Instituto de Ciências Matemáticas e de Computação

ISSN - 0103-2569

Descrição do Módulo de Análise de Regras do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$

Alan Keller Gomes
Maria Carolina Monard

Nº 155

RELATÓRIOS TÉCNICOS DO ICMC

São Carlos

Fevereiro/2002

Descrição do Módulo de Análise de Regras do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$*

Alan Keller Gomes
Maria Carolina Monard

Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação
Departamento de Ciências de Computação e Estatística
Laboratório de Inteligência Computacional
Caixa Postal 668, 13560-970 - São Carlos, SP, Brasil
e-mail: {akgomes, mcmonard}@icmc.sc.usp.br

Resumo

Encontra-se em desenvolvimento no Laboratório de Inteligência Computacional — LABIC — um projeto de grande porte denominado DISCOVER, que tem objetivo fornecer um ambiente integrado para apoiar as etapas do processo de descoberta de conhecimento.

No intuito de testar algumas idéias que poderão futuramente ser incorporadas no ambiente DISCOVER, foi proposto um sistema computacional protótipo, implementado na linguagem de programação lógica Prolog, denominado $\mathcal{R}_{ule}\mathcal{S}_{ystem}$.

O $\mathcal{R}_{ule}\mathcal{S}_{ystem}$ tem como objetivo (a) avaliar regras de conhecimento induzidas por algoritmos de Aprendizado de Máquina simbólico (b) avaliar diversas formas de combinar classificadores simbólicos bem como explicar a classificação de novos exemplos realizada por um *ensemble* de classificares simbólicos.

O objetivo (a) é implementado pelo Módulo de Análise de Regras e o objetivo (b) pelo Módulo de Combinação e Explicação. Esses são os dois módulos principais do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$.

Este trabalho tem por objetivo apresentar as funcionalidades implementadas no Módulo de Análise de Regras.

Palavras-Chave: Aprendizado de Máquina, Análise de Regras, Avaliação de Regras, Prolog.

Fevereiro 2002

*Trabalho realizado com auxílio do CNPq, CAPES e FAPESP.

Sumário

1 Introdução **1**

2 Aprendizado de Máquina Supervisionado **2**

2.1 Classificadores Simbólicos 3

2.2 Avaliação de Regras 3

2.3 Ensemble de Classificadores 3

3 Arquitetura do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$ **4**

4 Metodologia para Documentação do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$ **5**

4.1 Componentes do Diagrama de Fluxo de Execução de Procedimentos 5

4.1.1 Procedimentos 5

4.1.2 Condições 7

4.1.3 Fluxo de Execução 9

4.2 Descrição do Fluxo de Execução do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$ 10

5 Módulo de Análise de Regras **12**

5.1 Análise Qualitativa de Regras 12

5.1.1 Medidas de Avaliação de Regras 13

5.1.2 Procedimentos que Implementam Medidas de Avaliação de Regras 17

5.2 Análise Quantitativa de Regras 19

5.2.1 Medidas de Interessabilidade de Regras 19

5.2.2 Procedimentos que Implementam Medidas de Interessabilidade de Regras . 21

6 Exemplo de Execução **23**

7 Considerações Finais **26**

Referências Bibliográficas **27**

Lista de Figuras

1 Arquitetura do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$ 4

2 Diagrama de Fluxo de Execução de Procedimentos do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$ 6

3 Exemplo de Representação de um Procedimento 7

4 Exemplo de Representação de um Conjunto de Condições 7

5 Exemplo de Representação de um Subconjunto de Condições 8

6 Tipos de Barramento 10

7 Regras *Interclass* — Conjunto de Subconjuntos de Regras 15

8 Arquivo de Exemplos 23

9 Arquivo de Regras *Interclass* Induzidas por $\mathcal{C}4.5rules$ 23

10 Regras na Base de Fatos após a Execução do Procedimento `evaluateAllSetOf-RuleFrequency/1` 24

11 Surpresa dos Atributos Individuais das Regras 24

12 Medidas Qualitativas das Regras 25

13 Surpresa dos Pequenos Disjuntos das Regras 26

Lista de Tabelas

1 Conjunto de Exemplos no Formato Atributo-Valor 2

2 Procedimentos Principais do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$ 5

3 Condições Pertencentes ao Conjunto de Condições E 8

4 Condições Pertencentes ao Conjunto de Condições R 8

5 Condições 1, 2, 3, 4 e 5 9

6 Identificação das Medidas de Avaliação da Regra 19

Lista de Algoritmos

1 Cálculo de Informações para Avaliação de Regras *Unordered* 13

2 Cálculo de Informações para Avaliação de Regras *Ordered* 14

3 Cálculo das Informações para um Conjunto de Hipóteses $\mathcal{H}$ 18

4 Cálculo do Grau de Surpresa de Pequenos Disjuntos 22

[1]Este documento foi elaborado com o LaTeX, sua bibliografia é mantida com o auxílio da ferramenta BibView (Prati et al., 1999) e gerada automaticamente pelo BibTeX

1 Introdução

Encontra-se em desenvolvimento no Laboratório de Inteligência Computacional —LABIC[1]— um projeto de grande porte denominado DISCOVER, inicialmente proposto por Baranauskas e Batista (Baranauskas and Batista, 2000). O projeto DISCOVER tem como objetivo fornecer um ambiente integrado para apoiar as etapas do processo de descoberta de conhecimento, oferecendo funcionalidades voltadas para Aprendizado de Máquina (AM) (Batista, 1997; Caulkins, 2000; Milaré, 2000; Martins, 2001; Pila, 2001), Mineração de Dados[2](MD) (Batista, 2000; Félix, 1998; Horst, 1999; Lee, 2000; Nagai, 2000; Pugliesi, 2001; Baranauskas, 2001) e Mineração de Textos[3](MT) (Imamura, 2001).

As funcionalidades voltadas para AM consideram, entre outros, um formato padrão para as regras induzidas por algoritmos de AM simbólico, denominado $\mathcal{PBM}$ (Prati et al., 2001b; Prati et al., 2001a) bem como um formato padrão para os exemplos utilizados (Batista, 2001).

No intuito de testar algumas idéias que poderão futuramente ser implementadas no ambiente DISCOVER, foi proposto em (Gomes, 2001) e (Bernardini, 2001) um sistema computacional, denominado $\mathcal{R}_{ule}\mathcal{S}_{ystem}$, composto de dois módulos principais, os quais são:

1. Módulo de Análise de Regras (MAR), com o objetivo de implementar diversas medidas de avaliação de regras propostas na literatura para efetuar a análise automática de regras induzidas por algoritmos de AM simbólico; e

2. Módulo de Combinação e Explicação (MCE), com o objetivo de construir, dado um conjunto de classificadores simbólicos, um outro classificador simbólico que contenha regras desse conjunto de classificadores que sejam as "melhores" segundo as diversas medidas de qualidade de regras implementadas no MAR. Outro objetivo deste módulo é fornecer ao usuário, dado um conjunto de classificadores simbólicos e um exemplo do domínio dos classificadores, quais as regras que foram disparadas nesses classificadores e qual a "melhor" regra, segundo, também, alguma das diversas medidas de avaliação de regras implementadas no MAR. Em outras palavras, fornecer ao usuário uma explicação da classificação.

Foi decidido implementar o $\mathcal{R}_{ule}\mathcal{S}_{ystem}$ na linguagem de programação lógica Prolog (Bratko, 1990) mais especificamente em LPA-PROLOG (Westwood, 2000b; Westwood, 2000a). A escolha da linguagem Prolog deve-se ao fato de ser uma linguagem apropriada para o desenvolvimento de protótipos nos quais novas idéias possam ser facilmente testadas antes de serem implementadas em um sistema de grande porte. Uma outra característica importante de Prolog é a possibilidade de realizar consultas para obter informações que não ficam restritas àquelas pré-definidas no sistema.

Os dados de entrada para o $\mathcal{R}_{ule}\mathcal{S}_{ystem}$ consiste de:

1. um ou vários conjuntos de regras induzidas por algoritmos de AM simbólico; e

[1] http://labic.icmc.sc.usp.br
[2] Data Mining
[3] Text Mining

2. um conjunto de exemplos.

Ambos os dados de entrada estão na sintaxe padrão Prolog, que é equivalente à sintaxe padrão $\mathcal{PBM}$ de regras e exemplos adotada no projeto DISCOVER. A conversão do formato padrão $\mathcal{PBM}$ para a sintaxe padrão Prolog, descrita em detalhes em (Gomes et al., 2002), foi implementada através de scripts Perl.

O objetivo deste trabalho é descrever em detalhes o projeto e a implementação do Módulo de Análise de Regras (MAR) do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$. Também, para ilustrar melhor os procedimentos do MAR, é mostrado um exemplo de execução utilizando um conjunto de exemplos simples.

O trabalho está organizado da seguinte forma: na Seção 2 é feita uma breve descrição sobre alguns conceitos de aprendizado de máquina, necessários para um melhor entendimento do trabalho realizado. Na Seção 3, é apresentada a arquitetura do sistema $\mathcal{R}_{ule}\mathcal{S}_{ystem}$. Na Seção 4, é apresentada uma metodologia para a documentação do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$. Na Seção 5, é apresentada a descrição dos procedimentos que constituem o Módulo de Análise de Regras. Na Seção 6, é mostrado um exemplo de utilização desse módulo e, finalmente, na Seção 7, são apresentadas as conclusões do trabalho realizado.

2 Aprendizado de Máquina Supervisionado

No problema padrão de AM supervisionado, ao algoritmo de aprendizado de máquina é dado um conjunto de exemplos S de treinamento, com N exemplos $T_i, i = 1, ..., N$, escolhidos de um domínio X com uma distribuição D fixa, desconhecida e arbitrária, da forma $\{(\mathbf{x}_1,y_1),...,(\mathbf{x}_N,y_N)\}$ para alguma função desconhecida $y = f(\mathbf{x})$, como mostra a Tabela 1. Os $\mathbf{x}_i$ são tipicamente vetores da forma $<x_{i1},x_{i2},...,x_{iM}>$, com valores discretos ou numéricos, como peso, altura, cor, idade, e assim por diante. Assim, x_{ij} refere-se ao valor do atributo (ou *feature*) j, denominado $\mathbf{X}_j$ do exemplo T_i, como mostra a Tabela 1. Os valores y_i referem-se ao valor do atributo Y, freqüentemente denominado classe.

	X_1	X_2	$\dots$	X_M	Y
T_1	x_{11}	x_{12}	$\dots$	x_{1M}	y_1
T_2	x_{21}	x_{22}	$\dots$	x_{2M}	y_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
T_N	x_{N1}	x_{N2}	$\dots$	x_{NM}	y_N

Tabela 1: Conjunto de Exemplos no Formato Atributo-Valor

Os valores de y são tipicamente pertencentes a um conjunto discreto de classes $C_v, v = 1, ..., NCl$, da forma $\{C_1, ..., C_{NCl}\}$, quando se trata de *classificação* ou ao conjunto de números reais em caso de *regressão*.

Dado um conjunto S de exemplos de treinamento a um algoritmo de AM, um *classificador* $\mathbf{h}$ será induzido. O classificador consiste da *hipótese* feita sobre a verdadeira (mas desconhecida) função f. Dados novos exemplos $\mathbf{x}$, o classificador, ou hipótese, $\mathbf{h}$ prediz o valor correspondente y.

2.1 Classificadores Simbólicos

Algoritmos de AM simbólicos induzem o classificador **h** de tal forma que o conceito descrito por **h** é facilmente interpretável por seres humanos e, geralmente, pode ser transformado em conjuntos de regras *if-then*, ou seja, regras do tipo *Corpo* $\rightarrow$ *Cabeça* ou $Body \rightarrow Head$. Uma regra R_u pode então ser resumidamente denotada como $B \rightarrow H$.

Pode-se medir um conjunto de regras **h** dado um conjunto de exemplos S. Neste caso, **h** consiste de um conjunto de regras com NR regras $R_u, u = 1, ..., NR$, ou seja, $\mathbf{h} = \{R_1, ..., R_{NR}\}$, o qual denominamos classificador simbólico. Neste caso, além de medir a precisão de **h** como um todo, isto é, como um classificador tipo caixa preta, é possível avaliar separadamente cada uma das regras que constituem **h**.

2.2 Avaliação de Regras

Várias medidas de avaliação de regras têm sido propostas na literatura (Lavrac et al., 1999; Freitas, 1998b; Freitas, 1998a; Hilderman and Hamilton, 1999). Neste trabalho, serão utilizadas medidas de avaliação de regras propostas por Lavrac, em (Lavrac et al., 1999), as quais podem ser divididas em três categorias distintas:

1. Medidas genéricas de avaliação de regras;
2. Medidas relativas de avaliação de regras; e
3. Medidas relativas de avaliação de regras com peso.

O Módulo de Análise de Regras, descrito neste trabalho, implementa medidas de avaliação de regras nessas três categorias, as quais estão descritas sucintamente na Tabela 6 na página 19, bem como o nome pelo qual cada uma dessas medidas é referenciada nos procedimentos do MAR. Para uma descrição completa das medidas pertencentes a cada uma dessas categorias, veja (Gomes, 2001).

2.3 Ensemble de Classificadores

Dado um conjunto $\mathcal{H}$ de classificadores, denotados por $\mathbf{h}_1$,...,$\mathbf{h}_L$, pode-se combinar suas decisões individuais — tipicamente através da votação com ou sem peso — para classificar novos exemplos, formando, assim, um **ensemble** de classificadores. Entretanto, um ensemble de classificadores é um classificador caixa preta, isto é, sem condições de fornecer ao usuário uma explicação das suas decisões, independentemente de cada classificador $\mathbf{h}_1$,...,$\mathbf{h}_L$ que constitui o ensemble ser ou não ser simbólico.

O Módulo de Combinação e Explicação do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$, descrito em (Bernardini and Monard, 2002), tem por objetivo testar a possibilidade de, dado um conjunto $\mathcal{H} = \{\mathbf{h}_1, ..., \mathbf{h}_L\}$ de classificadores simbólicos, construir outro classificador simbólico $\mathbf{h}^*$ que consiste das "melhores" regras do conjunto $\mathcal{H}$ de classificadores simbólicos. Para determinar essas melhores regras, são utilizadas as medidas genéricas de avaliação de regras, mostradas na Tabela 6, implementadas no MAR.

A arquitetura do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$ é descrita a seguir[4].

3 Arquitetura do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$

A Figura 1 mostra a arquitetura do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$. Os dois módulos principais (MAR e MCE) consistem de um conjunto de procedimentos Prolog específicos de cada módulo. O Módulo Auxiliar (MA) contém procedimentos para ler arquivos especificados pelo usuário — arquivos que contém conjuntos de regras e o conjunto de exemplos na sintaxe padrão Prolog —, bem como uma biblioteca de procedimentos Prolog auxiliares, os quais são compartilhados pelos procedimentos pertencentes ao MAR e ao MCE. Deve ser observado que o usuário não tem acesso aos procedimentos auxiliares do MA.

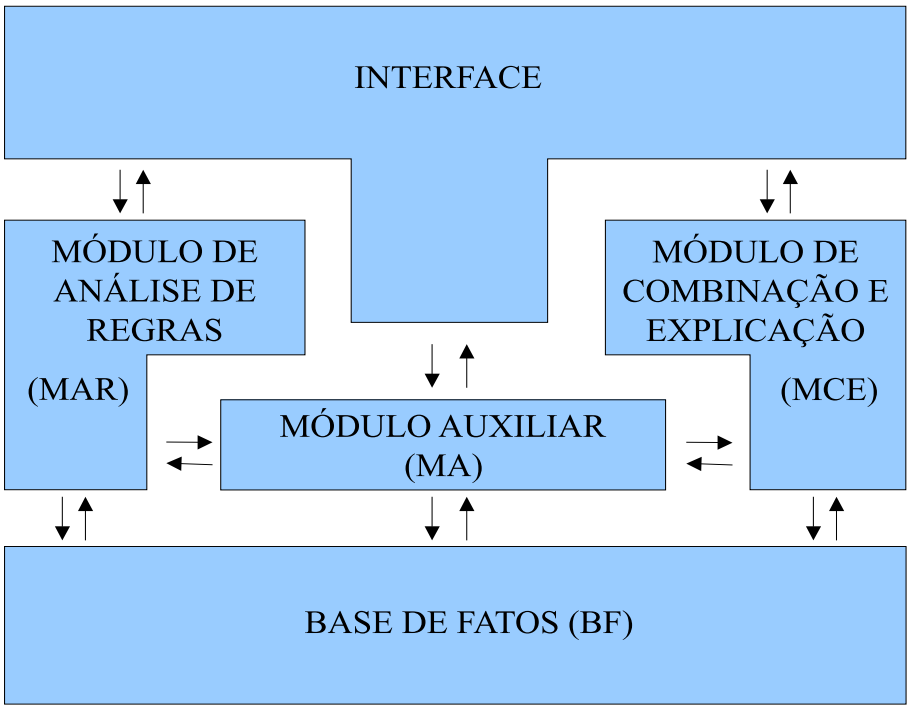

Figura 1: Arquitetura do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$

A Base de Fatos (BF) é utilizada para armazenar os dados de entrada (um ou mais conjuntos de regras e um único conjunto de exemplos) bem como para armazenar informações adicionais, resultantes da execução de algum procedimento ativado pelo usuário.

É importante observar que o $\mathcal{R}_{ule}\mathcal{S}_{ystem}$ é um sistema interativo, guiado pelas necessidades do usuário, ou seja, o usuário pode ativar os procedimentos principais de cada um dos três módulos.

A Tabela 2 mostra os procedimentos principais (`<nome do procedimento>/<aridade>`), acessíveis

[4] A descrição do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$ foi realizada conjuntamente com Flávia Bernardini. Assim, a mesma descrição encontra-se também em (Bernardini and Monard, 2002).

ao usuário, implementados em cada módulo do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$.

MA	MAC	MCE
`loadRules/0`	`evaluateAllSetOfRulesFrequency/1`	`rco/4`
`loadExamples/0`	`indAttSurp/1`	`classifyExample/2`
`verifyBase/0`	`calculateMeasuresOverAllSetOfRules/0`	`confusionMatrix/2`
	`smallDisjSurpr/3`	`classifyExampleByEnsemble/2`
		`kFoldCrossValidation/4`

Tabela 2: Procedimentos Principais do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$

Para a maioria desses procedimentos, há um conjunto de pré-condições que devem ser satisfeitas para sua correta execução. Também, vários desses procedimentos apresentam um conjunto de pós-condições. Essas condições (pré-condições para disparar um procedimento quanto pós-condições geradas após a execução de um procedimento) são armazenadas na Base de Fatos.

Com o objetivo de documentar o $\mathcal{R}_{ule}\mathcal{S}_{ystem}$, foi por nós desenvolvida uma metodologia específica, descrita a seguir.

4 Metodologia para Documentação do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$

Quando foi iniciada a implementação do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$ foram pesquisadas na literatura algumas metodologias que pudessem servir para documentar o sistema. Entretanto, nenhuma das metodologias pesquisadas foi suficientemente adequada para atender as especificidades de representação de pré e pós condições de procedimentos em fluxo de execução. Assim, foi decidido propor uma metodologia de representação na forma de um Diagrama de Fluxo de Execução de Procedimentos (PEFD — *Procedures Execution Flow Diagram*), para documentar o $\mathcal{R}_{ule}\mathcal{S}_{ystem}$. O PEFD do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$ é apresentado na Figura 2.

4.1 Componentes do Diagrama de Fluxo de Execução de Procedimentos

Os componentes do PEFD são 3:

1. Procedimentos;
2. Condições; e
3. Fluxos de Execução.

Cada um desses componentes é descrito a seguir.

4.1.1 Procedimentos

Um procedimento é representado por um retângulo identificado com o nome do procedimento. O módulo ao qual pertence esse procedimento encontra-se no canto superior direito, dentro de um

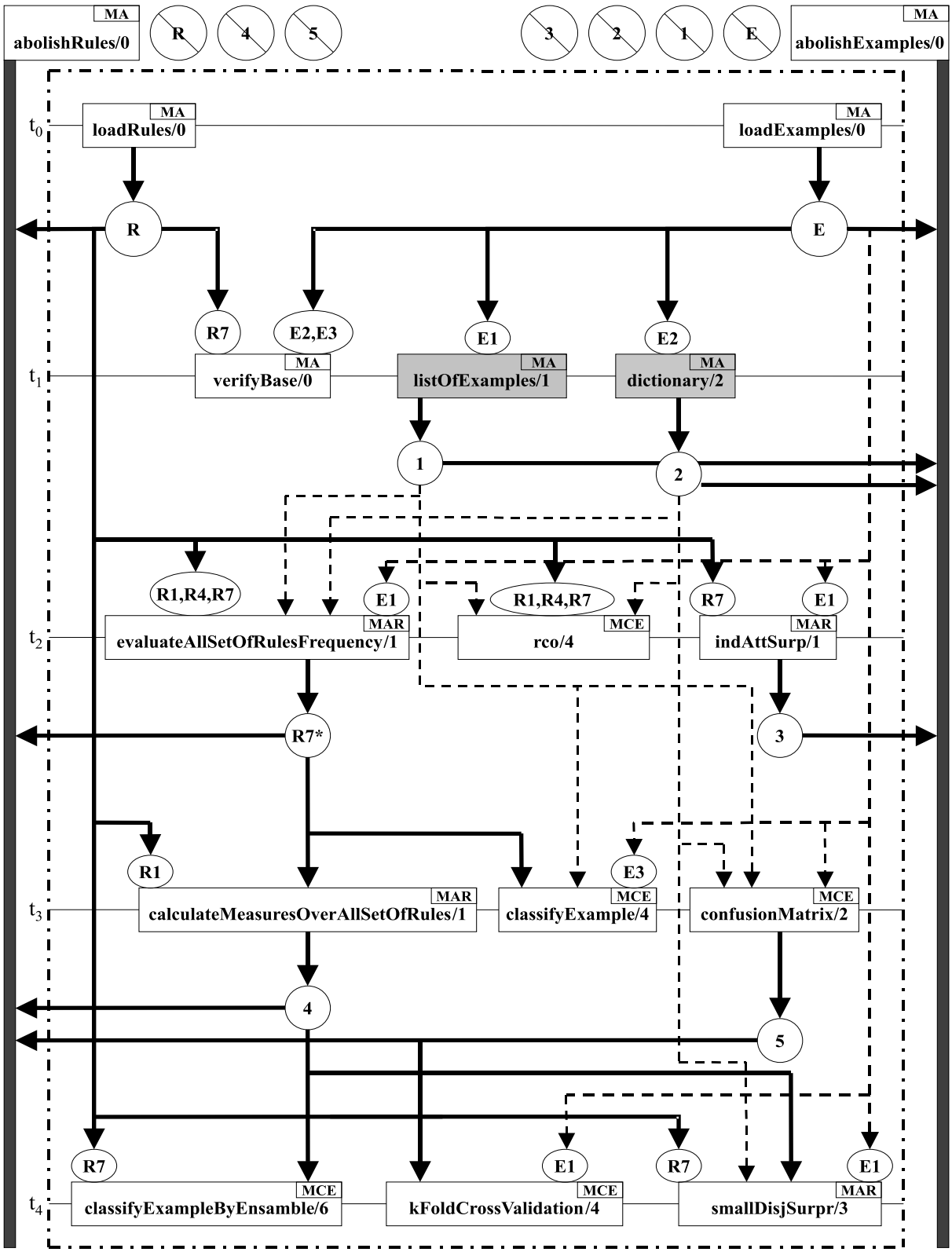

Figura 2: Diagrama de Fluxo de Execução de Procedimentos do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$

retângulo menor. No caso do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$, são 3 os módulos — MA, MAR e MCE. A Figura 3 mostra a representação do procedimento `evaluateAllSetOfRulesFrequency/1` do Módulo de Análise de Regras. Quando não for permitido ao usuário disparar um procedimento, o retângulo

Figura 3: Exemplo de Representação de um Procedimento

que se refere a esse procedimento encontrar-se preenchido com um tom de cinza claro. Por exemplo, na Figura 2, os procedimentos `listOfExamples/1` e `dictionary/2` não são acessíveis ao usuário.

4.1.2 Condições

As pré-condições necessárias para a execução de um procedimento, armazenadas previamente na Base de Fatos, bem como as pós-condições de um procedimento, armazenadas na BF após a sua execução, são representadas por círculos identificados por letras, letras seguidas de números, ou números, onde:

- `<letra>` nomeia um conjunto de condições;
- `<letra><númeroc>` nomeia a condição `<númeroc>` que pertence ao conjunto de condições nomeado `<letra>`; e
- `<número>` nomeia uma única condição que não pertence a nenhum conjunto de condições.

No $\mathcal{R}_{ule}\mathcal{S}_{ystem}$, essas condições são:

`<letra>:=E|R` onde `E={E1,E2,E3}` e `R={R1,R2,R3,R4,R5,R6,R7}`.

`<número>:=1|2|3|4|5`

O conjunto de condições `E` e `R` referem-se, respectivamente, a condições (informações) relacionadas ao conjunto de exemplos e ao conjunto de regras, todas elas na sintaxe padrão Prolog (Gomes et al., 2002). A Figura 4 ilustra a representação do conjunto de condições `E`.

Figura 4: Exemplo de Representação de um Conjunto de Condições

Uma descrição dos fatos relacionados a cada condição que pertence ao conjunto de condições `E` é encontrada na Tabela 3.

`E1: ex/2`	Representa cada um dos exemplos do conjunto de dados.
`E2: feature/3`	Representa informações de cada um dos atributos do conjunto de dados.
`E3: classFeature/1`	Identifica qual o atributo classe do conjunto de dados.

Tabela 3: Condições Pertencentes ao Conjunto de Condições E

Analogamente, uma descrição dos fatos relacionados a cada condição que pertence ao conjunto de condições `R` é encontrada na Tabela 4.

`R1: inducer/2`	Identifica o algoritmo de AM utilizado para induzir as regras.
`R2: inputFile/2`	Identifica o arquivo onde encontram-se armazenadas as regras originais induzidas pelo algoritmo de AM identificado por `R1`.
`R3: dAte/2`	Identifica data e hora em que as regras foram induzidas.
`R4: evaluatedAs/2`	Indica a forma de avaliação de regras utilizada pelo algoritmo de AM identificado por `R1`.
`R5: nameFile/2`	Identifica o arquivo `.names` relacionado ao conjunto de exemplos utilizado para induzir as regras.
`R6: dataFile/2`	Identifica o arquivo `.data` relacionado ao conjunto de exemplos utilizado para induzir as regras.
`R7: rule/6`	Representa cada uma das regras induzidas pelo algoritmo de AM utilizado.

Tabela 4: Condições Pertencentes ao Conjunto de Condições R

Quando é necessário representar um subconjunto de um conjunto de condições, esse subconjunto é identificado pelas várias condições que o compõe, separadas por vírgula. Um exemplo pode ser visualizado na Figura 5 onde é ilustrado o subconjunto {`R1`, `R4` e `R7`} de `R`.

Figura 5: Exemplo de Representação de um Subconjunto de Condições

E, finalmente, na Tabela 5 segue uma descrição das condições `1`, `2`, `3`, `4` e `5`.

Foram também definidos os seguintes dois símbolos de representação do estado de uma condição na BF em um dado instante:

- `*`: para indicar uma pré-condição atualizada pelo procedimento. Isto é, a pré-condição aparece também como pós-condição do procedimento, mas com diferentes informações. Por exemplo,

1: `listOfExamples/1`	Representa uma lista de identificadores (número) de cada exemplo pertencente ao conjunto de exemplos.
2: `auxDic/2`	Representa um dicionário binário com informações sobre os atributos do conjunto de exemplos. Este dicionário é utilizado para minimizar a busca pelos atributos, melhorando assim o tempo de execução dos procedimentos.
3: `infoC/1`	Representa o ganho de informação do atributo classe do conjunto de exemplos.
4: `ruleInfo/3`	Representa os valores obtidos no cálculo das medidas de avaliação de regras utilizadas.
5: `cm/2`	Representa a matriz de confusão, para um determinado conjunto de regras, dado o conjunto de exemplos presente na base.

Tabela 5: Condições `1`, `2`, `3`, `4` e `5`

a pré-condição `R7` do procedimento `evaluateAllSetOfRulesFrequency/1`, na Figura 2, após a execução do procedimento será atualizada para a pós-condição `R7*`.

`\`: indica a eliminação ou inexistência da condição na Base de Fatos. Um exemplo é a inexistência ou eliminação do conjunto de condições `R` que aparece ao lado do procedimento `abolishRules/0` na Figura 2. Nesse caso, o conjunto de condições `R` não existe no estado inicial da base ou foi excluído após a execução de `abolishRules/0`. O mesmo vale para as condições `4` e `5`.

Além disso, pós-condições de alguns procedimentos podem ser pré-condições para outros procedimentos. Assim, a execução desses procedimentos deve obedecer um fluxo temporal, detalhado a seguir.

4.1.3 Fluxo de Execução

O fluxo temporal é representado através de linhas de tempo $t_0, t_1, t_2, ..., t_q$ as quais são somente aplicáveis a procedimentos representados dentro de um retângulo tracejado[5]. Procedimentos fora desse retângulo tracejado são *atemporais*, ou seja, podem ser executados a qualquer momento.

No caso do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$, foram identificadas quatro linhas de tempo, ou seja, t_0, t_1, t_2, t_3 — Figura 2. Nessa figura, pode ser observado que os procedimentos `abolishRules/0` e `abolishExamples/0` encontram-se fora do retângulo tracejado, portanto eles podem ser executados a qualquer momento. Em outras palavras, são procedimentos atemporais.

Considerando os procedimentos representados dentro do retângulo tracejado, pode-se observar que os procedimentos `loadRules/0` e `loadExamples/0` na linha t_0 devem ser executados antes que os procedimentos `verifyBase/0`, `listOfExamples/1` e `dictionary/2`, na linha t_1. Analogamente, esses procedimentos da linha t_1 devem ser executados antes que os procedimentos da linha t_2, e assim por diante.

[5] tracejado do tipo $\{- \cdot - \cdot -\}$.

Outro aspecto a ser observado é que podem ocorrer relacionamentos entre conjunto de condições e procedimentos, ou entre um conjunto de condições e um subconjunto desse conjunto de condições. Esses relacionamentos são representados por meio de uma linha, contínua ou tracejada[6], denominada *barramento*. Existem três tipos de barramento, ilustrados na Figura 6, e explicados a seguir:

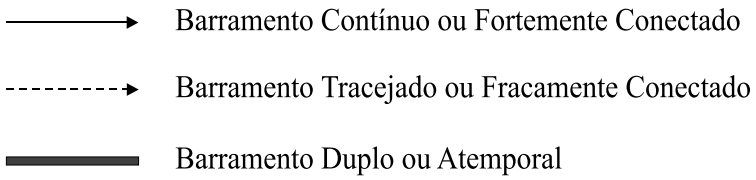

Figura 6: Tipos de Barramento

contínuo ou fortemente conectado indica que, se uma pré-condição for apagada da Base de Fatos, todas as pós-condições oriundas de procedimentos relacionados com essa pré-condição devem também ser apagadas da BF.

tracejado ou fracamente conectado indica que, se uma pré-condição for apagada da BF, as pós-condições oriundas de procedimentos relacionados com essa pré-condição não são apagadas da BF.

duplo ou atemporal indica que, ao executar um procedimento relacionado com esse tipo de barramento, todas as condições relacionadas serão apagadas da BF indistintamente.

Quando se relaciona um conjunto de condições a um procedimento, pode-se ter a necessidade de especificar qual é o subconjunto de condições que realmente faz parte do conjunto de pré-condições do procedimento. Para se detalhar esse tipo de informação, substitui-se o relacionamento entre o conjunto de condições e o procedimento em questão por um relacionamento entre o conjunto de condições e o subconjunto de pré-condições. Posiciona-se o círculo que representa esse subconjunto de pré-condições sobre o retângulo que representa o procedimento em questão.

Por exemplo, na Figura 2, o procedimento `verifyBase/0` somente necessita da pré-condição `R7` do conjunto de condições `R` e do subconjunto {`E2, E3`} de condições do conjunto de condições `E`.

4.2 Descrição do Fluxo de Execução do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$

Para um melhor entendimento do PEFD do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$(Figura 2), aqui é apresentada uma descrição do fluxo de execução dos procedimentos implementados nesse sistema.

Na parte superior da Figura 2 está representado o estado inicial da Base de Fatos. O estado inicial é caracterizado pela ausência de todas as condições que podem estar presentes na BF, ou também, pela exclusão de todas essas condições. Dessa forma, os procedimentos `abolishRules/0`

[6]tracejado do tipo $\{- - -\}$.

e `abolishExamples/0` são responsáveis por excluir, respectivamente, as condições `R, 4` e `5` e `E, 1, 2 e 3`. A ausência ou a exclusão dessas condições é representada por meio do traço que corta o círculo contendo a identificação de cada condição.

Considerando que nenhuma condição está presente na Base de Fatos, o início do fluxo de execução do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$ dá-se no instante t_0, quando podem ser executados seguidamente, em qualquer ordem, os procedimentos `loadRules/0` e `loadExamples/0`. Por exemplo, na execução de `loadRules/0` o conjunto de condições `R` é gravado na BF. Esse conjunto de condições é fortemente conectado ao barramento atemporal e pode ser removido da BF somente pela execução do procedimento `abolishRules/0`. Ao ser executado o procedimento `loadExamples/0`, o conjunto de condições `E` é gravado na BF. Esse conjunto de condições também é fortemente conectado ao barramento atemporal e apenas a execução do procedimento `abolishExamples/0` pode apagar esse conjunto de condições da BF.

Seguindo para o instante t_1, os procedimentos `verifyBase/0`, `listOfExamples/1` e `dictionary/2` podem ser executados seguidamente. O procedimento `verifyBase/0` tem como pré-condições subconjuntos de `R` e `E` aos quais encontram-se fortemente conectados. Especificamente, a pré-condição `R7` do conjunto de condições `R` e o subconjunto de condições `E2 e E3` do conjunto de condições `E`. A conexão forte indica que essas pré-condições específicas serão apagadas da BF se o conjunto `R` e `E` de condições for apagado da BF.

Ainda no instante t_1, os procedimentos `listOfExamples/1` e `dictionary/2` podem ser executados seguidamente em qualquer ordem. Como pode ser observado, os retângulos que identificam esses procedimentos estão preenchidos com um tom de cinza claro, o que significa que não podem ser acessados pelo usuário. Após serem executados, esses procedimentos gravam na BF, respectivamente, as condições `1` ou `2`. Essas duas condições estão fortemente conectadas ao barramento atemporal, portanto elas podem ser apagadas somente através da execução do procedimento `abolishExamples/0`.

Partindo para o instante t_2, os procedimentos `evaluateAllSetOfRulesFrequency/1`, `rco/4` e `indAttSurp/1` podem ser executados em qualquer ordem. Esses procedimentos estão fracamente conectados às pré-condições `1` e `2` e fortemente conectados com os conjuntos de condições `R` e `E`.

Por exemplo, o procedimento `evaluateAllSetOfRulesFrequency/1` está fracamente conectado às pré-condições `1` e `2`, fortemente conectado à condição `R` através do subconjunto de pré-condições `R1, R4, R7` e fracamente conectado com a pré-condição `E` através da pré-condição `E1`. Após sua execução, esse procedimento modifica a pré-condição `R7` gravando a pós-condição `R7*` na BF. Isso implica que, se `1`, `2` ou `E` forem apagadas da BF, a pós-condição `R7*` continua válida. `R7*` só pode ser apagada pela execução do procedimento `abolishRules/0`.

O fluxo de execução do PEFD do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$ nos instantes t_3 e t_4 é semelhante ao descrito no instante t_2.

Em qualquer dos instantes $t_0, ..., t_4$ os procedimentos atemporais `abolishRules/0` e `abolishExamples/0` podem ser executados. Por exemplo, se `abolishExamples/0` for executado no instante t_2, todas as condições presentes na BF até aquele instante e conectadas ao barramento atemporal serão excluídas — no caso, serão excluídas as condições `E`, `1` e `2`. Deve ser observado que os procedimentos atemporais `abolishRules/0` e `abolishExamples/0` retornam a BF ao seu estado inicial, ou seja, as condições ligadas a esses procedimentos (respectivamente `R, 4` e `5` e `E, 1, 2`

e 3) são apagadas da BF por meio do barramento fortemente conectado.

Na seção seguinte é apresentado o Módulo de de Análise de Regras do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$ — Figura 1 na página 4. O Módulo de Combinação e Explicação encontra-se detalhado em (Bernardini and Monard, 2002).

5 Módulo de Análise de Regras

Na Figura 2, página 6, os procedimentos que fazem parte do Módulo de Análise de Regras são identificados por meio da sigla MAR. Esses procedimentos são:

1. Cálculo de Informações de Avaliação de Regras - `evaluateAllSetOfRuleFrequency/1`
2. Cálculo de Medidas de Avaliação de Regras - `calculateMeasuresOverAllSetOFRules/1`
3. Cálculo do Grau de Surpresa de Pequenos Disjuntos - `smallDisjSurp/3`
4. Cálculo do Grau de Surpresa dos Atributos Individuais Discretos da Regra - `indAttSurp/2`

A instanciação[7] de cada um dos i argumentos de um procedimento de aridade i é descrita utilizando a notação padrão para documentar programas Prolog, ou seja,

+ <**arg-i**> o argumento deve estar instanciado;

- <**arg-i**> o argumento deve estar desinstanciado, ou seja, é uma variável livre;

? <**arg-i**> o argumento pode estar instanciado ou desinstanciado.

Os procedimentos implementados no MAR possibilitam a análise de regras sob os aspectos qualitativo e quantitativo. Esses procedimentos são descritos nas próximas seções.

5.1 Análise Qualitativa de Regras

Sob o aspecto qualitativo, regras são avaliadas com o objetivo de saber quais são aquelas melhor sustentadas pelos dados (Lavrac et al., 1999). No MAR, a análise qualitativa é efetuada por meio de procedimentos que implementam algumas medidas de avaliação de regras.

[7] Na realidade, a instanciação dos argumentos dos procedimentos aqui descrita refere-se à instanciação na execução padrão realizada por um usuário do sistema. Entretanto, para o implementador do sistema, a instanciação do tipo `+` pode, *às vezes*, ser substituída por uma instanciação do tipo `?`, e a do tipo `-` pode *sempre* ser substituída por uma instanciação do tipo `?`. Esta característica de Prolog permite verificar resultados específicos além de auxiliar ao implementador a realizar rapidamente a depuração dos procedimentos.

5.1.1 Medidas de Avaliação de Regras

Conforme mencionado na Seção 2.1, as regras R_u tratadas nesse trabalho são do tipo *Body* $\rightarrow$ *Head*, resumidamente $B \rightarrow H$, onde B é o corpo da regra e H é a cabeça da regra. Dado um exemplo $\mathbf{x}_i$ e a classe y_i a qual ele pertence, a avaliação da regra é feita através de um procedimento que determina se B é verdadeiro ou falso e se H é verdadeiro ou falso para esse exemplo. Por meio desse procedimento é possível determinar, para cada regra R_u de um classificador simbólico $\mathbf{h} = \{R_1, ..., R_{NR}\}$, as seguintes informações:

hb número de exemplos para os quais H é verdade e B é verdade e
$\overline{h}b$ número de exemplos para os quais H é falso e B é verdade e
$\overline{hb}$ número de exemplos para os quais H é falso e B é falso e
$h\overline{b}$ número de exemplos para os quais H é verdade e B é falso.

No cálculo dessas informações deve ser considerado a forma que as regras do classificador simbólico $\mathbf{h}$ foram induzidas pelo algoritmo de AM. O Módulo de Análise de Regras calcula essas informações para regras *unordered*, *ordered* e *interclass*, conforme descrito a seguir.

Unordered

Para um conjunto de regras *unordered*, essas informações são calculadas verificando a cobertura de cada regra do classificador simbólico $\mathbf{h} = \{R_1, ..., R_{NR}\}$ em todo conjunto de exemplos S. O Algoritmo 1 mostra como é realizado o cálculo dessas informações para um conjunto de regras *unordered*.

Algoritmo 1 Cálculo de Informações para Avaliação de Regras *Unordered*

Require: $\mathbf{h} = \{R_1, ..., R_{NR}\}$: Conjunto de Regras *Unordered*
$S = \{(\mathbf{x}_1,y_1),...,(\mathbf{x}_N,y_N)\}$: Conjunto de Exemplos
1: $L := [0, 0, 0, 0]$;
2: **for all** $R_u \in \mathbf{h}$ **do**
3: **for all** $(\mathbf{x}_i,y_i) \in S$ **do**
4: j:= cobertura$(R_u, (\mathbf{x}_i,y_i))$;
5: Incrementar em 1 o elemento j da lista L;
6: **end for**
7: **end for**
8: **return** L

A função cobertura$(R_u, (\mathbf{x}_i,y_i))$ retorna um número inteiro no intervalo $1 \cdot \cdot 4$ que é usado como índice para realizar o elemento correspondente da lista L. O índice 1 corresponde a hb, 2 a $\overline{h}b$, 3 a $\overline{hb}$, 4 a $h\overline{b}$.

Ordered

No caso de regras *ordered* existe um **else** implícito entre cada regra, assim o exemplo deve ser aplicado a partir da primeira regra, até que uma delas cubra esse exemplo (B é verdade para o exemplo). Esse exemplo deve ser coberto apenas por essa regra. Para as regras seguintes àquela que cobre o exemplo, as informações devem ser atualizadas incrementando-se os valores de $\overline{b}h$ e $\overline{bh}$ conforme a classe C_v do exemplo. O Algoritmo 2 mostra como é realizado o cálculo dessas informações para um conjunto de regras *ordered*.

Algoritmo 2 Cálculo de Informações para Avaliação de Regras *Ordered*

Require: $\mathbf{h} = \{R_1, ..., R_{NR}\}$: Conjunto de Regras *Ordered*
$S = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\}$: Conjunto de Exemplos

```
L := [0, 0, 0, 0];
first := false;
for all R_u ∈ h do
  for all (x_i, y_i) ∈ S do
    while first = false do
      if j = 1 then
        j := cobertura(R_u, (x_i, y_i));
        Incrementar em 1 o elemento j da lista L;
      end if
      if j = 1 or j = 2 then
        first := true;
      end if
    end while
    j := cobertura(R_u, (x_i, y_i)); {o exemplo já foi coberto por uma regra }
    if j = 3 or j = 4 then
      Incrementar em 1 o elemento j da lista L;
    end if
  end for
end for
return L
```

$$\mathbf{h}'_1 \qquad \begin{cases} R_1 : Corpo \text{ then CLASS } = \alpha \\ R_2 : Corpo \text{ then CLASS } = \alpha \end{cases}$$

else

$$\mathbf{h}'_2 \qquad \begin{cases} R_3 : Corpo \text{ then CLASS } = \beta \\ R_4 : Corpo \text{ then CLASS } = \beta \\ R_5 : Corpo \text{ then CLASS } = \beta \end{cases}$$

else

$\vdots$

$$\mathbf{h}'_{NCl} \qquad \begin{cases} R_{u-2} : Corpo \text{ then CLASS } = \gamma \\ R_{u-1} : Corpo \text{ then CLASS } = \gamma \end{cases}$$

com $\alpha, \beta, \gamma \in C = C_1, ..., C_{NCl}$.

Figura 7: Regras *Interclass* — Conjunto de Subconjuntos de Regras

Interclass

As regras *interclass* são separadas em C_v blocos para cada uma das $C_v, v = 1, ..., NCl$ diferentes classes, existindo um **else** implícito entre cada um desses blocos. Para regras em um mesmo bloco C_v, as informações são calculadas como descrito anteriormente para regras *unordered*. Ao ser encontrada uma regra que cubra o exemplo em um bloco C_v, as informações extraídas das regras subseqüentes devem ser atualizadas como descrito anteriormente para regras *ordered*, nos C_{NCl-v} blocos seguintes.

Por exemplo, a Figura 7 mostra que, para cada classe, existe um conjunto de subconjuntos de regras $\mathbf{h}' = \{\mathbf{h}'_1, ..., \mathbf{h}'_{NCl}\}$[8]. É importante observar que, além do **else** implícito entre cada um desses subconjuntos de regras, esses subconjuntos são formados conforme a ordem em que o algoritmo de AM induz as classes que compõem a hipótese $\mathbf{h}$. Cada elemento de $\mathbf{h}'$, ou seja, $\mathbf{h}'_1, ..., \mathbf{h}'_{NCl}$, é uma hipótese com as regras que predizem a mesma classe.

Assim, o exemplo deve ser aplicado a todas as regras de $\mathbf{h}'_1$ e as informações são calculadas como descrito anteriormente para regras *unordered*. Se nenhuma regra de $\mathbf{h}'_1$ cobrir o exemplo, ele deve ser aplicado a $\mathbf{h}'_2$ e assim por diante. Ao ser encontrada uma regra em um $\mathbf{h}'_v$ que cobre o exemplo, as informações extraídas das regras subseqüentes são atualizadas como descrito anteriormente para regras *ordered*, nos $\mathbf{h}'_{NCl-v}$ subconjuntos seguintes.

Após determinar as informações para cada uma das regras R_u[9] de um classificador simbólico $\mathbf{h}$, os valores encontrados podem ser expressos sob a forma de freqüências relativas, as quais são utilizadas nesse trabalho como uma estimativa de probabilidade. Por exemplo, considerando a

[8] $\mathbf{h}' = \mathbf{h}$ - *default*. A regra *default* é uma regra especial cujo corpo é vazio e a cabeça é a classe de maior freqüência no conjunto de exemplos. Assim, uma hipótese $\mathbf{h}$ sempre irá classificar um exemplo devido à existência da regra *default*.

[9] Se R_u é a regra *default*, então as informações não são calculadas para essa $R_u \in \mathbf{h}$.

informação hb, a probabilidade $P(HB)$ pode ser determinada da seguinte forma:

$$P(HB) = f_{bh} = \frac{hb}{N}$$

onde N é o número total de exemplos utilizados para calcular a informação hb.

De forma semelhante podem ser determinados os valores das probabilidades $P(\overline{H}B), P(\overline{HB})$ e $P(H\overline{B})$. Conhecidas essas probabilidades, os valores de $P(B)$, $P(\overline{B})$, $P(H)$ e $P(\overline{H})$ podem ser determinados. Por exemplo:

$$P(B) = P(\overline{H}B) + P(HB) \;\; ou \;\; P(\overline{B}) = P(H\overline{B}) + P(\overline{HB})$$

Neste trabalho, essas probabilidades são utilizadas para calcular as medidas de avaliação de regras.

No *framework* proposto por Lavrac em (Lavrac et al., 1999), algumas medidas de avaliação encontradas na literatura foram unificadas, considerando uma regra R_u na forma $B \rightarrow H$. Como já mencionado, essas medidas podem ser divididas nas seguintes categorias:

1. Medidas genéricas de avaliação;

Precisão	$Acc(B \rightarrow H) = P(H\|B) = \frac{hb}{b}$
Erro	$Err(B \rightarrow H) = P(\overline{H}\|B) = \frac{\overline{h}b}{b}$
Confiança negativa	$NegRel(B \rightarrow H) = P(\overline{H}\|\overline{B}) = \frac{\overline{hb}}{\overline{b}}$
Sensitividade	$Sens(B \rightarrow H) = P(B\|H) = \frac{hb}{h}$
Especificidade	$Spec(B \rightarrow H) = P(\overline{B}\|\overline{H}) = \frac{\overline{hb}}{\overline{h}}$
Cobertura	$Cov(B \rightarrow H) = P(B) = \frac{b}{N}$
Suporte	$Sup(B \rightarrow H) = P(HB) = \frac{hb}{N}$
Novidade	$Nov(B \rightarrow H) = P(HB) - P(H)P(B) = \frac{1}{N}(hb - \frac{h \cdot b}{N})$
Satisfação	$Sat(B \rightarrow H) = \frac{(P(\overline{H}) - P(\overline{H}\|B))}{P(\overline{H})} = \frac{\frac{\overline{h}}{N} - \frac{\overline{h}b}{b}}{\frac{\overline{h}}{N}} = 1 - \left(\frac{\overline{h}b}{b} \cdot \frac{N}{\overline{h}}\right)$

2. Medidas relativas de avaliação;

Precisão relativa	$RAcc(B \rightarrow H) = P(H\|B) - P(H) = \frac{hb}{b} - \frac{h}{N}$
Confiança negativa relativa	$RNegRel(B \rightarrow H) = P(\overline{H}\|\overline{B}) - P(\overline{H}) = \frac{\overline{hb}}{\overline{b}} - \frac{\overline{h}}{N}$
Sensitividade relativa	$RSens(B \rightarrow H) = P(B\|H) - P(B) = \frac{hb}{h} - \frac{b}{N}$
Especificidade relativa	$RSpec(B \rightarrow H) = P(\overline{B}\|\overline{H}) - P(\overline{B})\ \frac{\overline{hb}}{\overline{h}} - \frac{\overline{b}}{N}$

3. Medidas relativas de avaliação com peso;

Precisão relativa com peso	$WRAcc(B \rightarrow H) = P(B)(P(H\|B) - P(H)) = \frac{b}{N}\left(\frac{hb}{b} - \frac{h}{N}\right)$
Confiança negativa relativa com peso	$WRNegRel(B \rightarrow H) = P(\overline{B})(P(\overline{H}\|\overline{B}) - P(\overline{H})) = \frac{\overline{b}}{N}\left(\frac{\overline{hb}}{\overline{b}} - \frac{\overline{h}}{N}\right)$
Sensitividade relativa com peso	$WRSens(B \rightarrow H) = P(H)(P(B\|H) - P(B)) = \frac{h}{N}\left(\frac{hb}{h} - \frac{b}{N}\right)$
Especificidade relativa com peso	$WRSpec(B \rightarrow H) = P(\overline{H})(P(\overline{B}\|\overline{H}) - P(\overline{B})) = \frac{\overline{h}}{N}\left(\frac{\overline{hb}}{\overline{h}} - \frac{\overline{b}}{N}\right)$

Baseada na medida de novidade, Lavrac mostra que as medidas relativas de avaliação com peso são iguais entre si e iguais a medida de novidade da regra, i.e.

$$WRAcc(B \rightarrow H) = WRSens(B \rightarrow H) = WRSpec(B \rightarrow H) = WRNegRel(B \rightarrow H) = Nov(B \rightarrow H)$$

Maiores detalhes a respeito da abordagem utilizada no trabalho de Lavrac para unificação dessas medidas de avaliação de regras podem ser encontrados em (Gomes, 2002).

Geralmente, medidas de avaliação são úteis para determinar quais são as "melhores" regras de um classificador simbólico **h**. As "melhores" regras podem ser aquelas que apresentam o maior (ou menor) valor para uma determinada medida de avaliação. Por exemplo, pode ser considerado que as "melhores" regras R_u de um classificador simbólico **h**, são aquelas que possuem maior valor para a medida de Precisão (ou menor valor para a medida de Erro).

Os procedimentos do MAR que implementam avaliação de regras são descritos a seguir.

5.1.2 Procedimentos que Implementam Medidas de Avaliação de Regras

Como já mencionado, os procedimentos do MAR possibilitam a análise qualitativa de regras, implementando o cálculo das informações a serem usadas para determinar as medidas de avaliação de regras. Esses procedimentos são:

1. Cálculo de Informações de Regras - `evaluateAllSetOfRuleFrequency/1`.

2. Cálculo de Medidas de Avaliação de Regras - `calculateMeasuresOverAllSetOFRules/1`.

O procedimento `evaluateAllSetOfRuleFrequency/1` implementa o cálculo de informações para as três formas de indução de regras, conforme mencionado anteriormente. Na Figura 2, página 6, esse procedimento está representado na linha t_2. O argumento desse procedimento é:

+ <**arg-1**> **yes** para avaliar todos os conjuntos de regras, desconsiderando se foram anteriormente avaliados;
  no para avaliar somente os conjuntos de regras ainda não avaliados.

Dado um conjunto de classificadores simbólicos $\mathcal{H} = \{\mathbf{h}_1,...,\mathbf{h}_L\}$ e um conjunto de exemplos $S = \{(\mathbf{x}_1,y_1),...,(\mathbf{x}_N,y_N)\}$, o procedimento `evaluateAllSetOfRuleFrequency/1` implementa o cálculo das informações para cada classificador simbólico $\mathbf{h} \in \mathcal{H}$, aplicando todos os exemplos de S em cada regra R_u de $\mathbf{h}$. As informações são calculadas levando em conta, separadamente, exemplos com valores conhecidos e exemplos com valores desconhecidos. Essas informações são expressas sob a forma de freqüências relativas, i.e. f_{hb}, $f_{\overline{h}b}$, $f_{\overline{hb}}$, $f_{h\overline{b}}$.

O procedimento `evaluateAllSetOfRuleFrequency/1` então adiciona as informações expressas como freqüências relativas ao final de cada regra R_u de cada hipótese $\mathbf{h} \in \mathcal{H}$ e grava cada regra atualizada na Base de Fatos.

No Algoritmo 3 é descrito o cálculo das informações implementado no procedimento `evaluateAllSetOfRuleFrequency/1`.

Algoritmo 3 Cálculo das Informações para um Conjunto de Hipóteses $\mathcal{H}$

Require: $\mathcal{H} = \mathbf{h}_1,...,\mathbf{h}_L$: Conjunto de Hipóteses
$S = \{(\mathbf{x}_1,y_1),...,(\mathbf{x}_N,y_N)\}$: Conjunto de Exemplos

```
for all h ∈ H do
    Determine a forma de avaliar as regras de h, i. e. unordered, ordered ou interclass;
    for all R_u ∈ h do
        for all (x_i,y_i) ∈ S do
            Calcule as informações de R_u;
        end for
        Atualize R_u com essas informações;
    end for
end for
```

Dado um conjunto de classificadores simbólicos $\mathcal{H} = \{\mathbf{h}_1,...,\mathbf{h}_L\}$ e um conjunto de exemplos $S = \{(\mathbf{x}_1,y_1),...,(\mathbf{x}_N,y_N)\}$, o procedimento `calculateMeasuresOverAllSetOFRules/1` implementa o cálculo das medidas de avaliação de regra, utilizando as informações calculadas pelo procedimento `evaluateAllSetOfRuleFrequency/1`.

Esse procedimento está ilustrado na Figura 2, página 6, na linha t_3. O argumento desse procedimento é:

+ <**arg-1**> **'KNOW'** calcula as medidas de avaliação de cada regra considerando apenas as informações referentes aos exemplos com valores conhecidos;

'UNKNOW' calcula as medidas de avaliação da regra considerando apenas as informações referentes aos exemplos com valores desconhecidos.

'ALL' calcula as medidas de avaliação da regra somando as informações referentes aos exemplos com valores conhecidos e desconhecidos.

As medidas de avaliação de cada regra são gravadas na Base de Fatos e podem ser referenciadas segundo os nomes na Tabela 6.

Medidas genéricas	`accR`:	Precisão da regra
	`errR`:	Erro da regra
	`negrelR`:	Confiança negativa da regra
	`sensR`:	Sensitividade da regra
	`specR`:	Especificidade da regra
	`covR`:	Cobertura da regra
	`supR`:	Suporte da regra
	`novR`:	Medida que mostra o quanto uma regra é nova, interessante ou fora do comum
	`satR`:	Satisfação da regra
Medidas relativas	`raccR`:	Precisão relativa da regra
	`rnegrelR`:	Confiança negativa relativa da regra
	`rsensR`:	Sensitividade relativa da regra
	`rspecR`:	Especificidade relativa da regra
Medidas relativas com peso	`wraccR`:	Precisão relativa com peso da regra
	`wrnegrelR`:	Confiança negativa relativa com peso da regra
	`wrsensR`:	Sensitividade relativa com peso da regra
	`wrspecR`:	Especificidade relativa com peso da regra

Tabela 6: Identificação das Medidas de Avaliação da Regra

Ainda que todas as medidas relativas com peso sejam iguais entre si e iguais a medida genérica de novidade (seção 5.1.1, página 16), foi decidido manter a referência a essa medida utilizando os vários nomes freqüentemente apresentados na literatura para denominá-la.

5.2 Análise Quantitativa de Regras

Sob o aspecto quantitativo regras são avaliadas com o intuito de selecionar aquelas que possam trazer algum conhecimento surpreendente ou inesperado para o usuário, ou seja, o objetivo dessas regras é selecionar as regras mais interessantes (Silberschatz and Tuzhilin, 1996). No Módulo de Análise de Regras a análise qualitativa é implementada por meio do cálculo de medidas de interessabilidade de regras.

5.2.1 Medidas de Interessabilidade de Regras

Várias medidas de interessabilidade de regra podem ser encontradas na literatura (Freitas, 1998b; Freitas, 1998a; Hilderman and Hamilton, 1999). Essas medidas podem ser:

1. **Subjetivas ou orientadas ao usuário**: A interessabilidade está estritamente relacionada com as impressões gerais do usuário sobre uma regra e em geral depende do domínio de aplicação.

2. **Objetivas ou orientadas a dados**: A interessabilidade está estritamente relacionada com os dados da regra, assim, ela é independente do domínio de aplicação e pode ser computada.

Nos trabalhos de Freitas (Freitas, 1998b; Freitas, 1998a) são propostas duas medidas objetivas de interessabilidade. Essas medidas são:

1. Surpresa de Pequenos Disjuntos e

2. Surpresa dos Atributos Individuais da Regra.

as quais foram inicialmente estudas e implementadas por Horst em (Horst, 1999).

A idéia geral da medida de Surpresa de Pequenos Disjuntos considera um *pequeno disjunto* como conhecimento surpresa quando esse disjunto prediz uma classe diferente das classes preditas pelas suas generalizações mínimas GM. Saber se uma regra é um pequeno disjunto depende de algum critério específico relacionado com os exemplos, tal como um *threshold* ou outro critério mais flexível. Geralmente, o *threshold* adotado é o número de exemplos cobertos pela regra.

Um pequeno disjunto terá tantas generalizações mínimas quantas forem as suas condições, isto é, cada uma das g generalizações ($g = 1...m$) está associada com uma das m condições do disjunto original. Uma maneira de obter as generalizações mínimas de um disjunto é através da remoção de uma condição do disjunto original. A classe y_i mais freqüente nos exemplos cobertos por uma generalização mínima é a classe predita por essa generalização mínima. O número de vezes que as classes y_i das generalizações mínimas diferem da classe C_v do pequeno disjunto é o valor do grau de surpresa do disjunto. Essa medida de interessabilidade da regra pode ser calculada através da fórmula:

$$SurpDisj(R) = \sum_{g=1}^{m} diferente(y_i, C_v) \tag{1}$$

onde

$$diferente = \begin{cases} 1 & \text{se } y_i \neq C_v \\ 0 & \text{caso contrário} \end{cases}$$

O valor do grau de surpresa pode ser normalizado pelo número de condições m presentes no disjunto. Essa normalização procura evitar a confusão entre medidas de complexidade sintática e a medida de surpresa de disjuntos.

$$SurpDisj_{norm}(R) = \frac{SurpDisj(R)}{m} \tag{2}$$

A segunda medida proposta, Surpresa dos Atributos Individuais da Regra, no caso de atributos discretos, é determinada através do poder de predição dos atributos individuais discretos presentes no corpo de uma regra R_u, ou seja, por meio dos atributos que tenham alto ganho de informação. Quanto mais atributos com alto ganho de informação estiverem presentes no corpo de uma regra,

mais interessante a regra será. Essa medida de surpresa da regra pode ser calculada através da fórmula:

$$SurpAtr(R) = \frac{1}{\sum_{p=1}^{m} \frac{GInfo(At_p)}{m}} \tag{3}$$

na qual

$$GInfo(At_p) = Info(C) - Info(C|At_p) \tag{4}$$

com

$$Info(C) = -\sum_{v=1}^{NCl} \Pr(C_v) \log \Pr(C_v) \tag{5}$$

$$Info(C|At_p) = \sum_{q=1}^{NClp} \Pr(At_{pq})(-\sum_{v=1}^{NCl} \Pr(C_v|At_{pq}) \log \Pr(C_v|At_{pq})) \tag{6}$$

onde:

$Info(C)$ é a informação da classe;
$Info(C|At_p)$ é a informação do atributo classe C dado o atributo At_p;
At_{pq} denota o q-ésimo valor do atributo At_p;
C_v denota o v-ésimo valor da classe C;
NCl é o número de classes;
$NClp$ representa o número de valores possíveis do atributo At_p;
todos os log estão na base 2.

As probabilidades $\Pr(C_v)$, $\Pr(At_{pq})$ e $\Pr(C_v|At_{pq})$ podem ser obtidas aproximando-as das respectivas freqüências relativas.

As medidas de interessabilidade aqui apresentadas são calculadas de forma independente do domínio de aplicação e de impressões que o usuário tem a respeito da regra. Num processo de análise quantitativa, quanto mais alto o valor de uma dessas medidas para uma regra R_u, maior é a chance de que o conhecimento descrito por R_u seja um conhecimento interessante ou inesperado.

5.2.2 Procedimentos que Implementam Medidas de Interessabilidade de Regras

Os procedimentos do MAR que possibilitam a análise quantitativa de regras, implementando medidas objetivas de interessabilidade, são:

1. Cálculo do Grau de Surpresa de Pequenos Disjuntos - `smallDisjSurp/3`
2. Cálculo do Grau de Surpresa dos Atributos Individuais da Regra - `indAttSurp/2`

O procedimento `smallDisjSurp/3` está ilustrado na Figura 2, página 6, na linha t_4. Considerando uma hipótese **h**, esse procedimento encontra todas as regras $R_u \in \mathbf{h}$ cujo número de exemplos cobertos é menor ou igual ao valor Vt (pequenos disjuntos). Encontrados os pequenos disjuntos, o procedimento efetua os cálculos segundo a Equação 1 na página 20. Como resultado, esse procedimento retorna uma lista contendo valores da medida de Surpresa dos Pequenos Disjuntos, inclusive já normalizados, para cada regra $R_u \in \mathbf{h}$. Os argumentos do procedimento `smallDisjSurp/3` são:

+ <**arg-1**> identificador de uma hipótese;

+ <**arg-2**> valor do *threshold*;

− <**arg-3**> lista dos valores calculados do grau de surpresa dos pequenos disjuntos.

No Algoritmo 4 é descrito o cálculo do grau de Surpresa de Pequenos Disjuntos ($SurpDisj$ e $SurpDisjNorm$) das regras de uma hipótese **h**, dado um valor Vt para o *treshold* padrão.

Algoritmo 4 Cálculo do Grau de Surpresa de Pequenos Disjuntos

Require: $\mathbf{h} = \{R_1, ..., R_{NR}\}$: Conjunto de Regras
$S = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\}$: Conjunto de Exemplos
Vt: Valor do *threshold*
1: **for all** $R_u \in \mathbf{h}$ **do**
2: **for all** $(\mathbf{x}_i, y_i) \in S$ **do**
3: **if** número de exemplos cobertos por $R_u \leq Vt$ **then**
4: **for all** generalização mínima GM de R_u **do**
5: **if** C_v da $GM \neq y_i$ **then**
6: Some 1 ao grau de surpresa;
7: **end if**
8: **end for**
9: $SurpDisj$ = grau de Surpresa do Pequeno Disjunto ;
10: **end if**
11: **end for**
12: $SurpDisjNorm$ = grau de Surpresa do Pequeno Disjunto Normalizado;
13: **end for**
14: **return** $SurpDisj$ e $SurpDisjNorm$

Finalmente, o cálculo do grau de Surpresa dos Atributos Individuais (discretos) da Regra é implementado pelo procedimento `indAttSurp/2`, ilustrado na Figura 2, página6, na linha t_2.

Considerando uma hipótese **h**, o procedimento efetua os cálculos segundo a Equação 3 na página precedente, gerando como resultado uma lista com os valores com o grau de Surpresa dos Atributos Individuais (discretos) de cada regra $R_u \in \mathbf{h}$.

6 Exemplo de Execução

Um exemplo de execução do Módulo de Análise de Regras do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$ é apresentado utilizando um conjunto de exemplos muito simples mostrado na Figura 8, e o conjunto de regras induzido pelo algoritmo de AM $\mathcal{C}4.5rules$ (Quinlan, 1988) ilustrado na Figura 9. Ambos conjuntos de exemplos e regras induzidas por $\mathcal{C}4.5rules$, encontram-se na sintaxe padrão Prolog do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$ (Gomes et al., 2002). Deve ser observado que o $\mathcal{C}4.5rules$ induz regras *interclass*.

O conjunto de exemplos refere-se à medições diárias de condições meteorológicas, nas quais cada dia (exemplo) foi rotulado como "go" se as condições meteorológicas daquele dia são favoráveis para uma viagem à fazenda ou "dont_go", caso contrário.

```
ex(0, [sunny,25,79,yes,dont_go]).
ex(1, [sunny,28,?,yes,dont_go]).
ex(2, [sunny,22,70,no,go]).
ex(3, [sunny,23,95,no,dont_go]).
ex(4, [sunny,?,85,no,dont_go]).
ex(5, [sunny,22,90,yes,dont_go]).
ex(6, [overcast,19,90,yes,go]).
ex(7, [overcast,29,82,no,dont_go]).
ex(8, [overcast,19,85,yes,go]).
ex(9, [overcast,26,?,no,go]).
ex(10,[overcast,20,87,yes,go]).
ex(11,[rain,22,95,yes,go]).
ex(12,[rain,19,70,?,dont_go]).
ex(13,[rain,23,80,yes,dont_go]).
ex(14,[rain,25,81,no,go]).
ex(15,[rain,21,80,no,go]).
```

Figura 8: Arquivo de Exemplos

```
inducer(id_3,'c45-inter-class-ordered').
inputFile(id_3,'voyage.c45.inter.ordered_output').
dAte(id_3,day('Thu','Jun',14,2001),time(20:54:28)).
evaluatedAs(id_3,'INTER-CLASS ORDERED').
nameFile(id_3,'voyage.names').
dataFile(id_3,'voyage.data').

rule(id_3,0001,[equal(outlook,rain),equal(windy,no)],class(go),[],[]).
rule(id_3,0002,[equal(outlook,sunny),greater(humidity,78.80)],class(dont_go),[],[]).
rule(id_3,0003,[equal(outlook,rain),equal(windy,yes)],class(dont_go),[],[]).
rule(id_3,0004,[],class(go),[],[]).
```

Figura 9: Arquivo de Regras *Interclass* Induzidas por $\mathcal{C}4.5rules$

Como pode ser observado na Figura 2 na página 6, partindo da linha t_0, os procedimentos `loadRules/0` e `loadExamples/0` irão carregar na Base de Fatos o arquivo de regras e o arquivo de exemplos, respectivamente.

Na linha t_1 o procedimento `verifyBase/0` verifica se as regras e os exemplos na Base de Fatos pertencem ao mesmo domínio. Sendo bem sucedido, os procedimentos `listOfExamples/1` e `dictionary/1` são executados logo em seguida.

Na linha t_2, o procedimento `evaluateAllSetOfRuleFrequency/1` calcula os valores das freqüências relativas para o conjunto de regras armazenado na BF. Na Figura 10 são apresentados os resultados após a execução do procedimento.

```
rule( id_3, 1, [equal(outlook,rain),equal(windy,no)], class(go),
    [0.133333333333333,0,0.466666666666667,0.4,15],[0,1,0,0,1] ).

rule( id_3, 2, [equal(outlook,sunny),greater(humidity,78.8)], class(dont_go),
    [0.285714285714286,0,0.5,0.214285714285714,14],[0.5,0,0.5,0,2] ).

rule( id_3, 3, [equal(outlook,rain),equal(windy,yes)], class(dont_go),
    [6.66666666666667E-002,6.66666666666667E-002,0.466666666666667,0.4,15],
    [0,0,0,1,1] ).

rule( id_3, 4, [], class(go), [], [] ).
```

Figura 10: Regras na Base de Fatos após a Execução do Procedimento `evaluateAllSetOf-RuleFrequency/1`

Ainda na linha t_2, o procedimento `indAttSurp/2` calcula a medida de Interessabilidade dos Atributos Individuais de cada regra. Na Figura 11 são apresentados os valores encontrados dessa medida para cada regra.

```
rule(id_3,1,[equal(outlook,rain),equal(windy,no)],class(go)).
SurpAttR : 6.7527

rule(id_3,2,[equal(outlook,sunny),greater(humidity,78.8)],class(dont_go)).
SurpAttR : 4.4011

rule(id_3,3,[equal(outlook,rain),equal(windy,yes)],class(dont_go)).
SurpAttR : 6.7527

rule(id_3,4,[],class(go)).
SurpAttR : 0.0000
```

Figura 11: Surpresa dos Atributos Individuais das Regras

Na linha t_3 da Figura 2 o procedimento `calculateMeasuresOverAllSetOfRules/1` calcula as medidas qualitativas para cada regra, as quais são gravadas na Base de Fatos. Na Figura 12 são apresentados os valores dessas medidas para cada regra na BF. Esses valores foram calculados a partir da soma das informações referentes aos exemplos com valores conhecidos e desconhecidos. Deve ser observado que os valores foram pré-formatados para exibição no terminal de vídeo.

```
rule(id_3,1,[equal(outlook,rain),equal(windy,no)],class(go)).

Rule Measures =>

accR : 0.6667           errR : 0.3333           negRelR : 0.5385
sensR : 0.2500          specR : 0.8750          covR : 0.1875
supR : 0.1250           novR : 0.0313           satR : -0.1667
raccR : 0.1667          rnegRelR : 0.0385       rsensR : 0.0625
rspecR : 0.0625         wraccR : 0.0313         wrnegRelR : 0.0313
wrsensR : 0.0313        wrspecR : 0.0313

rule(id_3,2,[equal(outlook,sunny),greater(humidity,78.8)],class(dont_go)).

Rule Measures =>

accR : 1.0000           errR : 0.0000           negRelR : 0.7273
sensR : 0.6250          specR : 1.0000          covR : 0.3125
supR : 0.3125           novR : 0.1563           satR : 0.5000
raccR : 0.5000          rnegRelR : 0.2273       rsensR : 0.3125
rspecR : 0.3125         wraccR : 0.1563         wrnegRelR : 0.1563
wrsensR : 0.1563        wrspecR : 0.1563

rule(id_3,3,[equal(outlook,rain),equal(windy,yes)],class(dont_go)).

Rule Measures =>

accR : 0.5000           errR : 0.5000           negRelR : 0.5000
sensR : 0.1250          specR : 0.8750          covR : 0.1250
supR : 0.0625           novR : 0.0000           satR : -0.5000
raccR : 0.0000          rnegRelR : 0.0000       rsensR : 0.0000
rspecR : 0.0000         wraccR : 0.0000         wrnegRelR : 0.0000

rule(id_3,4,[],class(go)).

Rule Not Evaluated, No Measures Calculated ...
```

Figura 12: Medidas Qualitativas das Regras

Na linha t_4 o procedimento `smallDisjSurp/3` calcula o grau de surpresa dos pequenos disjuntos presentes na base. O *treshold* adotado para calcular essa medida é 0.20, ou seja, foram somente medidos os pequenos disjuntos que cobrem até 20% dos exemplos. Na Figura 13 são apresentados os pequenos disjuntos e os valores encontrados para essa medida. Como já mencionado, vários

```
rule(id_3,1,[equal(outlook,rain),equal(windy,no)],class(go)).

SurpDisj : 0.0000
SurpDisjNorm : 0.0000

rule(id_3,3,[equal(outlook,rain),equal(windy,yes)],class(dont_go)).

SurpDisj : 1.0000
SurpDisjNorm : 0.5000
```

Figura 13: Surpresa dos Pequenos Disjuntos das Regras

conjuntos de regras pertencentes a diferentes classificadores simbólicos podem ser armazenados na Base de Fatos. Entretanto, é responsabilidade do usuário executar o procedimento `verifyBase/0` para verificar que esses conjuntos de regras referem-se ao mesmo domínio do conjunto de exemplos (único) armazenado na BF.

7 Considerações Finais

Neste trabalho foi descrito em detalhes o projeto e a implementação do Módulo de Análise de Regras do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$. A arquitetura desse sistema foi apresentada e também foi proposta uma metodologia, denominada PEFD — *Procedures Execution Flow Diagram*, para documentá-lo. A metodologia proposta facilita muito o projeto de uma interface gráfica, que será construída futuramente, além de ser adequada à representação de pré e pós condições dos procedimentos implementados em cada módulo do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$.

A linguagem de programação lógica Prolog, utilizada na implementação do $\mathcal{R}_{ule}\mathcal{S}_{ystem}$, permitiu a rápida prototipagem dos módulos desse sistema. O ambiente de programação LPA Prolog (Westwood, 2000b; Westwood, 2000a), utilizado no desenvolvimento desse trabalho, já incorpora uma série de facilidades para o desenvolvimento de ferramentas computacionais, facilitando a familiarização ao paradigma de programação lógica e também a detecção de erros de programação.

As medidas de avaliação e de interessabilidade implementadas no MAR, tornam o $\mathcal{R}_{ule}\mathcal{S}_{ystem}$ útil para a análise automática de regras de conhecimento. Da mesma forma que o Módulo de Combinação e Explicação (MCE) utiliza as medidas genéricas de avaliação de regras para implementar suas funcionalidades, futuramente, novos módulos poderão utilizar as medidas de regras do MAR para implementar outras funcionalidades voltadas para Aprendizado de Máquina e Data Mining.

Referências

Baranauskas, J. A. (2001). Extração automática de conhecimento por múltiplos indutores. Tese de Doutorado, ICMC-USP.

Baranauskas, J. A. and Batista, G. E. A. P. A. (2000). O projeto DISCOVER: Idéias iniciais. (Comunicação Pessoal).

Batista, G. E. A. P. A. (1997). Um ambiente de avaliação de algoritmos de aprendizado de máquina utilizando exemplos. Dissertação de Mestrado, ICMC-USP.

Batista, G. E. A. P. A. (2000). Pré-processamento de dados em aprendizado de máquina supervisionado. Monografia do Exame de Qualificação de Doutorado, ICMC-USP.

Batista, G. E. A. P. A. (2001). Sintaxe padrão do arquivo de exemplos do projeto DISCOVER. http://www.icmc.sc.usp.br/ gbatista/SintaxePadraoFinal.htm.

Bernardini, F. C. (2001). Combinação de classificadores para melhorar o poder preditivo e descritivo de ensembles. Monografia do Exame de Qualificação de Mestrado, ICMC-USP.

Bernardini, F. C. and Monard, M. C. (2002). Um módulo de explicação de ensembles e combinação de regras. Technical report, ICMC-USP. Em elaboração.

Bratko, I. (1990). *Prolog Programming for Artificial Intelligence*. Addison-Wesley.

Caulkins, C. W. (2000). Aquisição de conhecimento utilizando aprendizado de máquina relacional. Dissertação de Mestrado, ICMC-USP.

Félix, L. C. M. (1998). Data mining no processo de extração de conhecimento de bases de dados. Dissertação de Mestrado, ICMC-USP.

Freitas, A. A. (1998a). A multi-criteria approach for the evaluation of rule interestingness. In *Proceedings of the International Conference on Data Mining*, pages 7–20, Rio de Janeiro, RJ.

Freitas, A. A. (1998b). On objective measures of rule surprisingness. In *Proceedings of the Second European Symp. LNAI*, volume 1510, pages 1–9.

Gomes, A. K. (2001). Análise de regras utilizando medidas de avaliação e de interessabilidade do conhecimento simbólico. Exame de Qualificação de Mestrado, ICMC-USP.

Gomes, A. K. (2002). Análise automática do conhecimento extraído de classificadores simbólicos utilizando medidas de avaliação e de interessabilidade de regras. Dissertação de Mestrado, ICMC-USP (A ser defendida).

Gomes, A. K., Bernardini, F. C., and Monard, M. C. (2002). Uma sintaxe padrão Prolog para classificadores simbólicos. Technical Report 154, ICMC-USP. ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/Rt_154.ps.zip.

Hilderman, R. J. and Hamilton, H. J. (1999). Knowledge discovery and interestingness measures: A survey. Technical Report s4s, University of Regina.

Horst, P. S. (1999). Avaliação do conhecimento adquirido por algoritmos de aprendizado de máquina utilizando exemplos. Dissertação de Mestrado, ICMC-USP.

Imamura, C. Y. (2001). Pré-processamento para extração de conhecimento de bases textuais. Dissertação de Mestrado, ICMC-USP.

Lavrac, N., Flach, P., and Zupan, B. (1999). Rule evaluation measures: a unifying view. In *Proceedings of the Ninth International Workshop on Inductive Logic Programming. LNAI*, volume 1634, pages 74–185.

Lee, H. D. (2000). Seleção e construção de features relevantes para o aprendizado de máquina. Dissertação de Mestrado, ICMC-USP.

Martins, C. A. (2001). Clustering conceitual em aprendizado de máquina. Exame de Qualificação de Doutorado, ICMC-USP.

Milaré, C. R. (2000). Extração de conhecimento de redes neurais. Exame de Qualificação de Doutorado, ICMC-USP.

Nagai, W. A. (2000). Avaliação do conhecimentos extraído de problemas de regressão. Dissertação de Mestrado, ICMC-USP.

Pila, A. D. (2001). Seleção de atributos relevantes para aprendizado de máquina utilizando a abordagem de Rough Sets. Dissertação de Mestrado, ICMC-USP.

Prati, R. C., Baranauskas, J. A., and Monard, M. C. (1999). BibView: Um sistema para auxiliar a manutenção de registros para o BibTeX. Technical Report 95, ICMC-USP. ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/Rt_95.ps.zip.

Prati, R. C., Baranauskas, J. A., and Monard, M. C. (2001a). Extração de informações padronizadas para a avaliação de regras induzidas por algoritmos de aprendizado de máquina simbólico. Technical Report 145, ICMC-USP. `ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/RT_145.ps.zip`.

Prati, R. C., Baranauskas, J. A., and Monard, M. C. (2001b). Uma proposta de unificação da linguagem de representação de conceitos de algoritmos de aprendizado de máquina simbólicos. Technical Report 137, ICMC-USP. `ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/RT_137.ps.zip`.

Pugliesi, J. B. (2001). O pós-processamento em extração de conhecimento de bases de dados. Exame de Qualificação de Doutorado, ICMC-USP.

Quinlan, J. (1988). *C4.5 Programs for Machine Learning*. Morgan Kaufmann Publishers.

Silberschatz, A. and Tuzhilin, A. (1996). What makes patterns interesting in knowledge discovery systems. *IEEE Trans. on Knowledge and Data Eng.*, 8(6):970–974.

Westwood, D. (2000a). LPA-PROLOG for windows programming guide.

Westwood, D. (2000b). LPA-PROLOG technical reference.