Repolho, porco |

Formato Vertical (Invertida)

| Itens | Transações |
|---|---|
| Maçã | #100, #300, #500 |
| Tomate | #200, #400 |
| Repolho | #300, #600 |
| Porco | #400, #500, #600 |
| Carneiro | #200, #300 |

**Figura 6.9 – Base de Dados em Formato Vertical.**

Além do exposto, o algoritmo FOntGAR suporta situações onde a relação entre itens da base e folhas da ontologia não é total, ou seja, situações em que nem toda folha da estrutura taxonômica representa um item da base de dados, e/ou nem todo item da base de dados possui um representante na estrutura. Nesse sentido, em relação à utilização do parâmetro de generalização mínima *mingen*, o processo de verificação do percentual de descendentes de cada ancestral foi adaptado ao novo conceito. Por exemplo, suponha que existam cinco tipos de leite descendentes do ancestral "leite" ($leite_1$, $leite_2$, $leite_3$, $leite_4$ e $leite_5$), mas apenas quatro estão presentes na base ($leite_1$, $leite_2$, $leite_3$, e $leite_4$). Como $leite_5$ não está no banco, ele jamais ocorrerá nas regras, sendo irrelevante a situação explorada. É como se $leite_5$ não existisse na estrutura, podendo inclusive ser removido da mesma.

Com base nesse exemplo, considerou-se que, em situações onde ocorra relação parcial, 100% dos descendentes de um ancestral devam ser apenas os elementos que, de fato, estão presentes no banco de dados. No exemplo anterior, teríamos que 100% dos ancestrais de "leite" são: $leite_1$, $leite_2$, $leite_3$, e $leite_4$, e não $leite_1$, $leite_2$, $leite_3$, e $leite_4$, $leite_5$, e isso faz total diferença na verificação do mingen. Nesse sentido, dependendo da complexidade, e também do tamanho da ontologia, tal verificação pode se tornar extremamente complexa, pois vários itens podem ser inúteis ao contexto.

Sendo assim, para facilitar a verificação mencionada, na varredura que é realizada no banco de dados durante a fase de geração de candidatos, o algoritmo armazena todos os diferentes itens do banco em um vetor de itens. Com base nesses elementos, e na estrutura taxonômica da ontologia difusa, por meio de uma estratégia "bottom-up" é gerada uma taxonomia reduzida, que corresponde à ontologia principal, mas possui apenas os elementos presentes no banco de dados. Por exemplo, considerando um item "a" presente no banco, verificam-se na ontologia quais são os ancestrais imediatos (pais) do mesmo, os ancestrais dos pais encontrados, e assim sucessivamente, até que se chegue à raiz. Isso é feito para todos os diferentes itens encontrados no banco de dados, que estão presentes do vetor de itens mencionado anteriormente.

A Figura 6.10 ilustra a ideia proposta. Os nós circulados representam itens ausentes no banco de dados. Portanto, note que, embora a estratégia de geração da taxonomia relativa seja bottom-up, ela permite que os itens irrelevantes não estejam presentes na nova estrutura. No entanto, é importante salientar que a taxonomia

relativa é utilizada apenas na verificação do parâmetro de generalização mínima (*mingen*) e, portanto não são inseridos os graus de especialização/generalização da ontologia difusa na mesma.

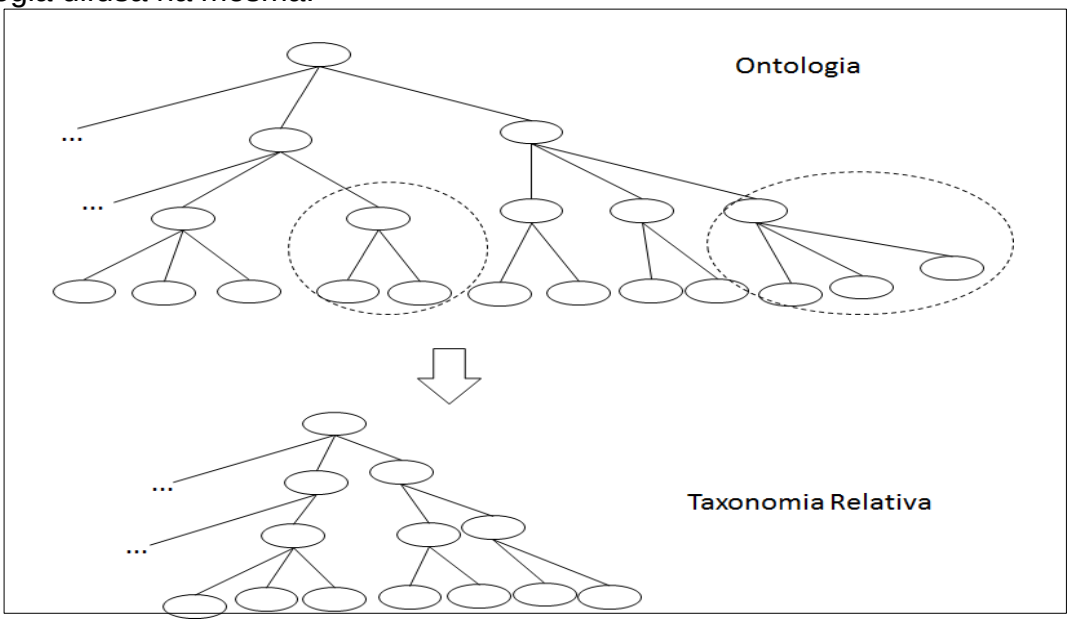


**Figura 6.10 – Redução de uma Ontologia a uma taxonomia relativa onde todos os itens folhas estrutura extraída estejam presentes na base de dados.**

Ao final do processo tradicional de extração de padrões, obtém-se o conjunto de regras especializadas que serão utilizadas no tratamento de generalização. Conforme dito, dentre outras coisas, o tratamento de generalização utiliza uma função de agrupamento groupingRules (linha 5 da Figura 6.12), que trabalha de acordo com a metodologia apresentada na subseção anterior. Sendo assim, as regras geradas, e o lado da generalização (valor do parâmetro *side*) são passados como parâmetros para a função mencionada.

Ao realizar o agrupamento, para cada grupo, FOntGAR guarda quais são os pais referentes ao antecedente, e consequente das regras do grupo, e também quais são os itens das regras do grupo (no antecedente, e no consequente) que não possuem pais, ou seja, ele já associa ao grupo sua regra generalizada correspondente (Figura 6.11). Essa regra será aqui denominada regra generalizada representante (linha 8), que é o padrão que irá substituir os demais.

Além disso, o tratamento de generalização também realiza os seguintes processamentos:

1 - verifica se o parâmetro *mingen* foi satisfeito (linha 9)*;*

2 - realiza duas verificações no padrão representante (substituto) (linha 10): Na primeira, é checado se os lados antecedente e consequente são disjuntos, e na

segunda é verificado se nenhum item consequente possui pai no lado antecedente, e vice-versa. Isso é feito para confirmar se os critérios conceituais de regras generalizadas estão sendo satisfeitos (seção 2.5).

Após as verificações mencionadas existem três ações possíveis de serem executadas, dependendo do resultado obtido:

1. Se essas verificações são satisfeitas (linha 11), então o cálculo de suporte da regra representante é realizado. Nesse caso, se a regra geral é frequente, então todas as regras do grupo são descartadas e, de fato, substituídas pela regra representante (generalizada), que é armazenada para posteriormente ser apresentada ao usuário.
2. Se as verificações são satisfeitas (linha 11), mas após o cálculo de suporte da regra generalizada verifica-se que a mesma não é frequente (linha 17), então a generalização não pode ocorrer. Nesse caso, as regras do grupo não são descartadas, pois poderão ser apresentadas ao usuário (linha 19).
3. Por outro lado, se as verificações apresentadas não forem atendidas, o suporte da regra representante não chega a ser calculado (linha 21), e as regras do grupo também não são descartadas. Portanto a generalização não ocorre.

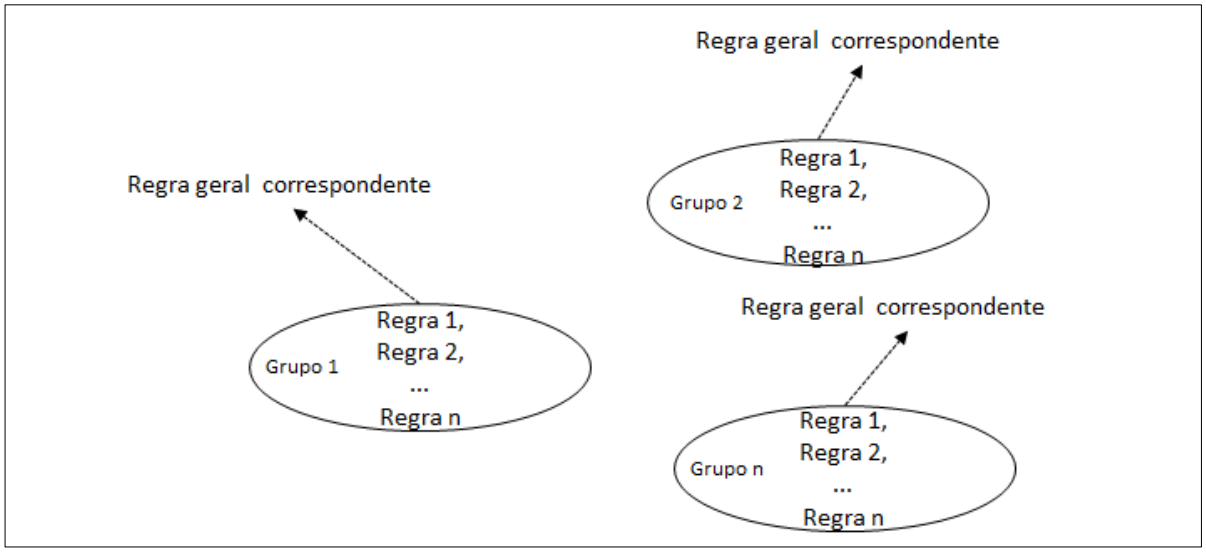


**Figura 6.11 – Ilustração de grupos de regras com uma regra mais geral representante.**

**Pseudocódigo do Tratamento de Generalização**

```
1 se (side = left) or (side = right) or (side = lr) então;
2      nível:= 1;
3      nonGeneralizedRules:= todas as regras tradicionais;
4     enquanto (nível < total de níveis) faça
5       groupingRules(nonGeneralizedRules e side);
6       groupedrules:= resultado de groupingRules;
7       para (cada grupo em groupedRules) faça
8           associa uma regra generalizada representante;
9          verifica se o minGen é satisfeito;
10         Verifica critérios conceituais de regras generalizadas;
11         Se (verificações são satisfeitas) então
12           Calcule o suporte estendido da regra representante;
13           Calcule a confiança estendida da regra representante;
14           Se (a regra geral é frequente)então
15             todas regras do grupo são substituídas pela regra geral;
16             generalizedRules:= a regra geral;
17           Se (a regra geral não é frequente)
18              generalização não pode ocorrer;
19              regras do grupo serão mostradas no resultado;
20         Fim se
21         Se (verificações não são atendidas)então
22           Generalização não pode ocorrer;
23           Se (nível = 1) então
24            regras do grupo serão mostradas no resultado;
25         Fim se;
26      fim para
27      se (generalizedRules possui regras generalizadas) então
28         nível: nível + 1;
29         para (cada regra de generalizedRules) faça
30           adiciona a regra em nonGeneralizedRules;
31         fim para;
32      fim se;
33      se (generalizedRules é vazio) então
34        break;
35    fim enquanto;
36 fim se;
```

**Figura 6.12 – Pseudocódigo Tratamento de Generalização.**

A generalização é feita nível a nível, ou seja, primeiro generaliza-se das folhas para os pais, e assim sucessivamente. Sendo assim, as regras generalizadas obtidas em cada nível são exploradas no nível subsequente de generalização, sendo que a cada iteração elas são fornecidas como parâmetro para a função groupingRules. O laço é repetido até que se chegue a um nível abaixo da raiz, ou até que não seja possível obter regras generalizadas em qualquer nível anterior.

Após o tratamento de generalização, o algoritmo usa o reasoner da ontologia para obter as relações de similaridade, inserindo as mesmas nos padrões gerados. Diferente do que é feito nos trabalhos relacionados, não ocorre inserção de itens difusos no conjunto de candidatos.

Por fim, após a identificação das similaridades, FOntGAR entra em seu estágio final, que é a organização do resultado. O conjunto final de regras é composto de padrões generalizados e tradicionais e, além disso, as regras são apresentadas ordenadamente de acordo com o nível de generalização. Sendo assim, o resultado é apresentado como a seguir:

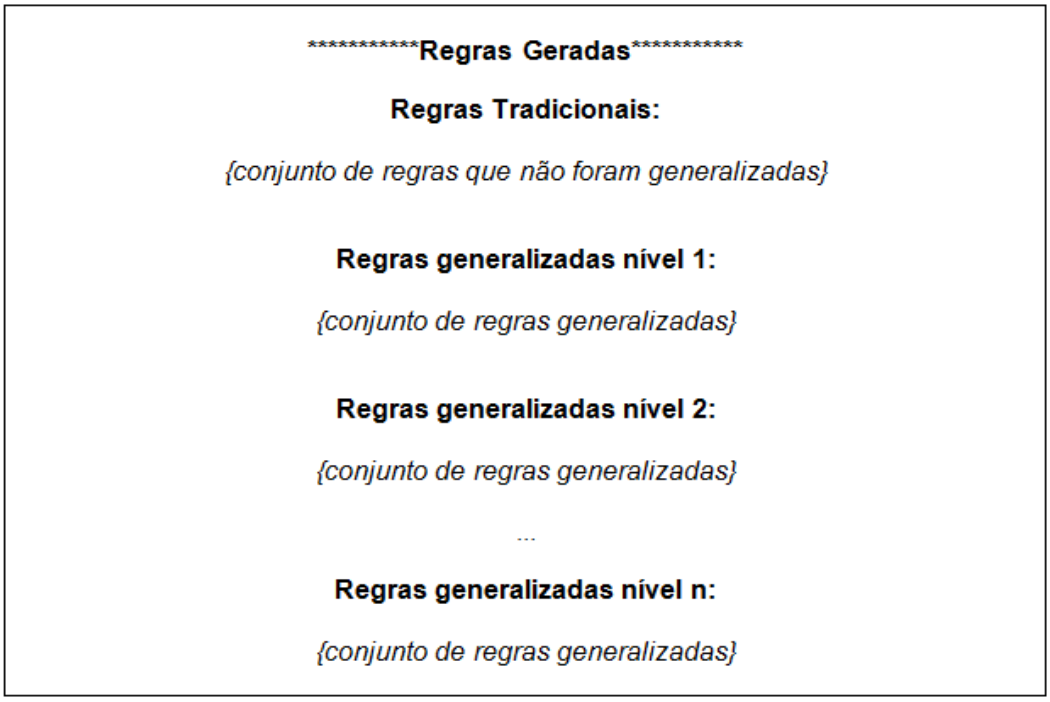


**Figura 6.13 – Formato em que o Resultado é Apresentado.**

### 6.2.4 Calculando Graus de Suporte e Confiança Estendidos no Pós-Processamento

Na seção 5.3 foram apresentadas as definições formais utilizadas no cálculo do suporte estendido, que é efetuado pela somatória dos graus que um conjunto de transações *T* suporta uma regra *X*. Toda regra *X* possui um formato itemset $\{x_1, \dots, x_n\}$ correspondente. Portanto, considerando a Figura 6.8, o banco de dados da Tabela 6.2, uma regra generalizada no formato "{vegetal, carne}" e os graus da Tabela 6.3, a soma dos graus das transações que suportam {vegetal, carne} é calculada como a seguir:

$$\min(0.3, 1) + \min(1, 1) + \min(0.3, 1) + \min(1, 1) = 2.6$$

Os graus de especialização/generalização são considerados apenas no suporte de regras generalizadas, pois nas demais o cálculo é feito de modo tradicional. Sendo assim, como no algoritmo FOntGAR a generalização ocorre

apenas no pós-processamento, o cálculo realizado na etapa de extração de padrões não é afetado.

Considerando a taxonomia difusa da Figura 6.8, $Fruta \rightarrow Carne$, por exemplo, seria uma regra generalizada e $\{Fruta, Carne\}$ é seu formato itemset. Como $\{Fruta, Carne\}$ é obtido e conhecido apenas no pós-processamento, para que fosse possível obter a somatória dos graus de transações que suportam $\{Fruta, Carne\}$ seria necessária uma nova varredura na base de dados. Com base nisso, como muitas regras generalizadas podem ser geradas, a quantidade de varreduras para computar o suporte de todas elas seria muito elevada e, dependendo da quantidade linhas da base, o desempenho do algoritmo certamente seria afetado.

Sendo assim, FOntGAR apresenta uma estratégia que permite atacar o problema do cálculo de suporte estendido, evitando a questão de varreduras excessivas na base de dados. Durante a varredura que ocorre na etapa de geração de candidatos, o algoritmo converte a base de seu formato horizontal para um formato vertical, incluindo também os ancestrais da ontologia nessa conversão. Ao mesmo tempo, o algoritmo calcula o grau que cada transação suporta os ancestrais da ontologia, armazenando-os para consultas posteriores. Portanto, duas estruturas compostas por chaves e valores são geradas nessa etapa. A Figura 6.14 representa a primeira, onde uma chave é um item da base de dados ou um ancestral da ontologia. Cada chave aponta para um valor, que é um vetor armazenando os identificadores das transações onde essa chave ocorre.

Sabendo que o suporte é calculado com base na Equação 2 do Capítulo 5, analisando a mesma, foi possível perceber que ela poderia ser decomposta em duas subpartes, denominadas aqui como Part1 e Part2, da seguinte maneira:

- **Part1** = $\max_{\forall\, a \in t}(\mu_{xa})$
- Parte2 = $\min_{\forall\, x \in X}(\boldsymbol{Part1})$

O principal objetivo do particionamento foi possibilitar que as duas subpartes envolvidas pudessem ser utilizadas em períodos diferentes do algoritmo. Assim, Part2 pode ser utilizada no pós-processamento, permitindo que o suporte de regras generalizadas seja calculado sem a necessidade de varreduras adicionais. Entretanto, para que isso ocorra, é necessário que o resultado de Part1 seja previamente armazenado.

Part1 retorna o grau que uma transação $t$ suporta um ancestral $x$. Como não se sabe quais regras generalizadas serão geradas, admite-se que o formato itemset das mesmas possa ser qualquer $X = \{x_1, \dots, x_n\}$, onde $X$ é o padrão generalizado, e $x_1, \dots, x_n$ são seus itens ancestrais. Sendo assim, durante a primeira varredura da base, utiliza-se Part1 para calcular os graus das transações em relação aos ancestrais da ontologia. Os valores retornados são armazenados na segunda estrutura, ilustrada na Figura 6.15.

Na Figura 6.15, uma chave é um ancestral $x$, e cada chave aponta para um valor, que é um vetor armazenando o grau que cada transação da base de dados suporta a chave $x$. Note que $x$ poderá estar presente nas regras generalizadas que serão geradas. Se $\mu_{tx} = 0$, então a transação não suporta $x_n$, portanto, o grau $\mu_{tx}$ não é armazenado no vetor da chave $x_n$.

Sendo assim, uma vez que os valores relativos aos ancestrais encontram-se disponíveis, durante o pós-processamento simplesmente utiliza-se a Part2 para efetuar o operador *min* em relação aos graus associados à regra $\{x_1, \dots, x_n\}$. Por exemplo, se os graus que uma transação T1 suporta $x_1$ e $x_2$ são 0.8 e 0.6, respectivamente, então o grau que T1 suporta a regra $\{x_1, x_2\}$ é *min*(0.8, 0.6) = 0.6. Note que *min*(0.8, 0.6) é a efetuação de Part2.

No entanto, para que o suporte estendido possa ser calculado, além dos graus relativos aos ancestrais da regra, é preciso que se saiba também em quais transações ela ocorre, pois como mencionado, essa medida é obtida com base em uma somatória. Sendo assim, o cálculo só é possível porque existe uma relação entre as duas estruturas geradas pelo algoritmo, ilustradas na Figura 6.14 e na Figura 6.15. Cada vetor ligado a uma chave da Figura 6.14 tem a mesma quantidade de posições do vetor ligado à mesma chave da Figura 6.15. Além disso, os valores de posições correspondentes dos vetores estão relacionados. Por exemplo, através da Figura 6.14 verifica-se que Fruta está presente em três transações, T1, T2 e T4.

Por conseguinte, T1, T2 e T4 estão respectivamente nas posições 1, 2 e 3 do vetor apontado por Fruta. Então, com base nessas posições, é possível inferir que os graus que T1, T2 e T4 suportam Fruta são 1, 0.7 e 0.7, respectivamente, pois esses graus estão nas mesmas posições (1, 2, 3) do vetor da chave Fruta na Figura 6.15.

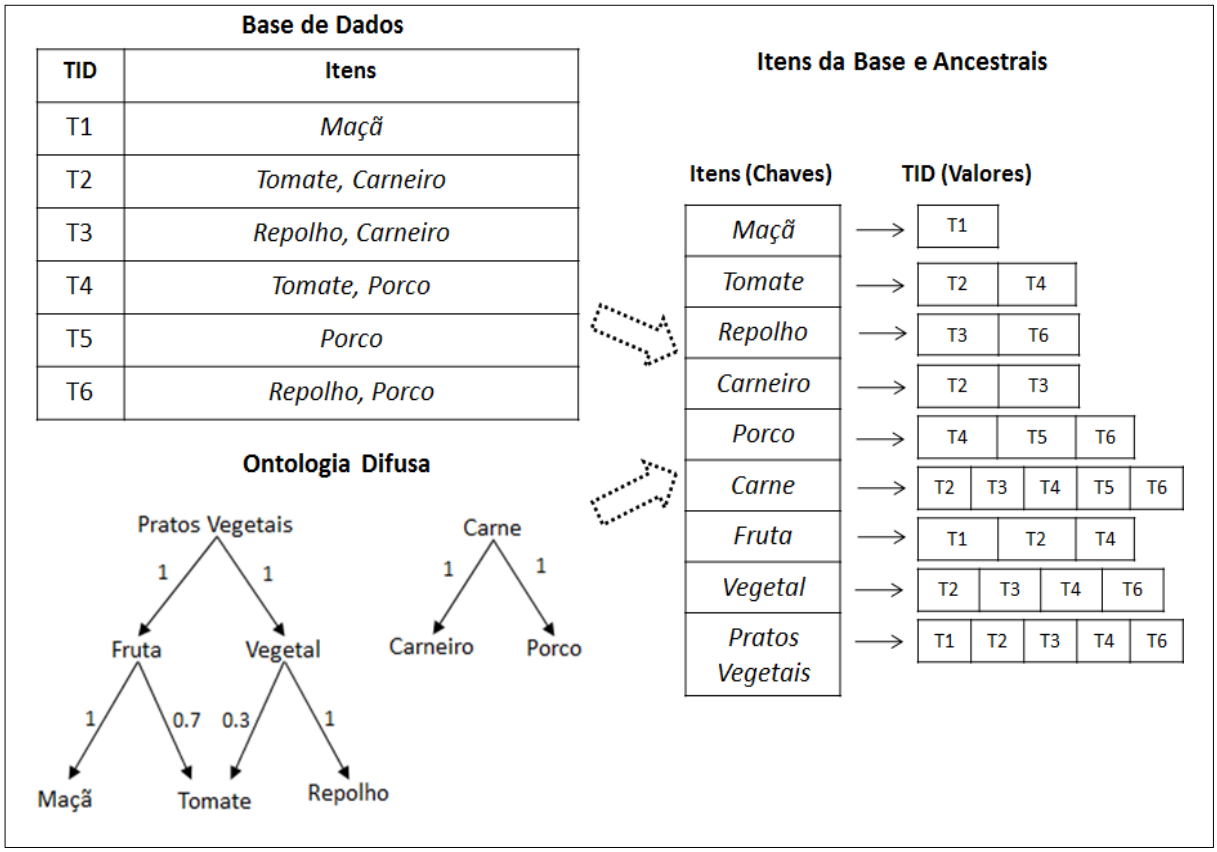


**Figura 6.14 – Estrutura utilizada para armazenar itens do banco e ancestrais da ontologia.**

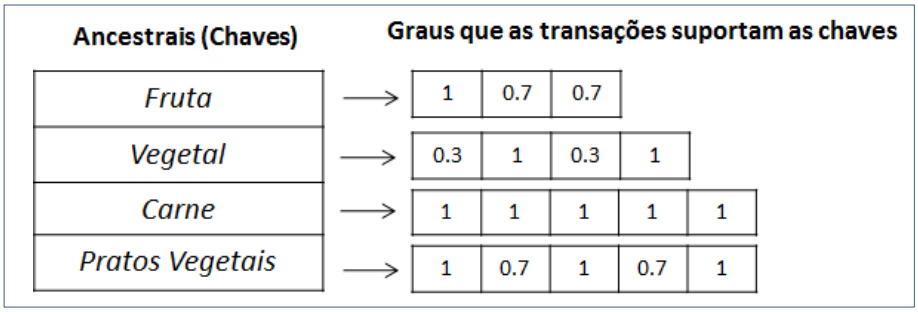


**Figura 6.15 – Estrutura utilizada para mapear as chaves (ancestrais) aos graus de suporte das transações.**

Para facilitar o entendimento, considere um exemplo prático de como calcular o suporte estendido da regra generalizada “Fruta → Carne”. Primeiro, o algoritmo utiliza a estrutura ilustrada na Figura 6.14 e identifica a intersecção dos valores armazenados nos vetores apontados pelas chaves “Fruta” e “Carne”. Essa intersecção representa todas as ocorrências simultâneas de “Fruta” e “Carne” nas transações da base de dados. Nesse caso, existem duas ocorrências. A Figura 6.16 ilustra a ideia.

Ainda em relação à estrutura da Figura 6.14, as transações referentes à intersecção possuem posições específicas nos vetores das chaves. Por exemplo, a

transação T2 está na segunda posição do vetor apontado por “Fruta”, e na primeira posição do vetor apontado por “Carne”. Então, o algoritmo utiliza as posições das transações T2 e T4, na Figura 6.12, para encontrar, na Figura 6.15, o grau que cada transação suporta “Fruta” e “Carne”. Portanto, teríamos: Fruta 0.7/T2, 0.7/T4; Carne: 1/T2, 1/T4. Com base nisso, o algoritmo utiliza Part2 para calcular o $\mu_{tX}$.

Para T2:

$$\mu_{tX} = \min_{\forall\, x \in X} (\, Part\ 1) = \min(0.7, 1) = 0.7$$

Para T4:

$$\mu_{tX} = \min_{\forall\, x \in X} (\, Part\ 1) = \min(0.7, 1) = 0.7$$

Sendo assim, como o suporte é calculado pela somatória do $\mu_{tX}$ de todas as transações, dividido pelo total de transações, teríamos (0.7 + 0.7) / 6 ≈ 0.23. é importante salientar que, embora tenha sido apresentado um exemplo específico, esse procedimento aplica-se a qualquer regra. Além disso, uma vez encontrado o suporte, a confiança pode ser facilmente obtida, já que deriva do mesmo.

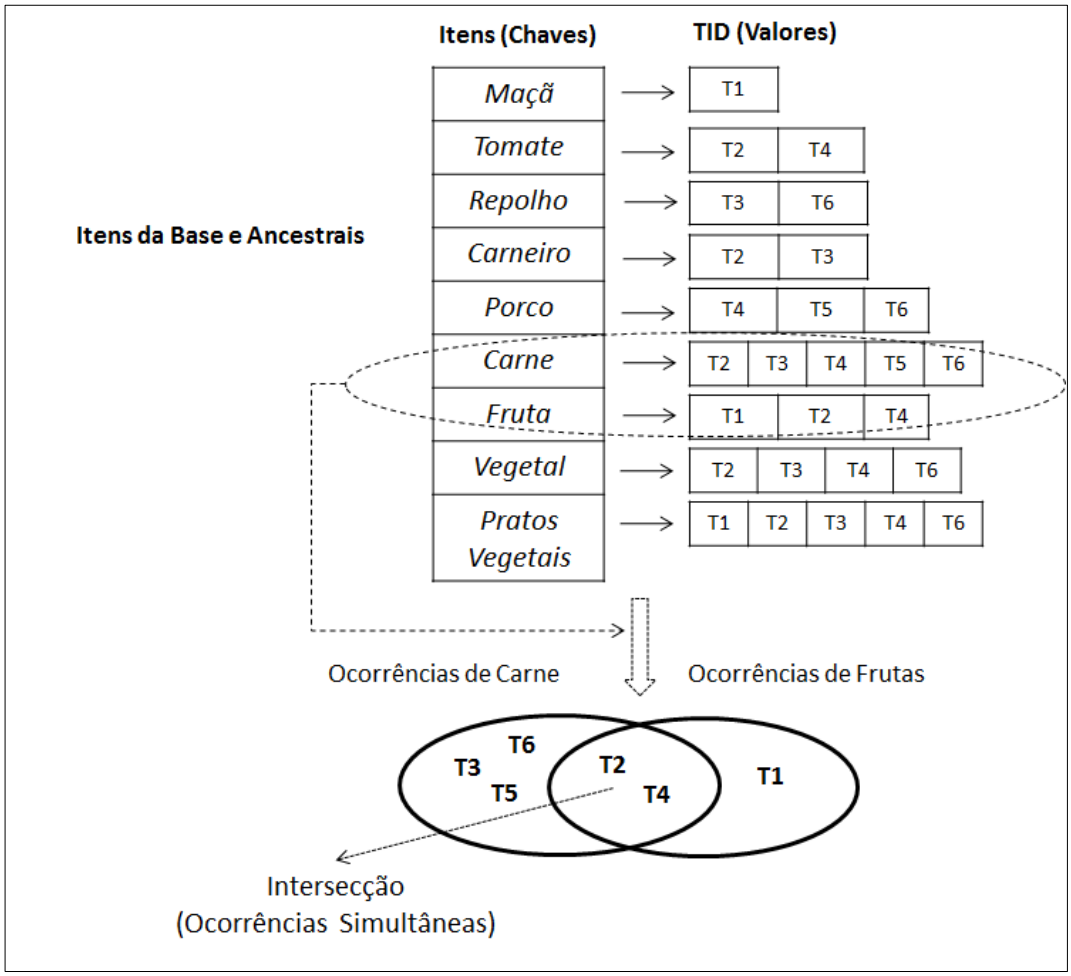


**Figura 6.16 – Representação da ideia utilizada no cálculo de suporte estendido.**

### 6.2.5 Utilizando Similaridade Baseada em Contexto nas Regras

Além dos graus de especialização/generalização, o algoritmo FOntGAR utiliza ontologias baseadas na metaontologia proposta por Cerri, *et al.* (2010), que permite a representação de relações de similaridade baseada em contexto. Essas relações são representadas nas folhas da ontologia, com o objetivo de enriquecer a semântica das regras.

Nessa etapa, o algoritmo utiliza o reasoner da estrutura para inferir similaridade semântica entre as folhas, de acordo com o *contexto* definido pelo usuário. Se o valor do grau de relacionamento entre dois ou mais indivíduos é maior ou igual ao valor do parâmetro de similaridade mínima *minsin,* uma associação de similaridade é detectada, e os elementos envolvidos são considerados suficientemente similares.

Uma questão importante é que toda associação de similaridade possui um tamanho, que é definido pela quantidade de indivíduos envolvidos nas mesmas. Dessa forma, de acordo com Escovar (2004), o tamanho é 2 quando existem dois elementos envolvidos, 3 quando são três elementos, e assim sucessivamente. A representação de uma associação de similaridade é feita utilizando o símbolo "~" entre os indivíduos envolvidos, por exemplo, $item_a \sim item_b$. Associações de similaridade de tamanho 2 são chamadas de binárias, e as que possuem tamanho maior ou igual a 3 são denominadas ciclos de similaridade.

Dessa forma, pode-se dizer que na etapa de identificação de similaridades o algoritmo procura por ciclos, porém, como nem sempre eles existem, associações binárias também podem ser identificadas. Por exemplo, considere que as ligações entre Frango/Peru, e Peru/Salsicha da Figura 6.17 representem a relação de similaridade entre os mesmos, e que os valores especificados sejam referentes ao grau da relação. Nesse caso, o algoritmo certamente detectaria o ciclo "frango~peru~salsicha", mas apenas se os valores dos graus fossem superiores ao *minsim* a associação seria estabelecida. Além disso, se a relação Peru/Salsicha não existisse, apenas a associação binária "frango~peru" seria identificada. Por conseguinte, todas as associações encontradas são armazenadas.

Um diferencial do trabalho é a obtenção de ciclos de similaridade por inferência, com base na propriedade transitiva da relação explorada. Nos trabalhos anteriores, o ciclo "frango~peru~salsicha" só seria encontrado se a relação entre

frango e salsicha fosse especificada na ontologia. Essa característica aumenta a complexidade da estrutura, pois demanda implementação elevada de relações.

Como se sabe, o algoritmo pode gerar um resultado composto por regras generalizadas e padrões que não puderam ser generalizados. Dessa forma, antes de apresentar o resultado ao usuário, ele verifica (em meio às regras não generalizadas) se existem padrões em que dois ou mais itens, no antecedente ou no consequente, façam parte de alguma associação encontrada. Caso isso ocorra, a associação é inserida na regra. A Figura 6.18 apresenta o pseudocódigo da etapa de similaridade.

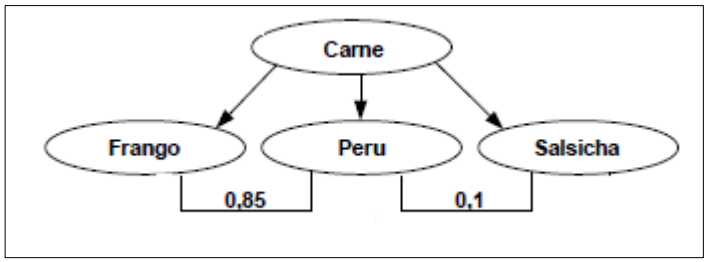


**Figura 6.17 – Exemplo Simples de Folhas com Similaridade.**

Considere, por exemplo, o padrão $frango, peru, alface \rightarrow pão, leite$, e o ciclo "frango~peru~salsicha". Ao analisar essa regra, o algoritmo verifica que os itens "frango" e "peru" fazem parte do ciclo, portanto, ambos são similares. Nesse caso, ele insere a associação "frango~peru" na regra, substituindo os itens "avulsos" pela mesma, de modo que o formato resultante do padrão é alterado para:

$$frango{\sim}peru,\ alface \rightarrow pão, leite.$$

Sendo assim, ao analisar a regra resultante a seguinte interpretação poderia ser feita: "sempre que frango, peru e alface são comprados, os itens pão e leite também são e, além disso, frango e peru são similares em relação ao sabor". Adote o termo "sabor" como sendo o contexto definido previamente.

Além da questão da realização de inferências, outro ponto que se diferencia dos trabalhos anteriores é que, nos mesmos, a representação de associações de similaridade nas regras é feita por itens difusos, que são inseridos no conjunto de itemsets candidatos, passando por um cálculo de suporte difuso. Esse cálculo é realizado com base no que se chama de ocorrências nebulosas.

A utilização de candidatos fuzzy é interessante por um lado, entretanto, possui o inconveniente de gerar uma quantidade de itemsets candidatos superior a que seria obtida por uma abordagem tradicional. Além disso, a questão de ocorrências nebulosas também pode ocasionar problemas semânticos nas regras.

```
1  Identifica associações de similaridade;
2  armazena associações de similaridade;
3  para cada regra não generalizadas faça
4    itensAntecedentes:= itens antecedentes da regra;
5    itensConsequentes:= itens consequentes da regra;
6    para cada associação de similaridade armazenada faça
7       se itensAntecedentes possui itens do ciclo então
8          Insere a associação correspondente na regra;
9          Remove esses itens de itensAntecedentes;
10      fim se
10      se intensConsequentes possui itens do ciclo
11           Insere associação correspondente na regra;
12           Remove esses itens de itensConsequentes;
13      fim se
14      Se itensAntecedentes & itensConsequentes estão vazios
15         Break;
16    fim para;
17 fim para;
```

**Figura 6.18 – Pseudocódigo utilizado na Etapa de Identificação e representação de Similaridade.**

## 6.3 Considerações Finais

Com base na revisão bibliográfica da literatura, nos capítulos apresentados no decorrer desta dissertação procurou-se descrever de forma sucinta os principais assuntos relacionados ao trabalho realizado.

Sendo assim, o Capítulo 6 introduz o detalhamento do algoritmo FOntGAR (*Fuzzy Ontology-based Generalized Association Rules Algorithm*), um novo algoritmo para mineração de regras generalizadas, utilizando ontologias difusas, em etapa de pós-processamento. Além da etapa explorada, pode-se destacar a utilização de ontologias com graus de especialização/generalização variando em [0,1] e relação de similaridade baseada em contexto.

A questão da relação entre base de dados e ontologia também é um ponto importante, principalmente porque a existência de relação parcial faz com que a abordagem de agrupamento e o uso do parâmetro mingen também sejam adaptados a essa possibilidade. É importante ressaltar que, no algoritmo proposto, a aplicação de relações de similaridade nas regras é feita de modo totalmente diferente do que foi apresentado até o momento na literatura.

# Capítulo 7

# EXPERIMENTOS

## 7.1 Considerações Iniciais

Conforme apresentado no Capítulo 6, o algoritmo FOntGAR realiza a mineração de regras de associação generalizadas, em etapa de pós-processamento, utilizando ontologias de conceito fuzzy, e relações de similaridade baseadas em contexto, objetivando obter um conjunto final de regras mais reduzido, compreensível e semântico

A abordagem de generalização que foi utilizada permite que as regras geradas não possuam o mesmo significado entre elas, eliminando a necessidade de medidas de poda. Dentre outras coisas, no algoritmo FOntGAR não existe a restrição de que a relação entre base de dados e estrutura taxonômica seja total. Devido a isso, tanto o agrupamento por similaridade quanto à utilização do parâmetro *mingen* tiveram seu uso estendido para permitir a relação parcial. É importante ressaltar também a obtenção de associações de similaridade por inferência, permitindo a exploração do axioma de transitividade.

A implementação do algoritmo foi realizada com base na tecnologia Java, e no framework Jena para desenvolvimento de aplicações baseadas em ontologias. As inferências sobre a ontologia difusa são efetuadas pelo reasoner de Jena, que oferece suporte a ontologias em OWL DL. Além disso, o framework Jena foi a base para a implementação dos mecanismos de inferência sobre conceitos e relacionamentos difusos contidos na ontologia. Sendo assim, a seção seguinte apresenta os experimentos realizados e os resultados obtidos.

## 7.2 Experimentos Realizados

Para realização dos experimentos foram utilizados dados reais do Censo Demográfico de 2000, fornecidos pelo IBGE (Instituto Brasileiro de Geografia e Estatística), e um conjunto de dados reais de um supermercado localizado na cidade de São Carlos, fornecido pelo Laboratório de Inteligência Computacional (LABIC) da USP São Carlos. Em relação ao IBGE, foi analisado um conjunto de dados com 10.000 transações e 12 itens distintos, contendo informações sobre sexo, anos de estudo, e etnia. O segundo conjunto de dados possui informações a respeito de compras efetuadas no supermercado, e possui 1716 transações com 200 itens distintos. Duas ontologias difusas foram modeladas, uma para trabalhar com o conjunto de dados do IBGE, chamada de FOnt-1 e outra considerando os dados do supermercado, chamada FOnt-2.

A ontologia FOnt-1 é uma extensão da ontologia de características demográficas utilizada por Miani (2009), onde foram incluídos graus de especialização/generalização variando em [0,1], e representação de contexto. Como a ontologia FOnt-2 especifica um domínio de itens de supermercado, ela foi modelada com base na estrutura taxonômica de itens utilizada na rede de supermercados Pão de Açúcar, identificada através do site da empresa. Assim como em Miani (2009), além das informações sobre sexo, anos de estudo e raça, a ontologia FOnt-1 também inclui outros tipos de características demográficas, tais como grupos de idade, local de moradia e estado civil.

A Figura 7.2 representa uma parte da ontologia difusa FOnt-1, ilustrada na Figura 7.1. Essa divisão foi feita para uma melhor compreensão de como o conjunto de dados explorado é relacionado com FOnt-1. Na Figura 7.2 tem-se a representação completa da ontologia da Figura 7.1 para *Anos de Estudo* (composta pelos itens *Não_determinado, Menos_de_três_anos, Quatro_a_dez_anos e Onze_ou_mais), Sexo* (itens *Masculino e Feminino*) e *Raça ou etnia* (itens *Sem_declaração, Negro, Mulato, Asiático, Indigena e Branco*).

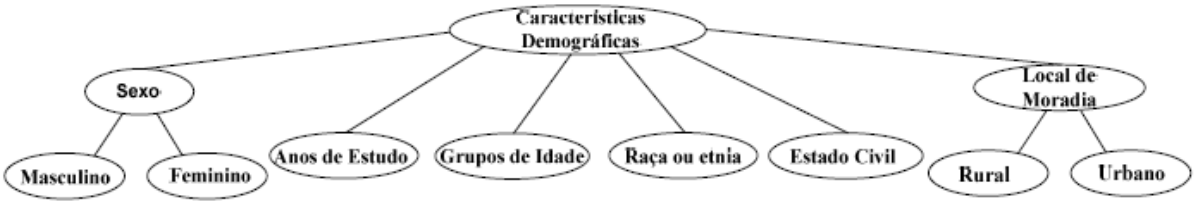


**Figura 7.1 – Taxonomia Crisp de Características Demográficas.**

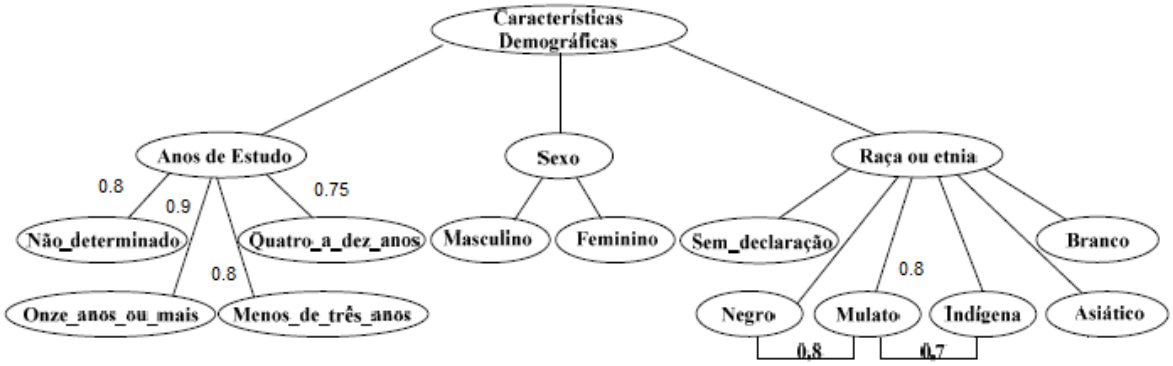


**Figura 7.2 – Representação para Anos de Estudo, Sexo e Etnia.**

Note que *Anos de Estudo, Sexo e Raça ou etnia* correspondem aos ancestrais, e as folhas da Figura 7.2 são os itens que compõem o conjunto de dados analisados. Portanto, conforme a ontologia utilizada em (Miani, 2009), todo item da base possui um representante nas folhas da estrutura. Os valores no relacionamento entre as instâncias são os graus de similaridade entre elas.

Conforme ilustrado, a ontologia FOnt-1 possui apenas um nível de generalização, excluindo a raiz. Por outro lado a ontologia difusa de supermercado FOnt-2 foi modelada contendo 4 níveis de generalização, excluindo a raiz, portanto, a partir de FOnt-2 foi possível explorar a generalização em mais níveis da estrutura, validando o algoritmo proposto. Além disso, FOnt-2 inclui mais relações de similaridade, permitindo a exploração dos ciclos de similaridade. A média de valores dos graus de especialização/generalização inseridos em FOnt-2 foi 0.8, e além disso, foram explorados dois contextos diferentes, *sabor* e *aparência*. É importante salientar que nem todos os itens da base foram incluídos nas folhas da estrutura, e nem todos os itens da estrutura estão presentes na base, de forma a validar o tratamento de agrupamento e o cálculo do *mingen* estendido.

Com o objetivo de verificar o desempenho do algoritmo FOntGAR, os experimentos foram realizados em relação a dois aspectos principais, tempo de execução, e taxa de redução. Primeiro, utilizando o conjunto de dados do IBGE, FOntGAR e mais dois algoritmo, NARFO e GARPA (citados em Capítulos anteriores) foram executados, e comparados. FOntGAR utilizando a ontologia difusa FOnt-1, NARFO a ontologia crisp correspondente, e GARPA uma taxonomia crisp correspondente. O propósito foi avaliar quais efeitos, em termos de desempenho, a utilização de conceitos fuzzy pode produzir, uma vez que os dois outros algoritmos utilizados na comparação utilizam estruturas crisp. Nessa comparação, dois experimentos foram conduzidos.

Em relação à taxa de redução, o algoritmo FOntGAR foi executado utilizando a ontologia difusa FOnt-2 e o conjunto de dados de supermercado. O propósito do experimento foi verificar o quanto o tratamento de generalização melhora a redução do volume de regras. Por exemplo, considerando que a partir de um total de 3500 regras extraídas o algoritmo apresente 1800 regras, incluindo generalizadas e tradicionais, a taxa de redução nesse caso é 48,75% (((3500-1800) / 3500) * 100).

Além dos experimentos em termos de desempenho e redução, também foi verificada a questão da obtenção e inclusão de associações de similaridade nas regras. Nesses testes também foram utilizados os dados do supermercado e a ontologia difusa FOnt-2.

### *A. Comparação entre FOntGAR, NARFO e GARPA*

Conforme mencionado no Capítulo 6, dentre outros fatores, dois pontos principais podem afetar o tempo de execução total do algoritmo, o pós-processamento em si (principalmente agrupamento e cálculo de suporte), e o tempo gasto pelo algoritmo durante a verificação do grau que cada transação suporta um ancestral da estrutura, na etapa inicial. Portanto, o objetivo desses experimentos foi avaliar o impacto de se utilizar ontologias de conceitos fuzzy, em comparação a duas abordagens que também trabalham com pós-processamento, mas utilizam estruturas taxonômicas que não consideram graus de especialização/generalização variando em [0,1].

Sendo assim, alguns experimentos foram realizados com os dados reais e as estruturas taxonômicas mencionadas anteriormente, analisando um aspecto diferente em cada teste. Primeiramente, avaliou-se o tempo de varredura inicial das

transações da base de dados, em função da quantidade de transações varridas. Posteriormente, avaliou-se o tempo total de execução do algoritmo, em função da variação do *minsup*. A hipótese adotada é que ao variar o *minsup* a quantidade de regras obtidas na etapa de extração de padrões também se altera, afetando o pós-processamento como um todo, mas principalmente a etapa de agrupamento e cálculo de suporte das regras generalizadas. Exceto pelo parâmetro verificado, os testes foram realizados com valores fixos, onde, *minsup*=0.02, *minconf*=0.4 e *mingen*=0.2. O lado da generalização foi definido como *lr*.

***Número de transações:***

Na Figura 7.3, o eixo vertical é a média do tempo de varredura por transação, em relação à primeira varredura da base de dados. Nesse experimento não foi utilizado o algoritmo GARPA, pois o mesmo trabalha somente no pós-processamento, e não inclui a etapa inicial mencionada. Portanto, a comparação foi realizada apenas com algoritmo NARFO.

Para identificação da média, a cada 2.000 transações varridas verificou-se o tempo total de varredura. Nesse teste, o intervalo de transações variou de 2.000 a 10.000. Portanto, a partir da Figura 7.3, o gap entre FOntGAR e NARFO mostra que a varredura com ontologias difusas com graus de especialização/generalização realmente consome mais tempo do que a varredura utilizando ontologias crisp sem esses graus. O motivo é que durante a varredura inicial ocorrem, simultaneamente, duas outras atividades: a geração da estrutura responsável por transformar a base em formato vertical, onde é possível obter acesso indexado às ocorrências dos itens, e também a geração da estrutura que armazena os graus que cada transação suporta um ancestral da ontologia. Portanto, as atividades adicionais contribuem com o consumo de tempo.

Entretanto, embora o tempo gasto tenha sido superior, a Figura 7.3 mostra que o gap tende a se manter estável com o aumento do número de transações, e a diferença entre as duas curvas se torna constante a partir de certo ponto. Isso mostra que, em ambos os casos, a complexidade computacional é linear em relação ao número de transações.

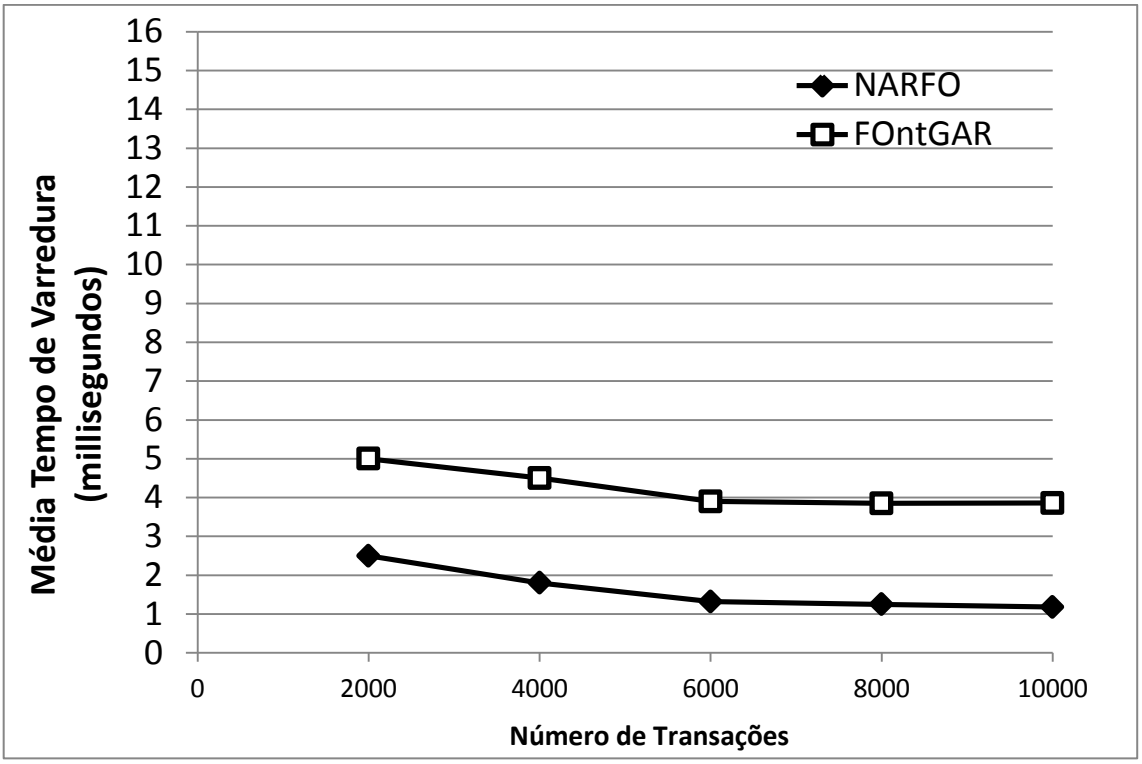


**Figura 7.3 – Análise do Tempo de Varredura por Transação gasto pelos Algoritmos.**

***Grau de suporte mínimo:***

Na Figura 7.4, o eixo vertical é o tempo de execução total do algoritmo, em função da variação do grau de suporte mínimo do eixo horizontal. O *minsup* foi variado de 0.05% a 0.2%, e a partir do gráfico é possível verificar que: 1) Como esperado, com o aumento do valor do *minsup,* o tempo de execução de FOntGAR e GARPA diminui; 2) quando o valor do *minsup* diminui, o tempo de execução de ambos os algoritmos aumenta. Em relação a esses resultados, é possível observar que à medida que o valor do minsup aumenta menos padrões tradicionais são extraídos, reduzindo o processamento necessário durante o agrupamento, e a quantidade de cálculos de suporte estendido das regras generalizadas. Por outro lado, quando o *minsup* decresce, um número maior de regras é extraído, ocorrendo o inverso do caso anterior.

Sendo assim, o experimento comprova a nossa hipótese de que quanto maior o número de regras, que é diretamente proporcional ao *minsup* ou *minconf*, mais tempo o algoritmo consome durante o processamento total, pois mais regras são pós-processadas, ou seja, mais grupos de regras necessitam ser gerados e mais cálculo de suporte necessita ser efetuado. É importante salientar que a etapa de

identificação de similaridades no FOntGAR foi anulada, pois GARPA não contempla essa funcionalidade.

Embora GARPA seja uma abordagem crisp, que não utiliza taxonomias com diferentes graus de especialização/generalização e, portanto, não necessita realizar calculo de suporte estendido, o tempo gasto por esse algoritmo é superior ao verificado em FOntGAR. Portanto, GARPA é mais custoso em seu pós-processamento, principalmente pelo cálculo de suporte das regras generalizadas, no qual são efetuadas novas varreduras para cada regra obtida. Além disso GARPA também realiza o tratamento de agrupamento baseando-se na similaridade das regras, no entanto, o algoritmo FOntGAR mostra-se mais eficiente, pois elimina etapas desnecessárias e, embora as estruturas de dados utilizadas consumam mais tempo durante a etapa inicial, elas evitam que novas varreduras sejam efetuadas na base durante o pós-processamento, permitindo acesso mais eficiente aos dados, e contribuindo para a diminuição do tempo total de execução.

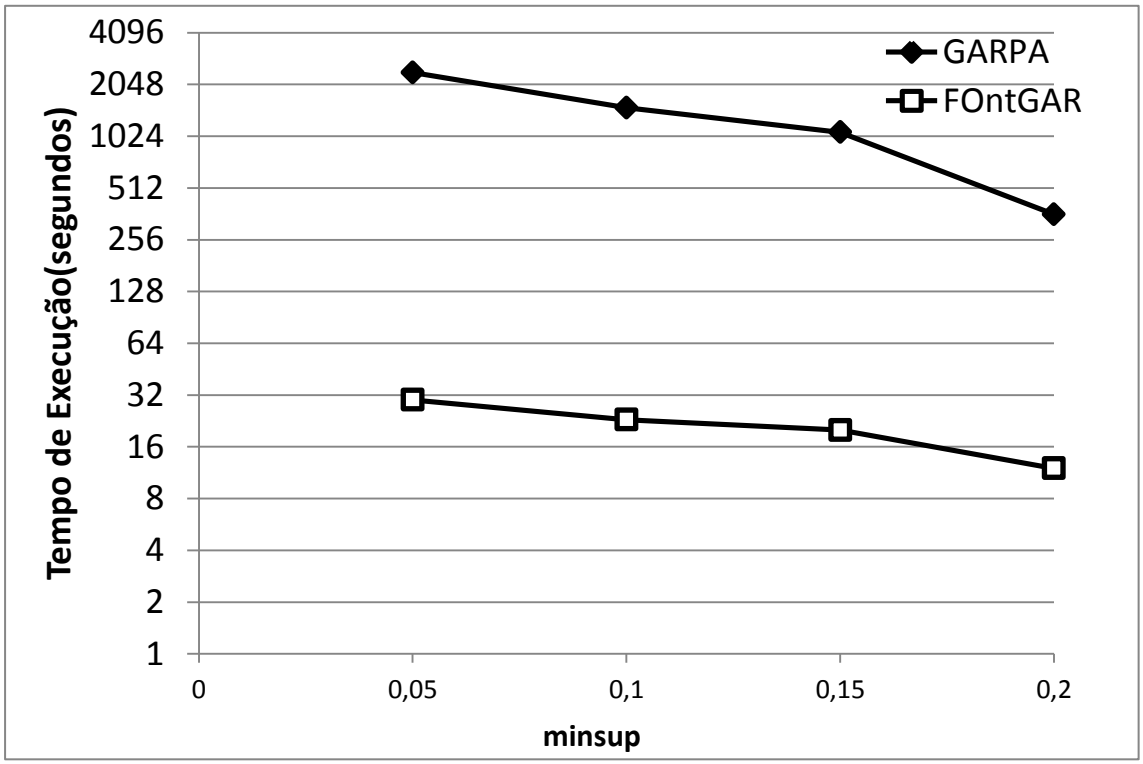


**Figura 7.4 – Tempo Total de Execução dos Algoritmos Variando o *minsup.***

### *B. Taxa de Redução do algoritmo FOntGAR*

Conforme destacado no Capítulo inicial, um dos objetivos do algoritmo proposto é a redução da quantidade de padrões gerados ao usuário. Sendo assim, a

Figura 7.5 apresenta a taxa de redução do algoritmo, que é o percentual de redução do conjunto de regras extraído em um conjunto de dados. O teste realizado, apresentado na Figura 7.5, mostra que a taxa de redução do algoritmo FOntGAR é significante, especialmente quando os valores do parâmetro *mingen* são baixos. Para valores altos do parâmetro de generalização mínima o número de regras generalizadas diminui, pois é necessário que existam mais descendentes dos ancestrais em cada grupo. Consequentemente, a quantidade de regras tradicionais no conjunto final gerado aumenta, refletindo no montante final.

Conforme descrito no Capítulo 6, o resultado apresentado ao usuário pode ser composto por regras generalizadas, separadas por níveis de generalização, e regras que não puderam ser generalizadas. Portanto, sabendo que a fórmula para cálculo da taxa de redução é (((total de regras extraídas – total de regras apresentadas ao usuário) / total de regras extraídas) * 100), adote "total de regras apresentadas ao usuário" como a soma das regras que não puderam ser generalizadas com as regras generalizadas em cada nível. Com base nisso, de acordo com a Figura 7.5, quando o *mingen* varia entre 0.2 e 0.4 (melhores casos) a taxa de redução é aproximadamente 65%. Se considerarmos o *mingen* valendo 0.6 a taxa de redução ainda é expressiva, sendo aproximadamente 40%.

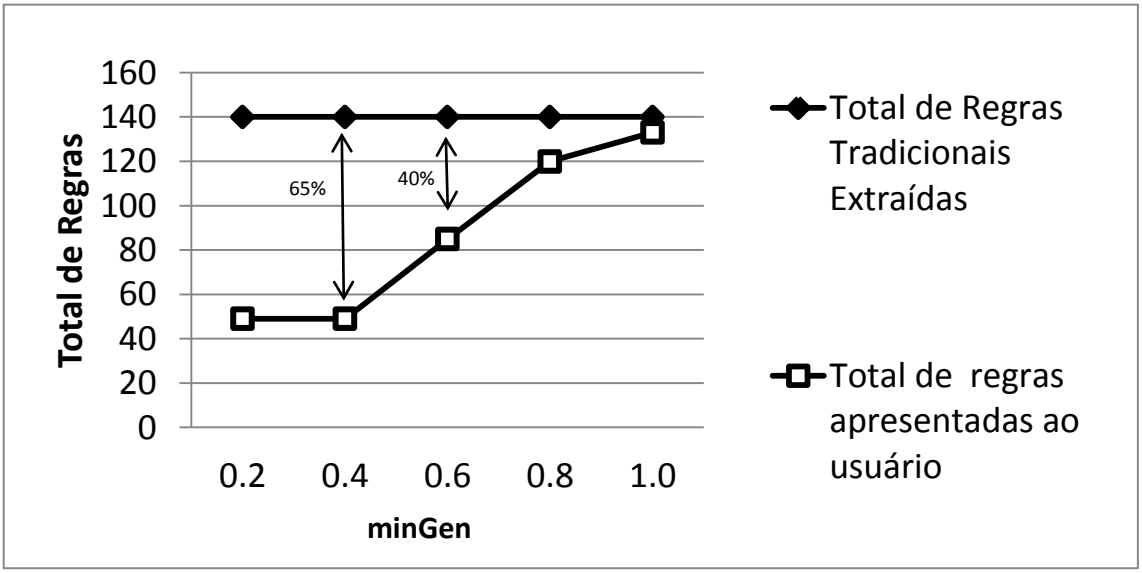


**Figura 7.5 – Redução de Regras (%) no Algoritmo FOntGAR usando a Segunda Base.**

Em relação ao teste anterior, optou-se em variar o *mingen*, pois assim a quantidade de regras geradas na etapa de extração de padrões se manteria constante, facilitando a verificação da redução. Assim, é possível concluir que o

aumento de regras apresentadas é decorrente da elevação da quantidade de padrões não generalizados. Para uma melhor compreensão dos efeitos da variação do *mingen* no resultado, considere a Figura 7.6 (relativa a segunda base de dados).

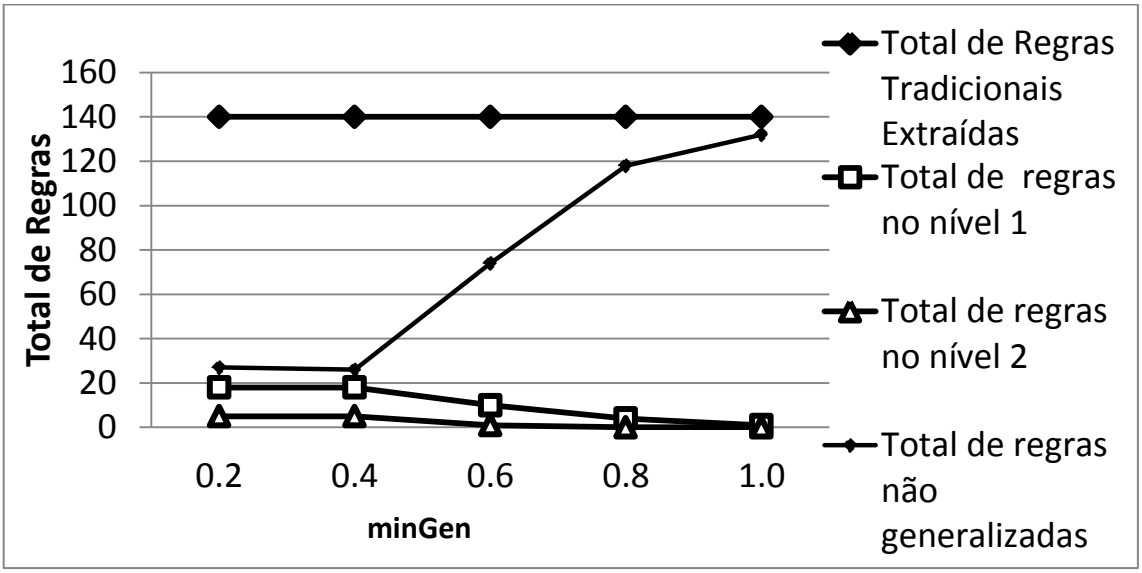


**Figura 7.6 – Variação da Quantidade de Regras de Acordo com o Parâmetro *minGen.***

### *C. Exploração de relações de similaridade*

Outra funcionalidade do algoritmo proposto é a exploração de relações de similaridade através de inferências. Conforme mencionado anteriormente, as relações de similaridade são representadas nas folhas da ontologia, podendo ser utilizadas nas regras. Sendo assim, para que uma regra contenha relação de similaridade, é necessário que a mesma possua itens que pertençam a algum ciclo encontrado. Nesse sentido, dois contextos diferentes foram inseridos na ontologia difusa FOnt-2, aparência e sabor. As tabelas 7.1 e 7.2 apresentam algumas regras obtidas de acordo com o contexto explorado. Note que a regra 4 da Tabela 7.1 possui um clico de similaridade de tamanho três, "leite_batavo~leite_parmalat~leite_nilza", mas de acordo com a Figura 7.7, apenas as relações entre "leite_batavo"/"leite_parmalat" e "leite_parmalat"/"leite_nilza" foram especificadas. Nesse caso, o cliclo "leite_batavo~ leite_parmalat~leite_nilza" foi obtido por meio de inferência, explorando o axioma de transitividade.

**Tabela 7.1 – Regras Tradicionais Incluindo Associações de Similaridade por Sabor**

| Regras com similaridade (contexto sabor) | |
|---|---|
| 1 | CAFE_SERRA_DA_GRAMA~CAFÉ_RIBERAO_BONITO~CAFÉ_ITAMARATY -> SPO_OMO, COCA_COLA suporte = 0.0058275056 confiança = 0.10309278. |
| 2 | BATATA_RUFFLES~BATATA_PALHA_ELMA_CHIPS -> ACUCAR_UNIAO suporte |

| | |
|---|---|
| | = 0.006993007 confiança = 0.2352941 |
| 3 | ACUCAR_UNIAO~ ACUCAR_DA_BARRA -> DETERGYPE suporte = 0.01398601 |
| 4 | LEITE_BATAVO~ LEITE_PARMALAT~LEITE_NILZA,ACUCAR_UNIAO -> PAPEL_HIGPERSONAL suporte = 0.0099067595 confiança = 0.2881356 |
| 5 | IOGNESTLE~IOGCORPUS~IOGBATAVO -> BISCNESTLE suporte = 0.012820513 confiança = 0.3548387 |
| 6 | ACHOCNESCAU~ACHOCTODDY->LEITE_BATAVO~LEITE_PARMALAT,BISCMARILAN suporte = 0.005244755 confiança = 0.0927835 |
| 7 | LEITE_MOCA,ACHOCNESCAU -> BISCMARILAN~ BISCNESTLE suporte = 0.005244755 confiança = 0.17999999 |
| 8 | ACHOCNESCAU -> SPO_MINERVA~SPO_OMO,BISCNESTLE suporte = 0.006993007 confiança = 0.12371134 |
| 9 | VINAGRE_CASTELO~VINAGRE_UNIÃO,AMACCOMFORT -> BISCNESTLE suporte = 0.005244755 confiança = 0.8181818 |
| 10 | TANG~SUCO_DEL_VALE -> PAPEL_HIGPERSONAL suporte = 0.006993007 confiança = 0.3076923 |

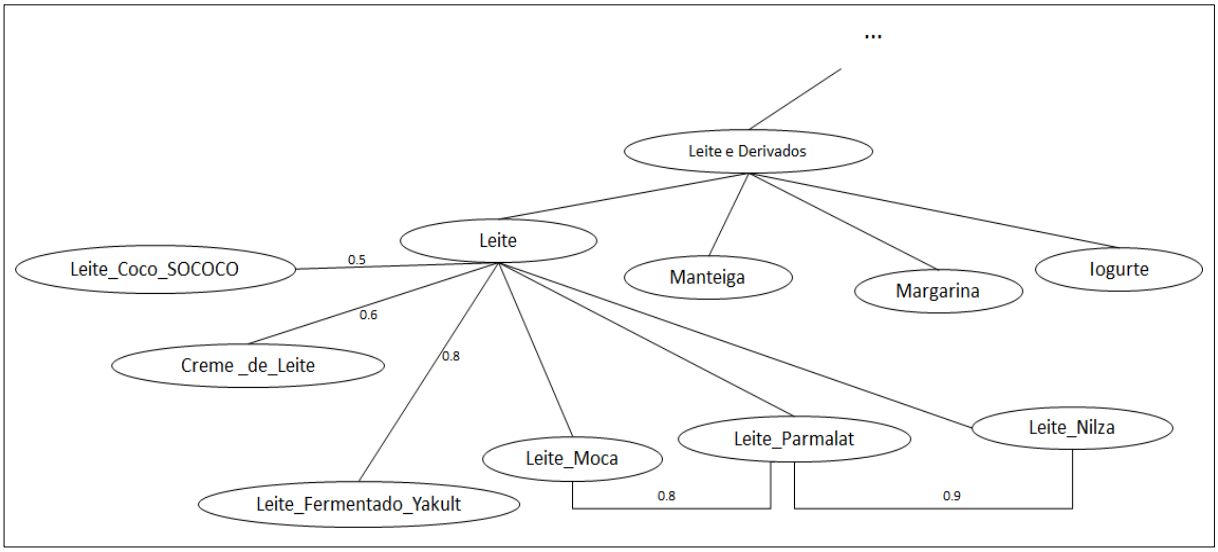


**Figura 7.7 – Parte da ontologia difusa FOnt-2.**

**Tabela 7.2 - Regras Tradicionais Incluindo Associações de Similaridade por Aparência**

| Regras com similaridade (contexto aparência do produto) | |
|---|---|
| 1 | ABSCAREFREE~ABSSLIVRE -> LEITE_PARMALAT suporte = 0.0064102565 confiança = 0.31428573 |
| 2 | ACUCAR_UNIAO -> VINAGRE_CASTELO,DETERGLIMPOL~DETERGYPE suporte = 0.0078102565 confiança = 0.31428573 |
| 3 | SPO_OMO,ACUCAR_UNIAO -> PAPEL_HIGNEVE~PAPEL_HIGPERSONAL suporte = 0.007575758 confiança = 0.35135135 |
| 4 | TOALHA_PAPEL_MASCOT~ TOALHA_PAPEL_SNOB -> CALDO_KNORR suporte = 0.0058275056 confiança = 0.19607842 |
| 5 | MACRENATA ->TOALHA_PAPEL_MASCOT,ESPONJA_BOMBRIL~ESPONJA_ASSOLAN suporte = 0.0034965035 confiança = 0.06185567 |
| 6 | EXTRATO_ELEFANTE_KNORRCICA~ MOLHO_TOMATE_CIRIO -> TOALHA_PAPEL_MASCOT suporte = 0.004079254 confiança = 0.175 |
| 7 | TOALHA_PAPEL_MASCOT -> FEIJAO_GRAO_DO_CAMPO~ FEIJAO_BROTO_LEGAL suporte = 0.0046620048 confiança = 0.15686275 |

A seguir serão apresentadas outras tabelas contendo alguns exemplos de regras tradicionais e generalizadas obtidas pelo tratamento de generalização do algoritmo. Note que algumas regras possuem o termo “exceto por” que indica que a generalização não inclui os itens da exceção. Essas exceções ocorrem devido ao uso do parâmetro *mingen*, que permite que a generalização ocorra mesmo se um grupo não possui todos os descendentes do ancestral em questão. Embora tenham sido ilustradas apenas generalizações nos níveis 1 e 2, o algoritmo permite a generalização em qualquer nível da ontologia.

**Tabela 7.3 – Exemplo de Regras Tradicionais Geradas**

| Regras Tradicionais | |
|---|---|
| 1 | ABSSLIVRE -> FARTRIGO_RENATA,ESPONJA_BOMBRIL suporte = 0.005244755 confiança = 0.13432835 |
| 2 | FARTRIGO_RENATA,ESPONJA_BOMBRIL -> ABSSLIVRE suporte = 0.005244755 confiança = 0.5625 |
| 3 | BATATA_RUFFLES -> FARTRIGO_RENATA suporte = 0.0064102565 confiança = 0.21568628 |
| 4 | CALDO_MAGGI -> ACUCAR_UNIAO suporte = 0.008158508 confiança = 0.24561404 |
| 5 | CALDO_MAGGI -> ACUCAR_DA_BARRA suporte = 0.008741259 confiança = 0.2631579 |
| 6 | SUCO_DEL_VALLE -> BATATA_RUFFLES suporte = 0.0034965035 confiança = 0.10344828 |
| 7 | PAO_WICKBOLD_FORMA -> SUCO_DEL_VALLE suporte = 0.0034965035 confiança = 0.072289154 |
| 8 | IOGCORPUS -> SUCO_DEL_VALLE suporte = 0.0046620048 confiança = 0.15686275 |
| 9 | PAO_WICKBOLD_LIGHT -> SUCO_DEL_VALLE suporte = 0.0046620048 confiança = 0.11594204 |
| 10 | BISCPIRAQUE -> PAPEL_HIGPERSONAL,FARTRIGO_RENATA suporte = 0.004079254 confiança = 0.09333333 |
| 11 | PAPEL_HIGPERSONAL,FARTRIGO_RENATA -> BISCPIRAQUE suporte = 0.004079254 confiança = 0.21212119 |
| 12 | TANG -> ACHOCNESCAU suporte = 0.0058275056 confiança = 0.25641024 |
| 13 | CDSORRISO -> TANG suporte = 0.0064102565 confiança = 0.1641791 |
| 14 | CALDO_MAGGI -> TOALHA_PAPEL_MASCOT,ESPONJA_BOMBRIL suporte = 0.0034965035 confiança = 0.10526316 |

**Tabela 7.4 – Exemplos de Regras Generalizadas no Nível 1**

| Regras Generalizadas (Nível 1) | |
|---|---|
| 1 | Feijao -> Sabonete suporte = 0.039627038 confiança = 0.2029851 regra generalizada, exceto por: FEIJAO_GRAO_DO_CAMPO,FEIJAO_BROTO_LEGAL,FEIJAO_TORRESAN |
| 2 | Molho_Tomate -> Achocolatado suporte = 0.026223775 confiança = 0.20642202 regra generalizada, exceto por: MOLHO_TOMATE_CIRIO |
| 3 | Molho_Tomate -> Creme_Dental suporte = 0.016317017 confiança = 0.1573034 regra generalizada, exceto por: MOLHO_TOMATE_CIRIO,EXTRATO_ELEFANTE_KNORRCICA |

| | |
|---|---|
| 4 | Refrigerante -> Molho_Tomate suporte = 0.064685315 confiança = 0.059011165 regra generalizada, exceto por: COCA COLA,FANTA |
| 5 | Detergente -> Molho_Tomate suporte = 0.027972028 confiança = 0.12182741 regra generalizada, exceto por: DETERGLIMPOL |
| 6 | Acucar -> Achocolatado suporte = 0.08682984 confiança = 0.13232683 regra generalizada, exceto por: ACUCAR_CRISTALCUCAR |
| 7 | Absorvente -> Acucar suporte = 0.028554779 confiança = 0.24019608 regra generalizada, exceto por: ABSSLIVRE,ABSCAREFREE,ACUCAR_CRISTALCUCAR |
| 8 | Acucar -> Achocolatado suporte = 0.08682984 confiança = 0.13232683 |

**Tabela 7.5 – Exemplos de Regras Generalizadas no Nível 2**

| Regras Generalizadas (Nível 2) | |
|---|---|
| 1 | Chocolate_e_Derivados -> Molho_e_Condimento suporte = 0.045454547 confiança = Infinity regra generalizada, exceto por: Caldos,Catchup |
| 2 | Molho_e_Condimento -> Chocolate_e_Derivados suporte = 0.045454547 confiança = Infinity regra generalizada, exceto por: Molho_Tomate,Caldos,Catchup |

## 7.3 Considerações Finais

Esse capítulo apresentou experimentos realizados no algoritmo FOntGAR, utilizando conjuntos de dados reais, com o objetivo de validar a abordagem proposta. Em relação aos graus de especialização/generalização, os mesmos foram inseridos manualmente, de forma coerente, mas apenas com o objetivo de validar a abordagem proposta, e não para representar um domínio difuso real. A respeito disso, é importante ressaltar que o objetivo do trabalho não é apresentar uma abordagem de fuzzificação de ontologias, e sim explorar estruturas que incluam esses graus. Uma vez que o algoritmo foi validado com os valores inseridos, qualquer estrutura pode ser utilizada.

# Capítulo 8

# CONCLUSÕES

## 8.1 Resultados Obtidos

Conforme apresentado no Capítulo 1, o principal objetivo deste trabalho foi desenvolver um algoritmo para mineração de regras de associação generalizadas que proporcionasse um conjunto final de regras mais reduzido, não redundante e semântico, facilitando a compreensão do usuário. Entretanto, é importante enfatizar que, embora apresente ganhos em relação à redução de regras, a generalização não é uma obrigatoriedade, pois muitas vezes padrões específicos também podem ser importantes, dependendo do domínio e do contexto de aplicação do algoritmo.

Sendo assim, considerando o conteúdo apresentado, e as análises dos experimentos do capítulo anterior, pode-se dizer que o trabalho atingiu seu objetivo, pois os testes mostraram uma importante taxa de redução no montante de regras gerado. Além disso, o pós-processamento foi fundamental para obtenção de regras sem redundância, com o diferencial de não ter sido necessário utilizar medidas de poda, e a utilização de similaridade nas regras enriqueceu o conteúdo semântico das mesmas. Em relação à similaridade, a exploração do axioma de transitividade permitiu a realização de inferências para identificação de ciclos de similaridade.

Ademais, os experimentos mostram que, embora a varredura inicial (candidatos) consuma mais tempo do que em uma abordagem crisp, o gap entre as curvas tende a se manter constante com o aumento do número de transações. Outro resultado interessante é a introdução de um método para cálculo de suporte estendido no pós-processamento que evita varreduras adicionais na base de dados.

Portanto, considerando todo o conteúdo, espera-se que o trabalho desenvolvido seja uma contribuição relevante à literatura.

## 8.2 Contribuições

Estão entre as contribuições deste trabalho:

- Um novo algoritmo (FOntGAR) que realiza a mineração de regras de associação generalizadas não redundantes em todos os níveis de ontologias fuzzy, incluindo relações de similaridade baseadas em contexto.
- Considerando a utilização de estruturas taxonômicas com graus de especialização/generalização variando em [0,1], este trabalho introduz na literatura a generalização em etapa de pós-processamento.
- A especificação de uma nova abordagem de agrupamento de regras baseada na similaridade entre as mesmas, que inclui a possibilidade de relação parcial entre itens da base e ontologia.
- A introdução de um método para cálculo de suporte e confiança estendidos em etapa de pós-processamento que dispensa a necessidade de varreduras adicionais na base de dados e contribui com o tempo de execução do algoritmo.
- Uma nova abordagem de utilização de relações de similaridade nas regras, que dispensa a utilização de itens difusos, e não aumenta a quantidade de itemsets candidatos.
- A definição de um parâmetro *mingen* estendido.
- Uma abordagem que dispensa o uso de medidas utilizadas para podar regras redundantes.

### 8.2.1 Publicações

As contribuições apresentadas anteriormente proporcionaram a oportunidade de publicação dos resultados em eventos internacionais importantes na área de mineração de dados. Seguindo os critérios do Programa de Pós-Graduação da

UFSCar, todos os artigos foram publicados somente em anais de eventos qualificados pela CAPES, sendo eles:

- AYRES, R. M. J. ; Santos, M. T. P. . FOntGAR Algorithm: Mining Generalized Association Rules Using Fuzzy Ontologies. In: IEEE International Conference on Fuzzy Systems (Fuzz-IEEE), 2012, Brisbane. Proceedings of the 2012 IEEE International Conference on Fuzzy Systems, 2012.

- AYRES, R. M. J; Santos, M. T. P. . OntGAR Algorithm: An Ontology-Based Algorithm for Mining Generalized Association Rules. In: International Conference on Fuzzy Systems and Knowledge Discovery, 2012, Chongqing. Proceedings of the International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), 2012.

- AYRES, R. M. J. ; Ribeiro, M. X. ; Santos, M. T. P. . Exploring Fuzzy Ontologies in Mining Generalized Association Rules. In: International Conference on Computational Science and its Applications (ICCSA), 2012, Salvador. Proceedings of the 2012 International Conference on Computational Science and its Applications, 2012.

- AYRES, R. M. J. ; Santos, M. T. P. . Mining Generalized Association Rules Using Fuzzy Ontologies with Context-Based Similarity. In: International Conference on Enterprise Information Systems (ICEIS), 2012, Wroclaw. Proceedings of the 2012 International Conference on Enterprise Information Systems, 2012.

- AYRES, R. M. J. ; Miani, R.G. ; Santos, M. T. P. . A New Equation For Calculating The Weight of Fuzzy Itemsets. In: 2011 IEEE International Conference on Information Reuse and Integration (IRI), 2011, Las Vegas. Proceedings of the 2011 IEEE International Conference on Information Reuse and Integration, 2011.

## 8.3 Limitações

Embora tenha sido utilizado o conceito de similaridade, as relações exploradas por este trabalho estão inseridas apenas entre itens folha da ontologia fuzzy. Sendo assim, sabendo que tais relações não se restringem somente a este caso, o trabalho não explora as mesmas em níveis superiores da estrutura, o que demandaria estudos relevantes nessa linha.

Um dos principais pontos explorados por este trabalho é a questão da inserção de graus de especialização/generalização entre os termos da ontologia

utilizada. No entanto, o trabalho se limita a atacar ontologias em que um filho pertença (com grau variando em [0,1]) a somente um único pai, deixando de explorar os casos em que um filho pertença a mais de um. Isso ocorre porque, embora o algoritmo de agrupamento e substituição tenha sido proposto para atender a essas situações, o algoritmo utilizado no cálculo de suporte e confiança não suporta esses casos, devido às muitas ramificações possíveis no caminho entre um item filho e o ancestral analisado.

## 8.4 Trabalhos Futuros

Nas subseções seguintes serão apresentados alguns assuntos interessantes a serem explorados em trabalhos futuros.

### 8.4.1 Realização de mais testes

Conforme apresentado no Capítulo 7, os testes foram realizados, dentre outras coisas, para verificar os impactos de se realizar a mineração utilizando ontologias em que os graus de especialização podem variar em [0,1]. Entretanto, seria interessante também comparar FOntGAR com outra abordagem que utiliza o mesmo tipo de estrutura taxonômica. Nesse caso, como os trabalhos anteriores estão relacionados com a etapa de pré-processamento, poderiam ser feitas comparações entre o uso de transações estendidas e a abordagem de pós-processamento, identificando pontos positivos e negativos de cada uma. Além disso, novas bases de dados e ontologias difusas poderiam ser utilizadas, aplicando o algoritmo em outros domínios.

Os experimentos realizados até o momento foram de caráter quantitativo, entretanto, seria interessante também a análise qualitativa. Nesse caso, ela poderia ser direcionada em relação à semântica das regras, ou seja, até que ponto elas realmente fornecem um conhecimento mais interessante. Para isso, seria importante que as análises fossem realizadas juntamente com especialistas de domínio.

### 8.4.2 Desenvolvimento de interface gráfica

Considerando que o objetivo principal da mineração é fornecer padrões interessantes e compreensíveis, seria interessante que fosse desenvolvido um módulo de visualização, no qual diversos algoritmos pudessem ser selecionados de acordo com a escolha do usuário. Uma área interessante que trata sobre assunto é a mineração visual.

### 8.4.3 Automatização de parâmetros providos pelo usuário

Um dos fatores críticos em algoritmos de mineração é a quantidade de parâmetros providos pelo usuário, que em geral são pouco compreensíveis. Muitas vezes, pela falta de entendimento, esses parâmetros não são bem definidos, e o resultado da mineração não ocorre da melhor maneira. Sendo assim, um trabalho relevante seria estudar métodos de automatização de parâmetros em algoritmos de mineração, utilizando, por exemplo, as preferências ou perfis do usuário.

A questão de preferências também poderia ser levada em consideração na definição dos níveis de generalização. No algoritmo FOntGAR adota-se a estratégia de generalizar até um nível abaixo da raiz, entretanto, ao contrário disso o algoritmo poderia generalizar até um nível “preferido” pelo usuário. Técnicas de mineração de textos poderiam ser estudadas para auxiliar esse processo.

# REFERÊNCIAS


ADAMO, J.M. **Data Mining for Association rules and sequential patterns**. Springer-Verlag., 2001. (59, 60).

ADOMAVICIUS, G.; TUZHILIN, A. Expert-Driven Validation of Rule-Based User Models. **Data Mining Knowledge Discovery,** v. 5, n. 1-2, p. 33-58, 2001. ISSN 1384-5810.

AGRAWAL, R.; IMIELINSKI, T.; SWAMI, A. Mining association rules between sets of items in large databases. In: BUNEMAN, P. e JAJODIA, S., 1993. Washington, DC, USA. Publ by ACM. p.207-216.

AGRAWAL, R.; SRIKANT, R. Fast algorithms for mining association rules. In: Conference on Very Large Databases (VLDB), 1994, Santiago, Chile. **Proceedings,** Morgan Kaufmann Publischers Inc., 1994, p. 487-499.

BERNERS-LEE, T.; HENDLER, J.; LASSILA, O. The Semantic Web. **Scientific American,** v. 284, n. 5, p. 34-43, 2001. ISSN 0036-8733. Disponível em: <http://dx.doi.org/10.1038/scientificamerican0501-34 >.

BOJADZIEV, G.; BOJADZIEV, M. **Fuzzy Sets, Fuzzy Logic, Applications**. River Edge, NJ, USA, World Scientific Publishing Co., 1996. 283

BRACHMAN, R. J.; ANAND, T. The Process of Knowledge Discovery in Databases: A First Sketch. In: AAAI Workshop on Knowledge Discovery in Databases, 1994, Seattle, Washington, USA. **Proceedings**, Seattle, Washington, USA**:** 19941-12 p..

BRAGA, L. P. V. **Introdução à Mineração de Dados**. 2. Rio de Janeiro: E-papers Serviços Editoriais, 2005. 211p.

CAI, C. H. et al. Mining Association Rules with Weighted Items. In: International Database Engineering and Application Symposium, 1998. p.68-77.

CALEGARI, S.; CIUCCI, D. Fuzzy Ontology, Fuzzy Description Logics and Fuzzy-OWL. In: MASULLI, F.; MITRA, S.*, et al*. **Applications of Fuzzy Sets Theory.** Berlin / Heidelberg: Springer, 2007. v.4578, p.118-126.

CARVALHO, V. O. D.; REZENDE, S. O.; CASTRO, M. D. Obtaining and evaluating generalized association rules. In: International Conference on Enterprise Information Systems, 2007, Funchal, Madeira: p.310-315.

CERRI, M. J. et al. UFOCoRe: Exploring Fuzzy Relations According to Specifics Contexts. In: International Conference on Software Engineering & Knowledge Engineering, 2010, San Francisco Bay, USA**:** p. 529-534.

CHEN, G.; WEI, Q. Fuzzy association rules and the extended mining algorithms. **Information Sciences - Informatics and Computer Science: An International Journal,** v. 147, n. 1-4, p. 201-228, 2002.

DUBOIS, D.; PRADE, H. **Fuzzy Sets and systems: Theory and Applications**. New York: Academic Press, 1980.

DOMINGUES, M. A. **Generalização de regras de associação**. 2004. Dissertação (Mestrado em Ciências de Computação e Matemática Computacional) - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2004.

ELMASRI, R.; NAVATHE, S. B. **Sistemas de Banco de Dados**. 4. São Paulo: Pearson Addison Esley, 2005.

ESCOVAR, E. L. G. **Algoritmo *SSDM* para a Mineração de Dados Semanticamente Similares**. 2004. Dissertação (Mestrado em Ciência da Computação) - Departamento de Computação, Universidade Federal de São Carlos, São Carlos, 2004.

ESCOVAR, E. L. G.; YAGUINUMA, C. A.; BIAJIZ, M. Using Fuzzy Ontologies to Extend Semantically Similar Data Mining. In: Brazilian Symposium on Databases, 21, 2006, Florianópolis, Brazil**:** 2006, p.16-30.

FAYYAD, U.; PIATETSKY-SHAPIRO, G.; SMYTH, P. The KDD Process for Extracting Useful Knowledge from Volumes of Data. **Communications of the ACM,** v. 39, n. 11, p. 27-34, 1996a.

FAYYAD, U.; PIATETSKYSHAPIRO, G.; SMYTH, P. From data mining to knowledge discovery in databases. **Ai Magazine,** v. 17, n. 3, p. 37-54, Fal 1996b.

FRAWLEY, W. J.; PIATETSKY-SHAPIRO, G.; MATHEUS, C. J. Knowledge Discovery in Databases: An Overview. **AI Magazine,** v. 13, n. 3, p. 57-70, 1992.

GRUBER, T. A translation approach to portable ontology specifications. **Knowledge Acquisition,** v. 5, n. 2, p. 199-220, 1993.

_____. Toward Principles for the Design of Ontologies Used for Knowledge Sharing. **International Journal of Human-Computer Studies,** v. 43, n. 5-6, p. 907-928, 1995.

GUARINO, N. Understanding, building and using ontologies. **International Journal of Human-Computer Studies,** v. 46, n. 2-3, p. 293-310, 1997.

_____. Formal Ontology and Information Systems. In: International Conference on Formal Ontologies in Information Systems, 1998. Trento, Italy. p.3-15.

HAN, J.; KAMBER, M. **Data Mining: Concepts and Techiniques**. 2. San Francisco, CA: Morgan Kaufmann, 2006. 743

HAND, D.; MANNILA, H.; SMYTH, P. **Principles of Data Mining**. Cambrige, Massachusetts: The MIT Press, 2001. 546

HONG, T.-P.; LIN, K.-Y.; WANG, S.-L. Fuzzy data mining for interesting generalized association rules. **Fuzzy Sets and Systems,** v. 138, n. 2, p. 255-269, 2003.

HUNG-PIN, C.; YI-TSUNG, T.; KUN-LIN, H. A Cluster-Based Method for Mining Generalized Fuzzy Association Rules. In: International Conference on Innovative Computing, Information and Control, 1, 2006, p.519-522.

JIAWEI HAN; FU, Y. **Discovery of Multiple-Level Association Rules from Large Databases**. In: VLDB Conference, 21,1995, Zurich, Switzerland**:** 1995, p. 420-431.

KEON-MYUNG, L. Mining generalized fuzzy quantitative association rules with fuzzy generalization hierarchies. In: IFSA World Congress and 20th NAFIPS International Conference, 2001, p.2977-2982.

KLIR, G. J.; YUAN, B. **Fuzzy Sets and Fuzzy Logic - Theory and Applications.** Upper Sanddle River, New Jersey, USA: Pratince Hall PTR, 1995. 592

LEE, J.-H.; LEE-KWANG, H. **An Extension of Association Rules Using Fuzzy Sets**. In: International Fuzzy Systems Association World Congress, 7, 1997, Prague. **Proceedings…** Czech**:** 1997, p. 399-402.

LEE, Y.-C.; HONG, T.-P.; WANG, T.-C. Multi-level fuzzy mining with multiple minimum supports. **Expert Systems with Applications: An International Journal,** v. 34, n. 1, p. 459-468, 2008.

MAGALHÃES JUNIOR, W. C. P. D. **Método @Risk-On para Pré-processamento de análises de Risco em Analitos e Amostras de Leite baseado em Ontologias e Lógica Difusa**. 2010. Monografia de Qualificação (Mestrado em Ciência da Computação) - Departamento de Computação, Universidade Federal de São carlos, São Carlos, 2010.

MAHMOUDI, E. V. et al. Mining generalized fuzzy association rules via determining minimum supports. In: Iranian Conference on Electrical Engineering, 19, 2011, p.1-6.

MIANI, R. G. **Algoritmo NARFO para Mineração de Regras de Associação Generalizadas não Redundantes Baseada em uma Ontologia Difusa**. 2009. Dissertação (Mestrado em Ciência da Computação) - Departamento de Computação, Universidade Federal de São Carlos, São Carlos, 2009.

MITRA, S.; ACHARYA, T. **Data mining : multimedia, soft computing, and bioinformatics**. Hoboken, NJ: Wiley, 2003.

MOHAMADLOU, H. et al. A method for mining association rules in quantitative and fuzzy data. In: International Conference on Computers & Industrial Engineering, 2009. p.453-458.

NOY, N.; MCGUINNESS, D. **Ontology Development 101: A Guide to Creating Your First Ontology**. Stanford Knowledge Systems Laboratory Techinical. Stanford, p.1-25. 2001

ORTEGA, N. R. S. **Aplicação da Teoria de Conjuntos Fuzzy a Problemas da Biomedicina**. 2001. 152 Tese (Doutorado em Ciências) - Instituto de Física, Universidade de São Paulo, São Paulo, 2001.

PASQUIER, N. et al. Efficient mining of association rules using closed itemset lattices. **Information Systems,** v. 24, n. 1, p. 25-46, 1999. ISSN 0306-4379.

PEDRYCZ, W. Fuzzy set technology in knowledge discovery. **Fuzzy Sets and Systems,** v. 98, n. 3, p. 279-290, 1998.

SHUHONG, Z.; JIANXUN, S.; PENGCHENG, W. Research on the Fuzzy Quantitative Association Rules Mining Algorithm and Its Simulation. In: International Conference on Fuzzy Systems and Knowledge Discovery, 4, 2007: Haikou, p.401-405.

SMITH, M. K.; WELT, C.; MCGUINNESS, D. L. W3C Proposed Recomendation: OWL Web Ontology Language Guide. 2004. Disponível em: < http://www.w3.org/TR/2004/REC-owl-guide-20040210>.

SRIKANT, R.; AGRAWAL, R. **Mining Generalized Association Rules**. In: International Conference on Very Large Data Bases, 21, 1995. **Proceedings…** Morgan Kaufmann Publishers Inc. 1995.

_____. Mining Quantitative Association Rules in Large Relational Tables. In: JAGADISH, H. V. e MUMICK, I., In: ACM {SIGMOD} International Conference on Management of Data, 1996. p.1-12.

STRACCIA, U. Chapter 4 A fuzzy description logic for the semantic web. In: ELIE, S. (Ed.). **Capturing Intelligence**: Elsevier, v.Volume 1, 2006. p.73-90.

TERANO, T.; ASAI, K.; SUGENO, M. **Fuzzy systems theory and its applications**. Academic Press Professional, Inc., 1992. 268

USCHOLD, M.; GRUNINGER, M. Ontologies: principles, methods, and applications. **Knowledge Engineering Review,** v. 11, n. 2, p. 93-155, 1996.

_____. Ontologies and semantics for seamless connectivity. **ACM SIGMOD Rec.,** v. 33, n. 4, p. 58-64, 2004.

WAI-HO, A.; CHAN, K. C. C. FARM: a data mining system for discovering fuzzy association rules. In: IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), 1999, p.1217-1222.

WEI, Q.; CHEN, G. Mining generalized association rules with fuzzy taxonomic structures. In: International Conference of the North American Fuzzy Information Processing Society (NAFIPS), 1999, 18th, 1999. p.477-481.

WEN-YANG, L.; MING-CHENG, T.; JA-HWUNG, S. Updating generalized association rules with evolving fuzzy taxonomies. In: IEEE International Conference on 1Fuzzy Systems (FUZZ IEEE), 2010. p.1-6.

YAGUINUMA, C. A. **Sistema FOQuE para Expansão Semântica de Consultas baseada em Ontologias Difusas**. 2007. Dissertação (Mestrado em Ciência da Computação) - Departamento de Computação, Universidade Federal de São Carlos, São Carlos, 2007.

YAGUINUMA., C.; SANTOS., M.; BIAJIZ., M. Fuzzy Meta-ontology for Representation of Imprecise Information in Ontologies. In: WOMSDE, 2007. p.57-67.

ZADEH, L. A. Fuzzy sets. **Information and Control,** v. 8, n. 3, p. 338-353, 1965.

_____. Fuzzy logic. **Computer,** v. 21, n. 4, p. 83-93, 1988.

_____. Is there a need for fuzzy logic? **Information Sciences,** v. 178, n. 13, p. 2751-2779, 2008.