Em Busca de Procedimentos de Inferência Eficientes via Relevância por Sintaxe

Joselyto Riani

Orientadora:
Profa. Dra. Renata Wassermann

Dissertação apresentada ao Instituto de Matemática e Estatística da Universidade de São Paulo como parte dos requisitos para obtenção do título de Mestre em Ciências na área de Computação.

Área de Concentração: Inteligência Artificial

São Paulo
Março / 2004

Em Busca de Procedimentos de Inferência Eficientes via Relevância por Sintaxe

Joselyto Riani

Este exemplar corresponde à redação final da dissertação de mestrado devidamente corrigida e defendida por Joselyto Riani e aprovada pela comissão julgadora.

Banca examinadora:

Profa. Dra. Renata Wassermann (Orientadora)
Prof. Dr. Marcelo Finger (IME-USP)
Prof. Dr. Marcos Castilho (UFPR)

São Paulo
Março / 2004

Dedicatória

A minha querida tia Lúcia.

Agradecimentos

Agradeço a Renata pelas longas manhãs e tardes de discussões e esclarecimentos, livros emprestados, depurações realizadas no texto (das mais diversas naturezas) e também pela paciência diante de meus inúmeros imprevistos e improvisos; além dos imensos documentos .pdf enviados para sua caixa postal.

Agradeço a Hyperion por ter concedido espaço em minha agenda de trabalho para que eu pudesse participar das atividades do IME.

Agradeço ao Flávio Ribeiro pelas incontáveis mensagens respondidas e livros emprestados (nos dias e horários mais inusitados: fins de semana, madrugadas, sexta-feira dia 26 de dezembro, etc).

Finalmente, e principalmente, agradeço a sociedade brasileira que, no final das contas, é quem financia a maior parte dos investimentos necessários para que este programa de mestrado possa existir. Espero que este trabalho, direta ou indiretamente, lhe seja útil num futuro (próximo).

Resumo

Riani J. **Em Busca de Procedimentos de Inferência Eficientes via Relevância por Sintaxe**. 2004. 80 f. Dissertação (Mestrado em Ciência da Computação) - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2004.

A solução de alguns problemas, principalmente em Inteligência Artificial, requer a capacidade de inferência, isto é, a capacidade deduzir novas informações a partir de uma base de conhecimento. Infelizmente, esta tarefa não é simples, sendo que quanto mais expressiva for a linguagem de representação da base de conhecimento, mais complexos serão os procedimentos de inferência. Esta dissertação apresenta uma forma de lidar com esta complexidade explorando o conceito de relevância que, em geral, está presente nas bases de conhecimento.

A estratégia proposta é não tentar fazer a inferência utilizando-se todo o conhecimento disponível de uma só vez. Ao invés disto, a inferência deve ser realizada em etapas, começando-se com um pequeno subconjunto das informações da base de conhecimento, acrescentando-se cada vez mais informações à medida que a solução não for encontrada com as informações consideradas na etapa anterior. Neste processo, são selecionadas primeiramente as informações mais relevantes para a inferência desejada.

O ponto chave da estratégia é a escolha da definição de relevância na base de conhecimento, o que poderia disparar uma extensa discussão filosófica sobre o assunto. Entretanto, como é mostrado nesta dissertação, uma forma simples e interessante de se definir a relevância é através da própria sintaxe da linguagem utilizada para representar a base de conhecimento, sem recorrer a nenhuma estrutura extra.

São apresentados resultados práticos da aplicação desta proposta com o provador de teoremas OTTER realizando inferências em uma base de conhecimento construída a partir da biblioteca de problemas TPTP.

Abstract

Riani J. **Towards an Efficient Inference Procedure through Syntax Based Relevance**. 2004. 80 f. Dissertation (Master in Computer Science) - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2004.

The solution of some problems, particularly in Artificial Intelligence, requires the capability of inference, which means being able to deduce new information from a given knowledge base. Unfortunately, this task is far from simple, as the more expressive is the language chosen to represent the knowledge base, the more complex are the inference procedures. This dissertation presents a method to deal with this complexity leveraging a simple notion of relevance which is claimed to exist in the knowledge base.

The proposed strategy suggests that one should not try to make inference taking all available knowledge into account at once. Instead, inference should be done in a step by step fashion, starting with a very small "relevant" subset of the knowledge base. The following steps add more information to the "relevant" subset if the solution can not be found in the previous step. In this process, the most relevant information to the query are selected first.

Note that a key point to this strategy is the chosen definition for relevance in the knowledge base, which is a long discussion among interdisciplinary researchers (philosophers, logicians, linguistics, etc). In spite of the complexity of the subject, it is claimed in this dissertation that one can get useful relevance information looking simply at the syntax of the sentences in the knowledge base (without requiring any other extra structure).

Some practical results of the application of the proposed technique are presented through experiments done with the theorem prover OTTER and the TPTP problem library.

Lista de Tabelas

Lista de Figuras

Sumário

1 Introdução

Muitas aplicações da área de Inteligência Artificial são baseadas em conhecimento que é representado através de um conjunto de fatos e regras chamado de *base de conhecimento*. Em geral, não é possível armazenar explicitamente todo o conhecimento do domínio em questão, sendo portanto necessária uma linguagem de representação associada a um mecanismo capaz de realizar inferências (um mecanismo capaz de derivar novas informações a partir daquelas que foram armazenadas explicitamente).

É importante que a linguagem de representação escolhida seja capaz de expressar com a maior exatidão possível as informações sobre o mundo real. Mas, infelizmente, existe uma relação direta entre a expressividade da linguagem e a complexidade dos processos de inferência. Brachman e Levesque (1985) oferecem uma excelente introdução a este assunto comparando alguns formalismos para representação de conhecimento em termos de poder de expressividade (em comparação com a lógica de primeira ordem) e complexidade computacional do respectivo processo de inferência.

O formalismo mais básico apresentado por Brachman e Levesque (1985) é o uso de bancos de dados relacionais a partir dos quais se consegue extrair informações muito eficientemente mas que possuem um poder de expressividade bem limitado (equivalente à lógica de predicados sem o uso de negações, variáveis, disjunções e quantificadores existenciais). Em seguida, outros formalismos são apresentados (cláusulas *Horn*, rede semântica, etc) sendo que cada um deles permite a utilização de alguma construção a mais da lógica de predicado (conectivos lógicos, sem limites na aridade dos predicados, etc) e, naturalmente, tem o seu processo de inferência cada vez mais complexo.

Sendo assim, faz parte do papel do construtor de um sistema baseado em conhecimento escolher o formalismo mais simples capaz de expressar as informações do domínio em questão. Quando a escolha recai em uma linguagem complexa, vem a necessidade da aplicação de estratégias e heurísticas para lidar com a complexidade inerente ao processo de inferência. Em seu artigo sobre métodos de raciocínio eficiente, Greiner, Darken e Santoso (2001) oferecem uma excelente síntese das principais iniciativas desenvolvidas nesta área.

1.1 Objetivos desta dissertação

Nesta dissertação, propõe-se uma estratégia para lidar com a complexidade dos procedimentos de inferência com base nas idéias apresentadas por Wassermann (2000) em sua tese de doutorado sobre revisão de crenças. Wassermann formaliza um modelo para organização de bases de conhecimento[1] que, além de ser interessante do ponto de vista computacional, é bastante coerente com os modelos cognitivos de memória existentes na literatura sobre o assunto. Um dos principais argumentos apresentados por Wassermann é a observação de que, para resolver seus problemas cotidianos, as pessoas não consideram tudo o que sabem. Ao invés disto, o que parece acontecer, é que elas selecionam conjuntos bem menores com as informações mais relevantes para o problema em questão.

Wassermann propõe que o tamanho do conjunto de informações relevantes seja ditado pela limitação de recursos disponíveis. Seguindo esta linha, é introduzida a idéia de uma estratégia gradativa na qual tenta-se resolver o problema primeiramente com um conjunto mínimo de informações (provavelmente menor que o limite de recursos disponível). Se uma primeira tentativa falha, então outras informações começam a ser consideradas, e assim sucessivamente até que uma solução seja encontrada ou que o limite de recursos disponíveis seja de fato atingido.

Esta é uma estratégia comumente chamada de *quick and dirty*, pois, apesar do forte apelo da intuição, não é provado que existe uma chance maior de se encontrar a solução nas primeiras tentativas. Sendo assim, faz-se necessário verificar na prática se a estratégia produz bons resultados. Para isto, é necessário ter uma linguagem de representação, uma base de conhecimento, um mecanismo de inferência e um conjunto de problemas. A escolha de cada um destes itens foi:

1. **Linguagem de representação:** foi escolhida a lógica de predicados de primeira ordem com igualdade e funções por apresentar expressividade suficiente para representar muitos dos problemas que a área de Inteligência Artificial se propõe a resolver, além de ser complexa o bastante para justificar o uso de heurísticas.
2. **Base de conhecimento:** seria necessário uma base de conhecimento de tamanho razoável com informações reais de algum domínio. Não foi possível obter exatamente esta base pronta. Para suprir esta necessidade, foram geradas bases de conhecimento a partir dos axiomas dos problemas apresentados na biblioteca de problemas TPTP (*Thousands of Problems for Theorem Provers*) descrita por Sutcliffe e Suttner (2004).
3. **Mecanismo de inferência:** foi utilizado o provador de teoremas OTTER (*Organized Techniques for Theorem-proving and Effective Research*) desenvolvido no *Argonne National Laboratory*. McCune (2003) oferece um manual de referência para o OTTER.

[1] Na realidade, Wassermann utiliza a expressão "base de crenças" que é mais comum na literatura da área de revisão de crenças.

4. **Conjunto de problemas**: como o objetivo é testar se a estratégia proposta funciona na prática no caso geral, foi necessário gerar uma extensa massa de testes. Para tal, utilizou-se os próprios problemas da biblioteca TPTP.

1.2 Organização desta dissertação

O Capítulo 2 oferece uma breve introdução à lógica de predicados de primeira ordem que é a linguagem de representação escolhida nos estudos desta dissertação.

O Capítulo 3 descreve o provador de teoremas utilizado nos testes realizados, o OTTER. São discutidos aspectos mais teóricos do que práticos. Isto é, o Capítulo 3 não tem como objetivo descrever como usar o OTTER, para isto veja o manual de referência McCune (2003).

No Capítulo 4 é dada uma formalização da noção de relevância que será utilizada nesta dissertação. É feita uma comparação com a lógica de relacionamento de Epstein (2001) e Krajewski (1986).

O Capítulo 5 descreve a estratégia (heurística) proposta para lidar com a inferência em grandes bases de conhecimento explorando a noção de relevância definida no Capítulo 4. É apresentado um algoritmo baseado no algoritmo de seleção de sentenças relevantes de Wassermann (2000).

Os testes realizados, bem como a análise dos respectivos resultados, são discutidos no Capítulo 6.

O Capítulo 7 conclui esta dissertação fazendo um resumo dos resultados alcançados até o momento e apontando os próximos passos na continuação dos trabalhos.

O apêndice A contém uma documentação técnica das alterações realizadas no código fonte do OTTER para a realização dos testes. São apresentadas as estruturas de dados do OTTER mais relevantes (para o trabalho desta dissertação). O apêndice B apresenta uma listagem dos resultados dos testes realizados.

1.3 Notação

As seguintes regras de notação são utilizadas nesta dissertação:

? Letras do alfabeto latino e grego são utilizadas para representar determinados objetos em estudo (conjuntos, fórmulas, etc). Nestes casos, as ocorrências de letras do alfabeto latino serão grafadas em itálico. Em prol da padronização, o autor reserva determinadas letras para representar determinados conceitos, sendo que quando o conjunto de letras reservadas não for suficiente, serão utilizados números subscritos após as letras (exemplo: a_1).

? Em algumas passagens da dissertação, utiliza-se termos cuja versão em inglês pode facilitar a compreensão do leitor por ser de conhecimento comum dos pesquisadores da área. Nestes casos, o texto da dissertação apresenta a tradução em português do termo entre aspas duplas e a expressão original em inglês é apresentada entre parêntesis com letras grafadas em itálico. O termo em inglês e as aspas na tradução para português são apresentados somente na primeira vez que o termo ocorre no texto.

? Em algumas raras exceções, o autor preferiu definitivamente não traduzir para o português determinadas expressões usuais em inglês. Nestes casos, estas palavras serão sempre grafadas em itálico.

As seguintes abreviaturas são utilizadas nesta dissertação:

1. **fbf**: fórmula bem formada.
2. **LPPO**: lógica de predicados de primeira ordem.
3. **sse**: se e somente se.

2 Lógica de Predicados de Primeira Ordem

Para testar a hipótese desta dissertação, será considerada uma base de conhecimento representada em uma linguagem de predicados de primeira ordem com igualdade e funções. A fim de manter este texto auto-contido, é apresentada neste capítulo uma breve introdução à lógica de predicados de primeira ordem. Maiores detalhes podem ser obtidos em Epstein (2001), Gamut (1991) e Enderton (1972). Epstein oferece uma discussão filosófica, Gamut tem uma abordagem lingüística enquanto Enderton apresenta uma visão matemática.

A lógica de predicados foi introduzida por Frege no final do século XIX e tem sido considerada atualmente como uma das principais formas de representação do conhecimento. O mundo é representado em LPPO através de fatos, objetos, propriedades destes objetos e relações entre os mesmos, o que lhe confere um alto grau de expressividade (compare com as lógicas proposicionais que expressam apenas fatos).

Para descrever um sistema lógico, é necessário apresentar os seguintes aspectos do mesmo:

- **Vocabulário**: conjunto de símbolos que podem aparecer nas expressões da linguagem do sistema lógico.
- **Sintaxe**: conjunto de regras que ditam como os símbolos podem se combinar para formar as expressões válidas da linguagem do sistema lógico.
- **Semântica**: conjunto de definições que estabelece um método para dar significado para as expressões da linguagem.
- **Teoria da prova**: um método que especifica como é possível (e válido) deduzir novas informações a partir de um conjunto de informações já conhecidas.

O conjunto de expressões que pode ser formado com o vocabulário e a sintaxe é chamado de a linguagem do sistema lógico. Nesta dissertação, a linguagem da LPPO considerada será denotada por *L*.

2.1 O vocabulário de *L*

O vocabulário de *L* contém os seguintes símbolos:

- **Constantes lógicas**: são os conectivos lógicos (? , ? , ? , ? e ?), os quantificadores (? e ?) e os símbolos ? e T que representam respectivamente falsidade e verdade.

- ? **Constantes de indivíduo**: representam os indivíduos, objetos ou entidades do mundo real que se está representando. Constantes de indivíduo serão denotadas pelas letras *c* e *d*. O conjunto de constantes de indivíduos da linguagem será denotado por *C*.
- ? **Variáveis**: símbolos utilizados para se fazer asserções sobre os indivíduos sem fazer referência direta aos mesmos. Variáveis serão denotadas pelas letras *w*, *x*, *y*, e *z*. O conjunto de variáveis da linguagem será denotado por *V*.
- ? **Constantes de predicado**: representam as propriedades e relacionamentos existentes entre os indivíduos. A quantidade de indivíduos que o predicado relaciona é chamada de aridade do predicado. A aridade de um predicado pode ser igual a 0, neste caso o predicado é chamado de proposição. Constantes de predicados serão chamadas simplesmente de predicados e serão denotadas pelas letras *p*, *q*, *r* e *s*.
- ? **Símbolo da igualdade**: uma constante de predicado especial de aridade 2 que indica que dois indivíduos são o mesmo. Esta constante de predicado será denotada pelo símbolo ~.
- ? **Constantes de funções**: utilizadas para representar as relações funcionais entre os indivíduos, isto é, relações em que um ou mais indivíduos estão relacionados com exatamente um indivíduo. A quantidade de indivíduos que uma função relaciona é chamada de aridade da função. As funções de aridade igual a 0 são na realidade as constantes de indivíduo. Funções serão denotadas pela letra *f*.
- ? **Símbolos auxiliares**: parênteses e vírgula.

2.2 A sintaxe de *L*

A sintaxe de *L* é definida através de termos e fórmulas bem formadas (fbf). Um termo de *L* é definido recursivamente da seguinte forma:

1. Uma constante de indivíduo é um termo.
2. Uma variável é um termo.
3. Se $t_1,...,t_n$ são termos e *f* é uma função de aridade *n* então $f(t_1,...,t_n)$ é um termo.
4. Somente as expressões formadas com aplicação de um número finito das regras 1, 2 e 3 acima são termos.

Os termos de *L* serão denotados pela letra *t*.

Uma fbf de *L* é definida recursivamente da seguinte forma:

1. Se $t_1,...,t_n$ são termos e *p* é um predicado de aridade *n* então $p(t_1,..,t_n)$ é uma fbf de *L*. Estas fórmulas são chamadas de fórmulas atômicas de *L*.
2. Se ? é uma fbf de *L*, então ? ? é uma fbf de *L*.
3. Se ? e ? são fbfs de *L*, então (? ? ?), (? ? ?), (? ? ?) e (? ? ?) são fbfs de *L*.
4. Se ? é uma fbf de *L* e *x* é uma variável, então (? *x*?) e (?*x*?) são fbfs de *L*.
5. Somente as expressões obtidas a partir de um número finito dos passos 1, 2, 3 e 4 acima são fbfs de *L*.

As fbfs de *L* serão denotadas pelas letras gregas ? , ? , ? e ? .

Uma variável x é dita livre em uma fórmula ? , se a mesma não aparece dentro do escopo de nenhum quantificador em ? . Uma fórmula que não contém variáveis livres é chamada de sentença. Exceto quando expresso o contrário, esta dissertação vai assumir que a base de conhecimento é composta de sentenças (e não fórmulas genéricas).

Uma substituição $[x_1/t_1, ..., x_n/t_n]$ é o processo de gerar uma nova fórmula ? substituindo-se as ocorrências livres da variável x_i numa fórmula ? pelo termo t_i. Substituições serão denotadas pelas letras gregas ? e ?. A fórmula obtida aplicando-se a substituição ? na fórmula ? será denotada por ? ?.

Uma substituição ? = $[x_1/t_1, ..., x_n/t_n]$ também pode gerar um termo t' substituindo as ocorrências da variável x_i num termo t pelo termo t_i. O termo obtido aplicando-se ? em t será denotado por t?.

O unificador de duas fórmulas ? e ? é uma substituição ? tal que ? ? = ? ?. Se não existir tal substituição, diz-se que o mesmo é nulo ou que as fórmulas não unificam.

Um unificador é chamado de "o mais genérico" se ele faz o menor número possível de substituições nas fórmulas.

Uma fórmula ? é uma instância de uma fórmula ? se existe uma substituição ? tal que ? = ? ?.

Se ? é uma fórmula atômica, então ? e ? ? são *literais*. ? é um *literal positivo* enquanto ? ? é um *literal negativo*. Além disto, o par ? e ? ? é chamado de complementar.

Uma *conjunção* é uma fórmula do tipo ? $_1$? ? $_2$? ... ? ? $_n$.

Uma *disjunção* é uma fórmula do tipo ? $_1$? ? $_2$? ... ? ? $_n$.

Uma *cláusula* é uma disjunção de literais. Uma *cláusula positiva* é uma cláusula na qual não ocorrem literais negativos. Uma cláusula unitária é uma cláusula com apenas um literal.

Uma fórmula está na *forma normal conjuntiva* se ela é constituída de uma conjunção de cláusulas.

2.3 Semântica

Para formalizar a semântica de uma LPPO, é necessário introduzir dois conceitos:

- ? A interpretação do mundo real que aponte quais são os indivíduos do mundo aos quais as fórmulas se referenciam bem como as relações existentes entre os mesmos no mundo real.
- ? Funções que permitam associar os termos da linguagem aos indivíduos do mundo real

Uma *interpretação I* para uma LPPO é composta dos seguintes elementos:

? Um conjunto não vazio D chamado de o *domínio* de I. D contém as entidades do mundo real sobre às quais as fórmulas dizem algo. Por exemplo, para expressar alguma informação sobre política no Brasil D deve conter indivíduos como *Brasil*, *Lula*, *FHC*, etc.

? Para cada constante de predicado p de aridade n, uma relação P ? D^n. A relação P indica para quais entidades do mundo real p deve ser considerado verdadeiro. Por exemplo, suponha que a linguagem contenha um predicado *sucessor* de aridade 3 cujo significado é indicar qual presidente é sucessor de qual presidente em um determinado país. Então a interpretação da linguagem deve conter uma relação *SUCESSORES* tal que <*FHC*, *Lula*, *Brasil*> ? *SUCESSORES* (perceba a importância da ordem dos indivíduos na relação). A relação associada a ~ deve conter o par <*d*,*d*> para todo *d* ? *D*.

? Uma função g : C ? D que associa cada constante de indivíduo da linguagem com uma entidade do mundo real.

? Para cada constante de função f de aridade n>0, uma função F : D^n? D. A função F é a interpretação direta de f no mundo real. Por exemplo, suponha que a linguagem contenha uma função *presidente* de aridade 1 cujo significado é indicar o presidente atual de um determinado país. Então a interpretação da linguagem deve conter uma função *PRESIDENTE* tal que *PRESIDENTE*(*Brasil*) = *Lula*.

Seja a' uma *função de atribuição* a' : V ? D que associa uma entidade do mundo real a cada variável da linguagem. Uma função de atribuição a, que associa termos a entidades do mundo real, é definida recursivamente da seguinte forma:

? Se t é uma variável, então $a(t) = a'(t)$.
? Se t é uma constante, então $a(t) = g(t)$.
? Se t=$f(t_1, t_2, ...t_n)$, então $a(t) = F(a(t_1), a(t_2),...,a(t_n))$.

Seja x uma variável e d ? D um indivíduo. A notação $a[x?\ d]$ indica uma função de atribuição que associa x com d e que, para todas as outras variáveis, faz as mesmas associações estabelecidas pela função de atribuição a.

Define-se então uma *função de valoração* $v_{I,a}$ que leva fórmulas de L para o conjunto {0,1} indicando a verdade ou falsidade das fórmulas de L de acordo com a interpretação I e a função de atribuição a da seguinte forma:

? Se ? = $p(t_1, ..., t_n)$ é uma fórmula atômica onde p é uma constante de predicado e t_i são termos, então $v_{I,a}$(?) = 1 sse <$a(t_1)$, ..., $a(t_n)$> ? P.
? $v_{I,a}$(? ?)=1 sse $v_{I,a}$(?)=0.
? $v_{I,a}$(? ? ?)=1 sse $v_{I,a}$(?)=1 e $v_{I,a}$(?)=1.
? $v_{I,a}$(? ? ?)=0 sse $v_{I,a}$(?)=0 e $v_{I,a}$(?)=0.
? $v_{I,a}$(? ? ?)=0 sse $v_{I,a}$(?)=1 e $v_{I,a}$(?)=0.
? $v_{I,a}$(? ? ?)=0 sse $v_{I,a}$(?) ? $v_{I,a}$(?).
? $v_{I,a}$(? x?)=0 sse existe algum d ? D tal que $v_{I,a[x?\ d]}$(?) = 0.
? $v_{I,a}$(?x?)=1 sse existe algum d ? D tal que $v_{I,a[x?\ d]}$(?) = 1.
? $v_{I,a}$(?)=0.
? $v_{I,a}$(T)=1.

A semântica definida da forma acima baseia-se no princípio da composicionalidade que diz que o significado de expressões compostas deve depender exclusivamente do significado das partes que a compõem.

Uma fórmula ? é uma tautologia sse $v_{I,a}(?)=1$ para toda interpretação I e função de atribuição a.

Se ? é uma sentença e $v_{I,a}(?)=1$ então $v_{I,b}(?)=1$ para toda função de atribuição b (é fácil verificar isto já que uma sentença não possui variáveis livres).

Se $v_{I,a}(?)=1$ para toda função de atribuição a, então a interpretação I é chamada de *modelo* da fórmula ? .

Uma interpretação I é um modelo de um conjunto de sentenças K sse I é modelo de cada uma das sentenças de K.

Um conjunto de sentenças K implica logicamente uma sentença ? sse todo modelo de K também é modelo ? . Também é usual dizer neste caso que ? é consequência lógica de K e utilizar a notação $K \models ?$.

2.4 Teoria da Prova

A teoria da prova estabelece métodos para se provar que uma sentença ? é consequência lógica de um conjunto de sentenças K. Estes métodos são chamados de métodos ou procedimentos de inferência. Se m é um método de inferência, então utiliza-se a notação $K \vdash_m ?$ para indicar que a sentença ? pode ser provada (inferida ou deduzida) a partir do conjunto de sentenças K utilizando o método m. Quando está claro a que método se está referindo, é comum suprimir o subscrito m.

Duas características são desejáveis em um método de inferência: correção e completude. Um método de inferência m é correto se $K \vdash_m ?$ implica $K \models ?$, isto é, se a fórmula ? é inferida a partir do conjunto de fórmulas K através do método m, então ? é consequência lógica de K.

Um método de inferência m é completo se todas as conseqüências lógicas de qualquer conjunto K podem ser deduzidas através de m. Isto é, se $K \models ?$ então $K \vdash_m ?$.

Um método trivial para saber se $K \models ?$ seria verificar se todo modelo de K também é um modelo de ? . Entretanto, isto é inviável do ponto de vista computacional (já que podem haver infinitos modelos de K). Sendo assim faz-se necessário desenvolver métodos que permitam realizar inferências independentemente da interpretação que se dá para as fórmulas. Isto pode ser feito através da utilização de axiomas lógicos e de regras de inferência.

Axiomas lógicos são fórmulas verdadeiras por definição enquanto que as regras de inferência especificam como deduzir uma fórmula a partir de um conjunto de outras fórmulas. Uma regra de inferência é dita ser correta se a partir dela se deduz apenas fórmulas que são consequência lógica do conjunto de fórmulas considerado na aplicação da regra. Um exemplo clássico de uma regra de inferência correta é a regra *Modus Ponens* mostrada abaixo:

Modus Ponens: $\frac{? \quad ? ? ?}{?}$. Esta é uma das notações mais adotadas para regras de inferência.

Esta regra diz que, para quaisquer fórmulas ? e ?, a partir de um conjunto de fórmulas que contém {? , ? ? ? } pode-se inferir a fórmula ? . Usualmente, esta regra é lida da como "de ? e ? ? ? infira ? ".

Seja uma teoria da prova formada por um conjunto de axiomas ? e um conjunto de regras de inferência ? . Seja K um conjunto de fórmulas e ? uma fórmula. Uma prova de ? a partir de K nesta teoria da prova é uma sequência finita de fórmulas $<?_1, ..., ?_n>$ tal que $?_n = ?$ e cada $?_i$ é tal que:

- ? $?_i ? K$ ou
- ? $?_i ? ?$ ou
- ? $?_i$ é o resultado da aplicação de alguma regra de inferência ? ? ? aplicada em um conjunto de sentenças $?_j$ tal que $j < i$.

Enderton (1972) demonstra que um sistema lógico correto e completo pode ser obtido utilizando a regra Modus Ponens e o conjunto de axiomas lógicos abaixo:

- ? todas as tautologias da lógica proposicional clássica,
- ? $? x ? ? ? [x/t]$ para toda fórmula ? , variável x e termo t tal que x não ocorre em t,
- ? $? x(? ? ?) ? (? x ? ? ? x ?)$ para toda fórmula ? e ? e toda variável x,
- ? $? ? ? x ?$, para toda fórmula ? e variável x tal que não há uma ocorrência livre de x em ? ,
- ? $x \sim x$ para toda variável x (axioma da reflexibilidade),
- ? $x \sim y ? (? ? ?)$ onde x e y são variáveis e ? é uma fórmula obtida substituindo-se uma ou mais ocorrências de x por y na fórmula ? .

Apesar de completo e correto, este sistema não possui uma implementação direta em um programa de computador. Robinson (1965) apresenta um sistema correto e completo (para lógicas de primeira ordem sem igualdade) orientado para uma implementação em um computador. Robinson formulou um sistema que trabalha apenas com cláusulas, o que não é uma restrição de expressividade já que toda fórmula da LPPO pode ser rescrita como um conjunto finito de cláusulas.[2] O sistema de Robinson utiliza uma única regra de inferência chamada de resolução mostrada abaixo:

Resolução: $$\frac{? _1 ? \ldots ? ? _j ? \ldots ? ? _m \qquad ? _1 ? \ldots ? ? ? _k ? \ldots ? ? _n}{(? _1 ? \ldots ? ? _{j-1} ? ? _{j+1} ? \ldots ? ? _m ? ? _1 ? \ldots ? ? _{k-1} ? ? _{k+1} ? \ldots ? ? _n) ?}$$

Onde $? _i$ e $? _i$ são literais e ? é o unificador mais genérico de $? _j$ e $? _k$.

Intuitivamente, esta regra diz que, como $? _j$ e $? _k$ unificam, $? _j$ e $? ? _k$ não podem ser simultaneamente verdadeiros, logo algum outro literal das duas cláusulas deve ser verdadeiro.

Robinson e Wos (1969) apresentam um regra de inferência, chamada de paramodulação, para tratar lógicas de primeira ordem com igualdade:

Paramodulação: $$\frac{? \qquad t_1 \sim t_2 ? ?}{? ' ? \ (? ?)}$$.[3]

Onde ? é uma fórmula, ? é uma substituição e t_1 e t_2 são termos tais que:

? contém um termo t_3,

existe uma substituição ? tal que $t_1 ? = t_3 ?$ e

? ' é a fórmula obtida substituindo-se as ocorrências de $t_3 ?$ em ? ? por $t_2 ?$.

Intuitivamente, esta regra diz que é possível fazer substituições nas fórmulas com base nas relações de igualdade existentes (exemplo: se $a \sim b$ então pode-se substituir as ocorrências de b em ? por a).

As regras resolução e paramodulação juntamente com o axioma lógico da reflexibilidade formam um sistema completo para lógicas de primeira ordem com igualdade (conforme demonstrado por Robinson e Wos (1969)). O provador de teoremas utilizado na parte prática desta dissertação (OTTER) é baseado neste sistema.

---

[2] Russell e Norvig (1995) explicam passo a passo o algoritmo que converte fórmulas para cláusulas. Nas notas bibliográficas do mesmo capítulo (página 292), são listados os autores cujos trabalhos contribuíram para o desenvolvimento deste algoritmo.

[3] Será utilizada a notação infixa com o predicado de igualdade ~.

2.5 Decidibilidade e complexidade da inferência em LPPO

O problema tratado nesta dissertação pode ser enunciado da seguinte forma: dada uma base de conhecimento K e uma sentença ?, deseja-se saber se $K \models ?$. Como a linguagem de representação escolhida é a LPPO, este problema será chamado aqui de "o problema da inferência em LPPO".

Em 1930, Gödel demonstrou em sua tese de doutorado que este é um problema *semi-decidível*, isto é, se de fato $K \models ?$ então é possível ter um algoritmo que encontre uma prova de ? a partir de K (o que ficou conhecido como o teorema da completude de Gödel). Entretanto, até 1936 não se sabia se o problema era *decidível*, isto é, não se sabia se é possível escrever um algoritmo que "decida" em uma quantidade de tempo finita que ? não é consequência lógica de K (quando este for o caso).

Foi Church quem, em 1936, provou a indecidibilidade da inferência em LPPO[4], provando que é impossível criar um algoritmo que responda corretamente se $K \models ?$ ou $K \not\models ?$ para todo K e ?. Isto é, para todo algoritmo que se desenvolva é possível apresentar um conjunto de fórmulas K e uma fórmula ? tal que $K \not\models ?$ e que faz com que o algoritmo execute indefinidamente.

Apesar da semi-decidibilidade provada por Gödel, foi Robinson (1965) quem apresentou o primeiro algoritmo completo para inferência em LPPO. Apesar de ineficiente do ponto de vista computacional, o algoritmo de Robinson tem sido o fundamento teórico para o desenvolvimento dos principais provadores de teorema da atualidade (incluindo aqui o OTTER, o provador de teoremas utilizado nesta dissertação).

Cook, em 1971, ao lançar as bases da teoria da complexidade, provou que o problema da satisfatibilidade (que é complementar ao problema da inferência) em lógica proposicional é um problema NP-completo. Se um problema P é NP-completo, então é "bem provável" que não exista um algoritmo polinomial que resolva P. O fundamento para esta crença é que existe uma grande classe (chamada de NP) de problemas que têm sido estudados por décadas e para os quais ainda não foi encontrado um algoritmo polinomial e que podem ser reduzidos polinomialmente para P (de forma que se alguém resolver P em tempo polinomial então todos os problemas da classe NP também serão resolvidos em tempo polinomial).

Como a lógica de predicados é um superconjunto da lógica proposicional, sabe-se que a inferência em lógica de predicado é um problema NP-difícil, isto é, pelo menos tão difícil quanto qualquer problema NP-completo.

---

[4] Church publicou seu teorema em 1936 no artigo "*An unsolvable problem in elementary number theory*" no volume 58 do "*American Journal of Mathematics*".

Conhecer estes resultados de decidibilidade e complexidade da inferência em LPPO é importante para entender que procurar um algoritmo ótimo para resolver este problema não é um caminho muito promissor. Logo, tem-se uma maior chance de sucesso pesquisando-se heurísticas que podem não produzir um resultado ótimo em todos os casos, mas que trazem benefícios no caso geral (que é exatamente o objetivo desta dissertação).

No próximo capítulo será apresentado um provador de teoremas (o OTTER) que é capaz de realizar inferência a partir de uma base de conhecimento representada em uma linguagem de LPPO aplicando as regras de resolução e paramodulação.

3 O Provador de Teoremas OTTER

Este capítulo descreve o provador de teoremas OTTER que será utilizado como o mecanismo de inferência na parte prática desta dissertação. O OTTER é um dos principais provadores de teorema atuais, desenvolvido no Laboratório Nacional de Argonne, um dos institutos pioneiros na pesquisa de provadores de teoremas. Ele foi escrito em C ANSI, seu código é livre e pode ser obtido em http://www-unix.mcs.anl.gov/AR/OTTER.

Apesar de trabalhar apenas com cláusulas, o OTTER aceita como entrada fórmulas gerais que são automaticamente transformadas para cláusulas antes de iniciar-se o processo de inferência.

O OTTER gera provas por refutação, isto é, para provar que $K \models ?$, o OTTER procura por uma inconsistência no conjunto K ? {? ? }. Percebe-se que uma premissa necessária aqui é que o conjunto K seja consistente pois realizar inferência a partir de uma base de conhecimento inconsistente é um assunto delicado já que qualquer fórmula ? é consequência lógica de um conjunto K de fórmulas inconsistentes. Ora, se K é consistente, a inconsistência de K ? {? ? } só poder vir do fato de que $K \models ?$.

3.1 O princípio da resolução e a estratégia do conjunto suporte

O algoritmo de inferência do OTTER é baseado no princípio da resolução introduzido por Robinson (1965). Robinson definiu dois operadores de resolução R e R^n da seguinte forma:

- ? Seja K um conjunto de cláusulas, $R(K)$ é o conjunto de cláusulas obtido tomando-se as cláusulas de K mais as cláusulas obtidas através da aplicação da regra de inferência da resolução em todos os pares de cláusulas de K para os quais é possível aplicar a regra da resolução.
- ? Seja K um conjunto de cláusulas e n um número natural, então $R^n(K)$ é o conjunto de cláusulas definido da seguinte forma: $R^0(K)=K$, $R^{n+1}=R(R^n(K))$.

Com estas definições, Robinson demonstrou o seguinte teorema:

> **Teorema 1 (teorema da resolução)**: se K é um conjunto finito de cláusulas, então K é inconsistente sse $R^n(K)$ contém []⁵ para algum n ? 0.

[5] Esta é a notação para a cláusula vazia que representa a inconsistência (?). Uma cláusula vazia é obtida aplicando a resolução em um par de cláusulas unitárias complementares, isto é, do tipo ? e ? ? .

Este teorema indica que é possível escrever um algoritmo de prova por refutação completo que aplica o operador R sucessivamente, começando com o conjunto K ? {? ? }, até que uma inconsistência (a cláusula vazia) seja encontrada.

Entretanto, Robinson mesmo observa que este método é muito ineficiente, pois a cada aplicação de R são geradas muitas cláusulas irrelevantes para a inferência desejada. Wos, Carson e Robinson (1965) introduziram a estratégia do conjunto suporte com o objetivo de lidar com este problema. Esta estratégia restringe os pares de cláusulas nos quais a regra da resolução é aplicada da seguinte forma:

- ? Suponha que se deseja provar que K |= ? . Então seja $K' = K$? {? ? }.
- ? Seja T ? K' tal que K'-T é um conjunto de cláusulas consistente. T é chamado de o conjunto suporte.
- ? A cada passo, o operador de resolução é aplicado somente nos pares de cláusulas nos quais uma das cláusulas está em T ou é uma cláusula que foi gerada em um passo anterior.

Apesar de não ser estritamente necessário, a idéia básica é ter em T a negação da asserção que se deseja provar. Assim, se uma cláusula é gerada no decorrer do processo, a prova da mesma tem em sua origem uma das cláusulas da asserção que se deseja provar. Isto dá um aspecto de "direcionamento por objetivo" (*goal directed*) no espaço de busca do algoritmo.

Wos, Robinson e Carson provaram que, com a premissa de que K'-T é consistente, este método é completo.

3.2 Regras de inferências utilizadas pelo OTTER

Além das regras de inferência da resolução e paramodulação apresentadas no Capítulo 2, o OTTER pode[6] utilizar as seguintes regras:

Resolução UR[7]:

$$\frac{?_1 ?_2 \ldots ?_n \quad ?_1 ? \ldots ? ?_n ? ?_{n+1}}{(?_{n+1}) ?_1 ?_2 \ldots ?_n} .$$

Onde $?_{n+1}$ é um literal e $?_i$ e $?_i$ são literais, $?_i$ é o unificador mais genérico de $?_i$ e $?_i$ e $?_i ?_i$ e $?_i ?_i$ formam um par complementar (para i=1,2...n).

Hiper-Resolução:

$$\frac{?_1 ?_1 \quad ?_2 ?_2 \ldots ?_n ?_n \quad ? ?_1 ? \ldots ? ? ?_n ? ?_{n+1}}{(?_1 ? \ldots ?_n ? ?_{n+1}) ?_1 ?_2 \ldots ?_n} .$$

Onde $?_i$ e $?_{n+1}$ são cláusulas positivas, $?_i$ e $?_i$ são literais positivos e $?_i$ é o unificador mais genérico de $?_i$ e $?_i$ (para i=1,2...n).

---

[6] "pode" porque na realidade o usuário é quem define quais regras de inferência serão utilizadas.

[7] O nome UR(=*Unit Resulting*) indica que a cláusula gerada é unitária (contém apenas um literal).

Demodulação: $\dfrac{? \quad t_1 \sim t_2}{?'}$.

Onde ? é uma fórmula, t_1 e t_2 são termos tais que:

? contém um termo t_3,

existe uma substituição ? tal que $t_1? = t_3$ (t_3 é uma instância de t_1)e

?' é a fórmula obtida substituindo-se as ocorrências de t_3 em ? por t_2?.

Fatoração: $\dfrac{? \; ? \; ? \; ? \; ?}{(? \; ? \; ?)?}$.

Onde ? é uma cláusula, ? e ? são fórmulas atômicas e ? é o unificador mais genérico de ? e ?.

Percebe-se que, na realidade, as duas primeiras regras acima correspondem à aplicação simultânea de várias regras da resolução (a regra de resolução original é chamada de resolução binária). A vantagem disto é evitar a geração de cláusulas intermediárias. Já a demodulação é um caso específico da paramodulação, sendo que a cláusula original não é mantida (a idéia é realizar simplificações nas cláusulas, por exemplo substituindo-se irmão(pai(x)) por tio(x)). A fatoração também é uma regra de simplificação que elimina um literal desnecessário de uma cláusula.

3.3 Simplificações das cláusulas geradas

Sempre que uma nova cláusula é gerada, o OTTER aplica uma série de simplificações com o objetivo de reduzir o tamanho da mesma ou até mesmo eliminar a cláusula definitivamente. Este tipo de funcionalidade é fundamental em provadores de teorema, caso contrário a quantidade de cláusulas geradas alcançaria rapidamente o patamar da inviabilidade. Além das regras de fatoração e demodulação, o OTTER aplica as seguintes simplificações nas cláusulas geradas:

- **Remoção unitária (*unit deletion*)**: remover um literal ? da cláusula gerada se, no conjunto de cláusulas geradas até então, existe uma cláusula unitária ? tal que ? é uma instância de ?.
- **Remoção de cláusulas subordinadas (*subsumption*)**: uma cláusula ? é subordinada por uma cláusula ? (ou ? subordina ?) se existe uma substituição ? tal que todos os literais de ?? aparecem em ?. Por exemplo $p(x)$ subordina $p(a)$? $p(b)$. Intuitivamente, $p(x)$ é mais simples e mais genérico do que $p(a)$? $p(b)$.
- **Remoção de tautologias**: se a cláusula gerada é uma tautologia então ela é descartada.
- **Remoções de cláusulas "muito grandes"**: pode-se configurar o OTTER para ignorar cláusulas com muitos literais e/ou muito termos (o que, naturalmente, compromete a completude).

3.4 O algoritmo do OTTER

O OTTER implementa o algoritmo da cláusula dada, desenvolvido com base nas idéias de Overbeek (1974), que pode ser visto como uma implementação da estratégia do conjunto suporte. A Figura 1 mostra o algoritmo da cláusula dada conforme a implementação (e notação) do OTTER.

Algoritmo da Cláusula Dada
Entrada
sos: o conjunto suporte (***set of support***).
usable[8]: suponha que se deseja provar que $K \models ?$,
então usable = K ? {? ?} - sos.
Saída
Sim: sos ? usable é inconsistente (insatisfazível).
Não: sos ? usable é consistente (satisfazível).

Enquanto (sos não estiver vazio e não encontrou uma prova) **faça**

Escolha a "melhor" cláusula em sos como a clausula_dada.

Mova a clausula_dada de sos para usable.

Aplique as regras de inferências ativadas para todos os pares de cláusulas formados pela clausula_dada e uma outra cláusula de usable. Cada nova cláusula gerada sofre um conjunto de simplificações e passa por um processo eliminação de redundâncias e/ou cláusulas indesejadas. As cláusulas que passam por este processo sem serem eliminadas são acrescentadas a sos.

Figura 1 - O algoritmo da cláusula dada

Percebe-se que um dos principais passos do algoritmo é a escolha da "melhor" cláusula. O procedimento padrão do OTTER é escolher a cláusula mais "leve" em *sos*, sendo que o peso de uma cláusula é proporcional à quantidade de ocorrência de termos e literais na mesma. Entretanto, é possível alterar a forma com que o OTTER calcula o peso das cláusulas.

A completude do OTTER depende muito das escolhas realizadas. Algumas combinações de regras são completas e outras não. O capítulo "*Soundness e Completeness*" do manual de referência discute vários pontos sobre a completude da inferência com o OTTER.

O próximo capítulo vai definir uma noção de relevância em bases de conhecimento que será utilizada em uma implementação, baseada no OTTER, para testar a hipótese desta dissertação.

[8] A nomenclatura *usable* vem do fato de que são as cláusulas deste conjunto que serão **usadas** no decorrer do processo para gerar novas cláusulas.

4 Relevância em Bases de Conhecimento

Neste capítulo, é dada uma formalização para o conceito de relevância entre as sentenças de uma base de conhecimento representada em LPPO com igualdade e funções.

O capítulo começa (seção 4.1) apresentando a lógica de relacionamento (*relatedness logic*) de Epstein (2001). Apesar da motivação de Epstein ser totalmente distinta da motivação desta dissertação, os argumentos nos quais Epstein fundamenta sua definição de relacionamento entre fórmulas são análogos aos conceitos utilizados na definição de relevância que será utilizada nesta dissertação.

4.1 Lógica de relacionamento

O conceito de relevância tem sido aplicado no estudo de lógicas com o objetivo de tratar os chamados paradoxos da implicação material das sistemas clássicos. O exemplo abaixo, comum na literatura da área, demonstra o tipo de paradoxo referido:

A lua é feita de queijo. Portanto está chovendo ou não está chovendo.

Ora, como a consequência da implicação acima é verdadeira, a implicação em si é verdadeira independentemente do valor verdade da premissa. Isto é:

? ? (? ? ? ?) para quaisquer fórmulas ? e ?.

Percebe-se que, apesar de parecer incoerente, este é um raciocínio válido de acordo com a semântica adotada nas lógicas clássicas. O conceito de relevância pode ser útil para evitar este tipo de anomalia através da restrição de que uma implicação coerente deve relacionar premissas e conseqüências relevantes entre si. De volta ao exemplo, não existe relação alguma entre o fato da lua ser feita de queijo e estar ou não chovendo. Logo, do ponto de vista teórico, faz-se necessário o desenvolvimento de lógicas que evitem este tipo de incoerência, caso contrário os sistemas lógicos não serão capazes de servir como instrumento de raciocínio em bases de conhecimento heterogêneas (que eventualmente contenham informações irrelevantes entre si).

Epstein (2001) apresenta um sistema de lógica proposicional, chamado de lógica de relacionamento, cuja semântica leva em consideração uma relação binária R entre fórmulas que indica se duas fórmulas estão relacionadas entre si. No sistema de Epstein, se duas fórmulas ? e ? não estão relacionadas, então ? ? ? é falso independentemente dos valores verdade de ? e ?. E se duas fórmulas ? e ? estão relacionadas, então o valor verdade de ? ? ? é ditado pelas regras da semântica clássica. A tabela abaixo mostra como fica a semântica do conectivo ? em uma lógica de relacionamento (onde "*" = 0 ou 1):

?	?	R(? ,?)	? ? ?
*	*	não	0
0	*	sim	1
1	0		0
1	1		1

Epstein sugere que R tenha as seguintes propriedades:

- **R1 (reflexibilidade)**: R(? , ?).
- **R2 (irrelevância da negação)**: R(? ,?) sse R(? ? , ?).
- **R3 (simetria)**: R(? , ?) sse R(? , ?).
- **R4 (irrelevância dos conectivos lógicos)**[9]: R(? , ? ? ?) sse R(? , ?) ou R(? , ?), onde ? é qualquer um dos conectivos lógicos.

Com estas propriedades, Epstein demonstrou que é possível ter R como primitiva apenas das fórmulas atômicas e definir R (indutivamente) para fórmulas complexas com base nos constituintes das mesmas como sugere o seguinte lema:

Lema 1: R(? , ?) sse existem fórmulas atômicas ? ' e ? ' tal que ? ' é constituinte de ? e ? ' é constituinte de ? e R(? ', ? ').

Epstein oferece uma formulação teórica completa para a lógica de relacionamento de forma tal que a lógica proposicional clássica pode ser vista como um caso particular da lógica de relacionamento assumindo a premissa de que qualquer fórmula está relacionada com qualquer outra fórmula.

Estender estes resultados para lógicas de predicados não é simples. Krajewski (1986) apresenta algumas idéias sobre como fazê-lo, entretanto não foi dada (ainda) uma formulação teórica completa.[10] Krajewski sugere acrescentar à semântica de uma lógica de predicados uma relação primitiva *r* tal que:

- se *p* e *q* são constantes de predicado, então r(*p*,*q*) indica que *p* e *q* estão relacionados entre si;

---

[9] Na realidade, Epstein apresenta duas propriedades de R (uma para o conectivo ? e outra para ?). Epstein não considera os outros conectivos (? e ?) já que eles podem ser escritos em função de ? e ? .

[10] Krajewski termina seu artigo dizendo ser necessário um desenvolvimento mais detalhado das suas idéias, idéias estas baseadas em um curso ministrado por Epstein na Universidade de Iowa em 1982. Na realidade, está planejado um novo volume da série *The Semantic Foundations of Logic* no qual Epstein vai abordar o assunto.

? se c e d são constantes de indivíduo, então r(c,d) indica que c e d estão relacionadas entre si;
? se p é uma constante de predicado e c é uma constante de indivíduo, então r(p, c) indica que p e c estão relacionados entre si.

A partir da primitiva r, as seguintes condições podem ser utilizadas para definir a relação de relacionamento R entre duas fórmulas (de uma lógica de predicados) ? e ? :

I. Existem constantes de predicados p e q tais que p aparece em ? e q aparece em ? e r(p,q).
II. ? e ? possuem alguma variável livre em comum.
III. ? ocorre em ? ou ? .
IV. Existem constantes de indivíduo c e d tais que c aparece em ? e d aparece em ? e r(c,d).
V. Existem uma constante de predicado p e uma constante de indivíduo c tais que p aparece em ? (?) e d aparece em ? (?) e r(p,c).

As condições I, IV e V são bastante intuitivas. A intuição por trás da condição II acima é que se duas fórmulas compartilham uma variável livre, então elas sempre vão referenciar a mesma entidade do mundo real em qualquer modelo, e por isto elas devem ser consideradas relacionadas entre si. Já o fundamento da condição III é que uma fórmula que contenha ? faz alguma asserção sobre todas as entidades do domínio, logo ela estará relacionada com qualquer outra fórmula.

Não é fácil apresentar sistema lógico formal para lógicas de predicados que utiliza uma relação de relacionamento R considerando todas as 5 condições acima (Krajewski oferece alguns exemplos do tipo de complicações que são introduzidas pelas condições IV e V). Epstein (1982 apud Krajewski, 1986, p. 10)[11] oferece uma formulação completa para dois sistemas: um que considera apenas a condição I e outro que considera as condições I e II.

4.2 Uma noção de relevância baseada em sintaxe

A noção de relevância que se deseja obter nesta dissertação tem uma motivação bastante distinta da motivação que levou ao desenvolvimento da lógica de relacionamento de Epstein. Neste último, a motivação é evitar os paradoxos da implicação das lógicas clássicas enquanto que nesta dissertação a motivação é definir uma forma que, dada uma base de conhecimento K e uma sentença ? , permita selecionar as sentenças de K que são relevantes para se provar ? a partir de K. Apesar desta distinção, fundamentos análogos serão utilizados aqui para definir uma relação ? que relaciona fórmulas relevantes com o propósito aqui definido.

[11] Epstein, R. L. Notes for a course of lectures. Iowa State University. 1982.

Uma característica importante que se deseja na definição de ? é que a mesma não dependa de primitivas extras do sistema lógico (diferentemente das propostas de Epstein e Krajewski). Uma possibilidade para isto é ter a seguinte definição para ? :

> **Definição 1**: ?(? , ?) sse *C*(?) ? *C*(?) ? ? , onde *C* é uma função que retorna o conjunto de constantes (de predicado, de indivíduo e de função) que ocorrem em uma determinada fórmula.

Isto é, duas fórmulas são relevantes entre si caso possuam alguma constante de predicado, indivíduo ou função em comum. É fácil verificar que esta definição de relevância possui as propriedades R1-R4 apresentadas na subseção anterior, como é argumentado abaixo:

- ? **R1 (reflexibilidade)**: consequência imediata da definição dada para ?.
- ? **R2 (irrelevância da negação)**: ? e ?? possuem o mesmo conjunto de constantes de indivíduo, predicado e função, logo ?(? ,?) sse ?(? ? , ?).
- ? **R3 (simetria)**: consequência imediata da definição dada para ?.
- ? **R4 (irrelevância dos conectivos lógicos)**: seja *c* uma constante comum de ? e ? ? ? (? é um dos conectivos lógicos). Ora, *c* pertence a pelo menos uma das fórmulas ? e ? (já que a presença do conectivo ? não acrescenta nenhuma constante de predicado, indivíduo ou função). Logo, ?(? , ?) ou ?(? , ?). Então é verdade que ?(? , ? ? ?) sse ?(? , ?) ou ?(? , ?)

É interessante notar que a definição dada para ? pode ser obtida considerando-se as condições I e IV de Krajewski (apresentadas na seção anterior), estendo-se a condição IV para constantes de funções (isto é, considerando que r também relaciona constantes de funções) e assumindo que r é reflexiva. Ora, se duas fórmulas ? e ? compartilham uma constante *c* qualquer e r(c,c) então a condição I ou IV é satisfeita.

Percebe-se então que uma das hipóteses desta dissertação é que, para se implementar um algoritmo de inferência que faça uma seleção prévia das sentenças relevantes para a consulta, é suficiente que cada constante esteja relacionada com ela mesmo (isto é, que r seja reflexiva), o que significa dizer que é possível extrair informação de relevância a partir da própria sintaxe das sentenças. Ou seja, não é necessário que alguém defina uma relação que capture todos os relacionamento entre indivíduos e predicados relevantes entre si. Na realidade, este seria o melhor dos casos: ter atrelado na base de conhecimento estruturas extra lógicas que indicassem quais sentenças são relevantes entre si. A criação de grandes bases de conhecimento é uma tarefa laboriosa e a necessidade de se criar estruturas extras seria muito provavelmente um inconveniente. Entretanto, como será mostrado no Capítulo 6, bons ganhos de desempenho podem ser obtidos utilizando apenas a sintaxe das sentenças.

A condição II de Krajewski não foi utilizada na definição de ? pelo fato de que deseja-se uma relação aplicável em sentenças (que não possuem variáveis livres). Já a condição III desconsidera o fato de que é bastante comum ter o quantificador ? ocorrendo em uma fórmula do tipo $?x(p(x)?\ ?)$ o que, intuitivamente, estaria na realidade restringindo a asserção representada por ? para os indivíduos que possuem a propriedade representada pelo predicado p. Sendo assim, por motivos pragmáticos, a condição III não foi observada na definição dada para ? (já que esta considera que ? é sempre aplicável a todos os indivíduos do domínio).

A condição V não pode ser utilizada dado o objetivo de se definir ? sem a necessidade de primitivas extras no sistema lógico (veja que somente através de uma primitiva extra-lógica é possível relacionar uma constante de predicado com uma constante de indivíduo).

A noção de relevância obtida aqui através da definição da relação ? será chamada de relevância baseada em sintaxe pois nenhuma outra estrutura, além da própria sintaxe das fórmulas, é requerida para a definição de ?.

4.3 A coerência da relevância por sintaxe

A primeira argumentação colocada para a noção de relevância definida na seção anterior é o fato de que a mesma pode ser vista como a aplicação de algumas das condições apresentadas por Krajewski (e Epstein) para a noção de relacionamento entre fórmulas de uma lógica de predicado.

Além disto, nesta seção será analisada a coerência da noção de relevância acima definida através de uma sequência de exemplos. Serão analisados separadamente os casos em que se considera duas fórmulas relevantes entre si pelo fato de possuírem em comum cada um dos tipos de constante (de indivíduo, predicado e função). Será argumentado que no caso geral a noção de relevância proposta é coerente, entretanto podem existir casos particulares para os quais ela é inadequada (pelo menos para os propósitos desta dissertação). Estes casos particulares serão discutidos na próxima subseção.

Se duas sentenças distintas ? e ? possuem uma constante de indivíduo c em comum, então elas fazem declarações distintas sobre o mesmo indivíduo associado a c (um objeto, entidade, fato, etc) o que provavelmente faz as mesmas serem relevantes uma para a outra. O exemplo abaixo[12] traça um processo de raciocínio razoável, ilustrando a idéia apresentada acima:

Exemplo 1: suponha que deseja-se provar o seguinte teorema:

se A ? B = ? então A-B=A.

Ora, é razoável que tudo o que se sabe sobre o "indivíduo" ? seja considerado na busca da solução. Abaixo estão alguns exemplos de informações que podem estar disponíveis sobre o "indivíduo" ? :

x ? ? , para todo elemento x;

? ? X, para todo conjunto X;

X ? ? = ? para todo conjunto X e

se X ? ? então X = ? .

É bem intuitivo pensar que informações como estas sejam úteis no processo de dedução de qualquer informação relacionada com o "indivíduo" ? .

Já a análise do caso de duas sentenças ? e ? possuírem uma constante de predicado p em comum é pouco mais capciosa. Em primeiro lugar é interessante analisar os seguintes casos separadamente:

- ? ***p* tem aridade 0**: neste caso p é uma proposição e geralmente representa um fato do mundo real.
- ? ***p* tem aridade 1**: geralmente p representa uma propriedade dos indivíduos do mundo real.
- ? ***p* tem aridade maior que 1**: p representa relações entre os indivíduos.

Para o primeiro caso acima, a argumentação é bem parecida com a que foi dada para o caso das sentenças compartilharem uma constante de indivíduo. Ora, se deseja-se provar uma sentença ? que diz algo sobre um fato p, é natural que todas as informações sobre p da base de conhecimento sejam relevantes.

---

[12] Para manter a leitura dos exemplos mais natural, não foi utilizada a sintaxe da linguagem LPPO definida. Ao invés disto, os exemplos foram escritos em uma notação bem informal. Por exemplo, utiliza-se a notação usual da teoria dos conjuntos, no lugar do conectivo ? utiliza-se a construção "se ... então ...", etc.

Já para os outros dois caso, é interessante traçar um raciocínio separado para sentenças com e sem variáveis. Uma sentença sem variáveis que contém um predicado p de aridade maior que 0, contém alguma informação específica (propriedade ou relação) sobre um ou mais indivíduos (exemplo: filho(joselyto,allan)). Se a sentença contém variáveis, então ela deve descrever alguma regra envolvendo a propriedade ou relação representada pelo predicado p (exemplo: filho(x,y)? ama(x,y)).

Supondo que deseja-se provar a sentença ? que faz referência a um predicado p, parece ser mais propício considerar primeiro todas as sentenças que dizem alguma regra sobre p antes de se considerar os casos específicos. Isto é, as sentenças com variáveis que referenciam p devem ser consideradas mais relevantes que as sentenças sem variáveis que referenciam p, que por sua vez, são mais relevantes que as sentenças (com ou sem variáveis) que não referenciam p. Um exemplo aqui também pode ser bastante ilustrativo:

Exemplo 2: suponha que se deseja provar que

se A ? B e B ? C = ? então A ? C = ? .

Ora, todas as regras (sentenças com variáveis) conhecidas para as relações ? e ? devem ser relevantes para se encontrar a solução. Abaixo estão alguns possíveis exemplos de tais regras:

se A ? B e x ? A então x ? B;

se A ? B e B ? A então A = B e

se x ? A ? B então x ? A e x ? B.

Também podem ser relevantes os casos específicos (sentenças sem variáveis) envolvendo as relações ? e ? como os exemplos abaixo:

N ? R, onde N é conjunto dos números naturais e R é o conjunto dos números reais;

Cidades(São Paulo) ? Cidades(Brasil) e

Feriados ? TodosDiasDoAno.

Percebe-se que é bem intuitivo que as regras sejam mais relevantes que os casos específicos.

O exemplo abaixo evidencia mais ainda a intuição de que sentenças com variáveis devem ser mais relevantes que sentenças sem variáveis.

Exemplo 3: suponha que se deseja provar que

Cidades(São Paulo) ? Cidades(Brasil).

Ora, o que é mais relevante? Sentenças com variáveis como:

se A ? B e x ? A então x ? B;

se A ? B e B ? A então A = B e

se x ? A ? B então x ? A e x ? B.

Ou sentenças sem variáveis:

N ? R, onde N é conjunto dos números naturais e R é o conjunto dos números reais;

Cidades(Florida) ? Cidades(EUA) e

Feriados ? TodosDiasDoAno.

Por último, seja feita uma análise para o caso de duas sentenças ? e ? que compartilham uma constante de função *f*. Se a função *f* constitui o mesmo termo sem variáveis em ambas as sentenças ? e ?, então ? e ? devem ser consideradas relevantes pelo mesmo motivo que ? e ? seriam relevantes se possuíssem uma constante de indivíduo em comum. É fácil perceber isto já que um termo sem variável, assim como uma constante de indivíduo, representa exatamente um indivíduo do mundo real.

Caso contrário, então a argumentação de que ? e ? são relevantes é similar à argumentação apresentada quando ? e ? compartilham um predicado *p* que constitui uma sentença com variáveis. Isto é, uma sentença que contenha *f* constituindo um termo com variáveis deve ser alguma regra relacionada com *f* e portanto deve ser considerada relevante para provar qualquer sentença que referencie *f*. Novamente será apresentado um exemplo para melhor ilustrar as idéias discutidas acima:

Exemplo 4: nos exemplos anteriores, foi sugerido que a relação ? fosse representada por um predicado. Entretanto, ? pode ser representada como uma função já que dado dois conjunto A e B existe um único conjunto C tal que A ? B = C. Suponha que deseja-se provar a seguinte afirmação:

A ? B = A - (A-B).

Ora, todas as regras sobre a função ?, como a listada abaixo, devem ser consideradas:

se x ? A ? B então x ? A e x ? B.

Já fatos como "R ? N = N"[13] poderiam ser considerados menos relevantes.

Agora se a pergunta fosse "0 ? R ? N", então "R ? N = N" seria tão relevante quanto se "x ? A ? B então x ? A e x ? B" é relevante para "A ? B = A - (A-B)".

[13] A interseção dos conjuntos dos números reais com o conjunto dos números naturais é igual ao conjunto dos naturais.

Nota-se que a função ? de relevância definida anteriormente não faz distinção entre sentenças (ou termos) com e sem variáveis. Isto é, apesar de observada uma certa diferença na noção de relevância entre sentenças (ou termos) com e sem variáveis, esta dissertação não vai considerar este fato, ficando o mesmo como objeto de estudo futuro. Entretanto, fica aqui uma observação interessante, é bem provável que se ? for considerar esta diferença na estrutura das sentenças (ou termos), ? não será mais uma relação simétrica o que pode ser visto pelo seguinte exemplo:

> **Exemplo 5**: suponha que deseja-se provar a seguinte sentença sem variáveis "Cidades(São Paulo) ? Cidades(Brasil)".
>
> Ora, é bem razoável que as regras contendo ? , como "se A ? B e B ? C = ? então A ? C = ? ", sejam consideradas relevantes.
>
> Já o contrário pode não ser coerente, isto é, é bem provável que "Cidades(São Paulo) ? Cidades(Brasil)" não seja relevante para se provar que "se A ? B e B ? C = ? então A ? C = ? ".

Os exemplos e argumentos apresentados até agora apóiam a hipótese de que se duas sentenças ? e ? compartilham uma constante (predicado, indivíduo ou função), então ? e ? são relevantes entre si. Entretanto, nada foi dito ainda para argumentar o outro lado da definição de relevância apresentada, isto é, se duas sentenças ? e ? são relevantes então existe uma constante comum em ? e ? . A intuição parece dizer que isto não é verdade, isto é, podem haver sentenças relevantes entre si que não possuem constantes em comum. Entretanto, não é muito fácil enumerar exemplos destes casos. Veja um exemplo abaixo:

> **Exemplo 6**: suponha que se deseja provar a seguinte afirmação: "José é suspeito de estar infectado com febre amarela" representada pela seguinte sentença em LPPO:
>
> ? = SuspeitoEstarInfectado(Jose, FebreAmarela).
>
> Suponha que entre as informações disponíveis está o fato de que o inseto Aedes Aegypti reside nas florestas tropicais da América do Sul representado aqui pela seguinte sentença:
>
> ? = Reside(AedesAegypti, FlorestasTropicaisAmericaSul).
>
> Ora ? e ? não compartilham nenhuma constante, então pela definição dada de relevância, ? e ? não devem ser consideradas relevantes. Entretanto, a intuição diz o contrário.

O tipo de relevância sugerido pelo exemplo acima poderia ser capturado se a base de conhecimento contivesse estruturas extra-lógicas que permitisse relacionar por exemplo as constantes de indivíduo AedesAegypti e FebreAmarela, isto é, dentro da proposta de Krajewski, a interpretação da linguagem deveria conter a informação explícita de que a relação r(AedesAegypti,FebreAmarela) vale. No entanto, pode-se argumentar aqui que, na realidade, esta relação pode ser obtida analisando outras sentenças existentes na base de conhecimento como "o vetor da febre amarela é o inseto Aedes Aegypti" e "José vive numa cidade da América do Sul perto de uma floresta tropical". Isto pode feito através do conceito de grau de irrelevância entre sentenças introduzido por Wassermann (2000). Para definir formalmente este conceito é necessário antes definir o conceito de caminho entre sentenças em uma base de conhecimento conforme abaixo:

> **Definição 2**: seja *K* uma base de conhecimento. Seja ? uma relação de relevância das sentenças de *K*. Então um caminho P, baseado em ?, entre duas sentenças ? e ? é uma sequência de sentenças P = [$?_0$, $?_1$, ..., $?_n$] tal que $?_0$=? e $?_n$=? e ?($?_i$,$?_{i+1}$) para todo 0 ? i < *n*. O tamanho de P, denotado por |P|, é igual a *n*.

Agora pode-se dar a definição de grau de irrelevância:

> **Definição 3**: seja *K* uma base de conhecimento. Seja ? uma relação de relevância entre as sentenças de *K*. Então o grau de irrelevância entre duas sentenças ? e ? com relação a ? é definido da seguinte forma:
>
> Se ? = ?, então ?(? , ?) = 0.
>
> Senão, se não existe um caminho P baseado em ? entre ? e ?, então ?(? , ?) = ? , indicando que ? e ? são totalmente irrelevantes de acordo com ? .
>
> Caso contrário, seja P um caminho mínimo[14] entre ? e ?, então ?(? , ?) = |P|.

A Figura 2 mostra a noção de grau de irrelevância entre uma sentença ? e as outras sentenças de uma base de conhecimento. Por exemplo, a figura mostra uma base tal que ?(? , ?) = ?(? , ?) = ?(? , ?) = ?(? , ?) = 1.

[14] Não existe um caminho P' entre ? e ? tal que |P'| < |P|.

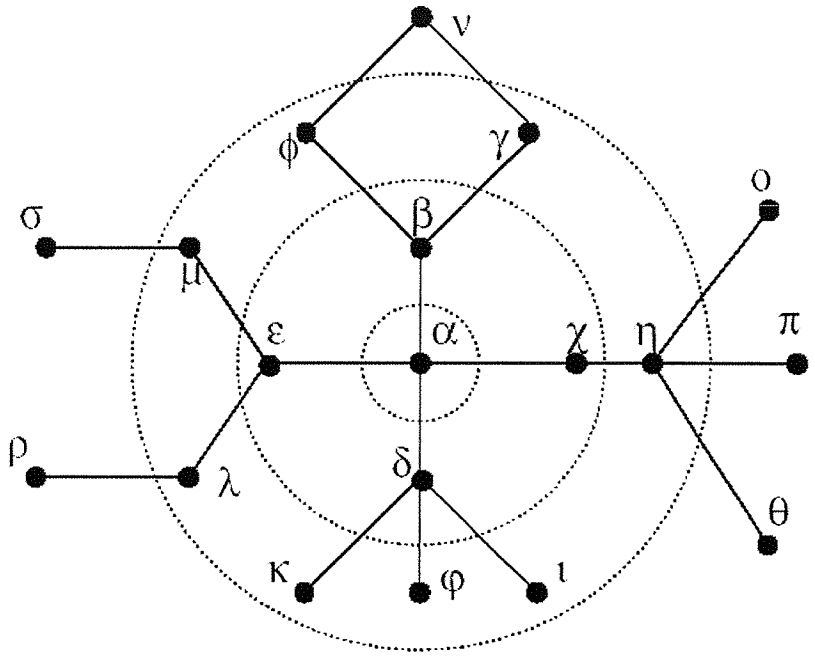

Figura 2 - Níveis relevância em uma base de conhecimento (obtida de Wassermann (2000), página 89).

Com esta definição de irrelevância, pode-se então estabelecer o grau de irrelevância máximo permitido para que uma sentença ? seja considerada no processo de inferência de uma sentença ?. Por exemplo, considere uma base de conhecimento K com as sentenças mostradas na Figura 2 exceto ? . Suponha que se deseja provar ? a partir de K e que seja estabelecido que o grau máximo de irrelevância para que uma sentença de K seja considerada relevante na busca por uma prova é 2. Então as sentenças ?, ?, ?, ?, ? e ? não participariam no processo de inferência. No Capítulo 5 será apresentada uma estratégia para inferências baseada nesta idéia.

4.4 Deficiências da relevância por sintaxe

A proposta desta dissertação é utilizar o conceito de relevância para selecionar um subconjunto (supostamente pequeno) de uma base de conhecimento a fim de obter um processo de inferência mais eficiente na prática. Neste sentido, a relevância por sintaxe definida anteriormente apresenta algumas deficiências que podem impedir o seu uso em certas situações. Isto é, podem haver casos em que a aplicação da seleção por relevância sintática não seja suficiente para reduzir consideravelmente o conjunto de sentenças a ser usado no processo de inferência.

Particularmente, isto vai ocorrer quando a base de conhecimento contiver uma quantidade demasiada de sentenças que referenciam uma determinada constante (e a fórmula que se deseja provar também referencia esta constante). Não é fácil pensar em exemplos desta situação para constantes de indivíduo ou função. Entretanto, é bem comum acontecer este fato para constantes de predicado, pois existem certas relações muito freqüentes, inclusive em domínios distintos. É o caso da relação especial de igualdade. Imagine uma base de conhecimento heterogênea com centenas de domínios, onde cada domínio estabelece dezenas de asserções envolvendo a relação de igualdade. Ora, para provar qualquer sentença que faça referência ao predicado que representa a igualdade nesta base de conhecimento, seriam consideradas todas as milhares de sentenças envolvendo o conceito de igualdade em diversos domínios.

Nestes casos, uma boa estratégia é identificar previamente estas constantes muito freqüentes e não considerá-las na definição da função de relevância ?. Isto é, seja X o conjunto de constantes muito freqüentes presente na base de conhecimento, então a função relevância fica assim definida:

? (? ,?) sse C(?) ? C(?) - X ? ? .

Analise também o seguinte exemplo:

> **Exemplo 7**: suponha que a constante que representa a igualdade, seja considerada uma constante muito freqüente e que deseja-se provar a seguinte sentença:
>
> ? ? se A ? B = A e A ? B = ? então A = B.
>
> Pela nova definição de relevância a condição = ? C(?) ? C(?) não é suficiente para se considerar ? e uma outra sentença ? relevantes. Desta forma a seguinte sentença ? não é relevante para ? :
>
> ? ? se A ? B e B ? A então A = B.
>
> Pois C(?) ? C(?) = {=}.

Novamente, comparando a definição de ? com o modelo de Krajewski apresentado anteriormente, percebe-se que está-se estabelecendo aqui que se uma constante x (de predicado, indivíduo ou função) ? X então não existe y tal que r(x,y).

Na realidade, Krajewski observa que o predicado de igualdade ~ deve realmente ser tratado de forma especial, apresentando 3 opções:

- ? r(x,~) para todo x ou
- ? não existe x tal que r(x,~) ou (esta é a abordagem adota nesta dissertação)
- ? se r(x,~) então x = ~ (esta é a abordagem que Krajewski utiliza em seu modelo).[15]

[15] Esta abordagem foi testada mas não apresentou bons resultados na prática (em relação ao desempenho).

Além disto, dado o objetivo proposto, é interessante estender este conceito para outros predicados que eventualmente ocorram com muita freqüência em determinados domínios. Por exemplo, se a base de conhecimento possui informações sobre o domínio de matemática, então os predicados que representam < (menor que) e > (maior que) são candidatos a pertencerem ao conjunto X acima definido.

Em uma primeira análise, poderia-se pensar que, desta forma, existe um risco alto de deixar de se considerar sentenças importantes para a inferência. Entretanto, na prática, o que parece acontecer é que as outras constantes presentes nas sentenças serão suficientes para fazer com que a seleção sem as constantes muito freqüentes seja uma boa seleção, (os resultados práticos obtidos mostram isto, como é descrito no Capítulo 6). No exemplo **7** acima, todas as sentenças que possuem algumas das constantes ? , ? e ? seriam consideradas relevantes para ? , o que parece ser suficiente para o objetivo de se provar ? .

Uma forma interessante de visualizar estas chamadas "constantes muito freqüentes" é comparando-as com as constantes lógicas da linguagem. Ora, considerar duas sentenças relevantes porque elas possuem a constante de igualdade em comum seria parecido com considerá-las relevantes se possuem, por exemplo, a constante de disjunção (?) em comum, o que é intuitivamente não muito coerente. Além disto, pode-se dizer que na realidade é muito comum tratar o predicado de igualdade de forma especial, por exemplo, como foi feito nesta dissertação, tratando-o como um predicado especial com significado fixo.

Também vale a pena notar aqui os seguintes casos em que a relevância baseada em sintaxe pode não ser adequada:

- ? A base de conhecimento é pequena ou muito homogênea, isto é, possui muitas informações sobre o mesmo assunto.
- ? Em geral, as consultas apresentadas ao sistema representam sentenças muito complexas cuja derivação de uma prova vai requerer praticamente considerar quase todas as sentenças da base de conhecimento.
- ? A base de conhecimento representa um conjunto limitado de axiomas para provar um único e (bastante) complexo problema. Este caso é bastante comum entre os matemáticos que se utilizam de provadores automáticos de teorema como ferramenta auxiliar.

Nos casos acima, de fato não há muito o que se argumentar a favor da utilização da noção de relevância apresentada. Sendo assim, estabelece-se que as idéias apresentadas nesta dissertação se aplicam a sistemas com grandes bases de conhecimento heterogêneas a partir da qual se deseja realizar uma grande quantidade de inferências diversas, sendo que a maior parte delas podem ser consideradas inferências simples. Não é difícil enumerar aplicações de um sistema como este:

- **Agentes inteligentes**: suponha por exemplo um robô que disponha de uma base de conhecimento com informações que o possibilite tomar decisões nas mais diversas situações. A cada momento, o robô se depara com uma situação para qual precisa tomar uma decisão. É intuitivo pensar que, em cada uma destas situações específicas, o robô selecione um subconjunto (bem) limitado das informações que o ele dispõe de forma que as inferências eventualmente necessárias seja realizadas rapidamente (caso contrário o robô não seria capaz de "sobreviver" em seu ambiente).
- **Sistemas de ajuda ao usuário**: um sistema deste tipo deve conter uma vasta quantidade de informações disponíveis em sua base de conhecimento, entretanto é bem provável que um pequeno subconjunto da mesma seja utilizado para responder cada pergunta do usuário.

Perceba que a restrição de aplicabilidade colocada aqui não quer dizer que está-se propondo um sistema que falha no caso geral. O algoritmo que será proposto no Capítulo 5 é de fato um algoritmo completo que em determinados casos (grandes bases heterogêneas e consultas simples) deverá melhorar a eficiência do algoritmo de inferência enquanto que em outros casos, ele poderá até mesmo piorar a eficiência. O objetivo da restrição é orientar em que casos alguém poderá ter ganhos aplicando-se as idéias apresentadas nesta dissertação.

Em um primeiro momento, pode-se questionar a utilidade de uma proposta que melhora a eficiência dos algoritmos de inferência somente nos "casos simples". Ora, este pensamento está equivocado pois não são raros os casos em que um algoritmo de inferência não consegue derivar uma prova, por mais simples que seja, em uma base de conhecimento muito grande. Por exemplo, entre os testes realizados houve dezenas de problemas que não foram resolvidos pelo OTTER no limite de tempo estabelecido (150 segundos) e que foram resolvidos instantaneamente pelo algoritmo implementado nesta dissertação (ver Capítulo 6 e Apêndice B).

No próximo capítulo será apresentado um algoritmo para inferência baseado no conceito de relevância definido neste capítulo.

5 Uma Estratégia para Inferência Baseada em Relevância

Neste capítulo é apresentada uma estratégia baseada em relevância para controlar a quantidade de sentenças geradas no decorrer do processo de inferência em grandes bases de conhecimento. A proposta apresentada é baseada na observação de que, em geral, não é uma boa estratégia considerar ao mesmo tempo tudo o que se sabe para resolver um determinado problema, diante do risco do processo de raciocínio se tornar impraticável em face a tantas possibilidades.

Logo, o que parece ser mais sensato é atacar um problema com um conjunto mínimo de informações consideradas mais relevantes para o mesmo. Após um certo tempo de trabalho sem sucesso, pode-se chegar a conclusão que as informações selecionadas não foram suficientes para resolver o problema, sendo necessário então acrescentar mais informações e tentar novamente. Esta é uma estratégia do tipo minimizar a quantidade de munição a ser usada para matar o inimigo, só que não se sabe exatamente o quanto o inimigo suporta, aí você vai colocando mais munição até o inimigo cair.

No contexto de inferência em grandes bases de conhecimento, o objetivo por trás desta estratégia é reduzir a geração de sentenças possivelmente irrelevantes para a inferência em questão. Isto é particularmente interessante em algoritmos baseados em saturação, como o algoritmo da cláusula dada, pois, em grandes bases de conhecimento, estes algoritmos podem gerar uma quantidade demasiada de sentenças no passo da saturação. Veja na Figura 1 do Capítulo 3 que, no terceiro passo dentro do laço do algoritmo da cláusula dada, são realizadas todas as inferências possíveis utilizando-se a cláusula dada e cada uma das cláusulas do conjunto *usable*. De forma que, se o conjunto *usable*, onde em geral estarão as cláusulas representando a base de conhecimento, for muito grande, existe a possibilidade de se gerar um grande número de cláusulas neste passo.

Três conceitos são necessários para viabilizar a implementação desta estratégia:

- A noção de relevância que vai definir a ordem em que as sentenças serão consideradas no processo de inferência.
- Os parâmetros que servirão para indicar que é chegado o momento de se acrescentar mais sentenças para a inferência. É necessário ter critérios para se decidir até onde (ou quando) deve-se tentar a inferência com o conjunto atual de sentenças relevantes antes de partir para a próxima etapa acrescentando-se mais sentenças.
- O tamanho inicial do subconjunto de sentenças relevantes a ser considerado bem como o ritmo de crescimento do mesmo no decorrer do processo.

Neste capítulo, será utilizada a noção de relevância baseada em sintaxe que foi definida no Capítulo 4 através da relação de relevância ? e da função de grau de irrelevância ?.

5.1 Um algoritmo para seleção de sentenças relevantes baseado em grafo de relevância

Wassermann (2000) sugere, no contexto de revisão de crenças, um algoritmo (Figura 3) que, dada uma sentença ? e uma base de conhecimento B, seleciona as sentenças de B relevantes para ? . Wassermann supõe a existência de uma relação binária R entre as sentenças de B indicando os pares de sentenças relacionadas entre si.

O algoritmo considera que o conjunto B ? {? }é representado como um grafo, chamado de grafo de relevância, no qual os vértices representam as sentenças existindo uma aresta entre dois vértices sse as sentenças representadas pelos vértices são relevantes entre si. Assume-se também que a cada sentença ? está associada uma lista *Adjacent*(?) com os vértices adjacentes a ? [16] no grafo de relevância.

No decorrer da execução do algoritmo, são construídos os conjuntos ?^{i}(? ,B) que contêm as sentenças ? ? B tais que ?(? ,?)=i, isto é o conjunto ?^{i}(? ,B) são as sentenças de B que possuem grau de irrelevância i em relação a ? . Percebe-se que, da forma como estes conjuntos são construídos e utilizados (passo 4.3), o algoritmo faz, na realidade, uma busca em largura no grafo de relevância. Nesta busca, a função *mark* marca os vértices já visitados o que possui dois propósitos: dentro do laço principal as "marcas" evitam tratar a mesma sentença mais de uma vez, e, no final do algoritmo (passo 5), as "marcas" são utilizadas para se saber quais sentenças devem ser retornadas.

O algoritmo de Wassermann é do tipo "interruptível a qualquer momento" (*anytime algorithm*), isto é, trata-se de um algoritmo que pode ser interrompido a qualquer momento, fornecendo sempre uma solução aproximada. Quanto mais tempo o algoritmo rodar, mais perto da solução ótima será a resposta retornada. Veja que esta funcionalidade do algoritmo é implementada através do uso da variável de controle *stop* no laço principal.

O trabalho de Wassermann considera uma lógica clássica proposicional. Entretanto, os mesmos conceitos podem ser aplicados em lógicas de predicados desde que seja dada uma definição apropriada para uma relação que defina as sentenças relevantes entre si na base de conhecimento

[16] Abusando um pouco da notação, ? aqui é na realidade o vértice do grafo de relevância que representa ?. Isto será feito no restante deste capítulo.

Algoritmo de Recuperação de Sentenças Relevantes
Entrada
? : uma sentença.
B: uma base de conhecimento.
Saída
Relevant: um subconjunto das sentenças de *B* relevantes para ? .

1. If ? ? *B*, then mark(?)
2. ? ^{1}(?,*B*):= Adjacent(?)
3. *i* := 1; stop := false
4. While not stop do
   4.1 For all ? ? ? ^{i}(?,*B*), mark(?)
   4.2 *i* := *i*+1; ? ^{i}(?,*B*) := ?
   4.3 For all ? ? ? $^{i-1}$(?,*B*),
   ? ^{i}(?,*B*) := ? ^{i}(?,*B*) ? {? ? Adjacent(?) s.t. not marked(?)}
   4.4 If ? ^{i}(?,*B*) = ? , then stop := true
5. Relevant := {? ? *B* s.t. marked(?)}

Figura 3 - O algoritmo de recuperação de sentenças relevantes (de Wassermann (2000), página 95)

5.2 Um algoritmo para inferência progressiva baseado em relevância

Nesta seção será apresentada a estrutura de um algoritmo cujo objetivo é controlar o espaço de busca de um determinado algoritmo de inferência explorando a noção de relevância entre sentenças numa base de conhecimento. A idéia é disparar o processo de inferência repetidamente, começando por um pequeno subconjunto das sentenças da base e acrescentando-se mais sentenças a cada passo até encontrar a solução do problema ou atingir um determinado limite de recursos. O conceito de relevância é utilizado para definir a ordem em que as sentenças participarão do processo de inferência (as sentenças menos relevantes participam do processo de inferência mais tarde). Este algoritmo será chamado de algoritmo para inferência progressiva baseado em relevância.

A idéia básica do algoritmo é similar à do algoritmo de Wassermann (mostrado na seção anterior), exceto que ao invés de retornar o subconjunto de sentenças relevantes (como faz o algoritmo de Wassermann), dispara-se um dado processo de inferência com o subconjunto de sentenças selecionado. Se a inferência falha com aquele subconjunto, então o algoritmo continua selecionando mais sentenças até um próximo ponto quando a inferência é disparada novamente. Este processo é ilustrado na Figura 4 onde K é a base de conhecimento (completa) considerada e K^i ? K é o subconjunto de K utilizado para inferência no i-ésimo passo do algoritmo progressivo.

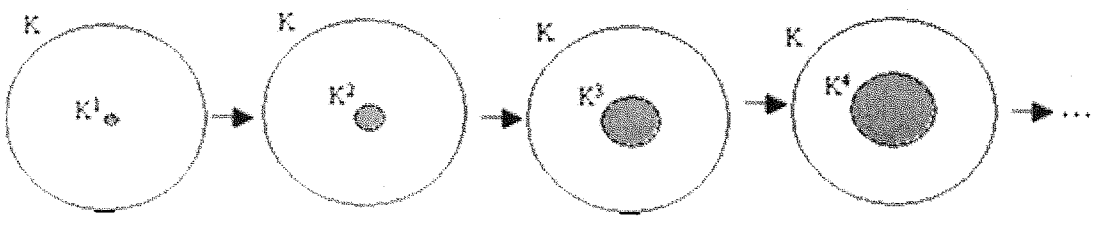

Figura 4 - Inferência progressiva em grandes bases de conhecimento

A Figura 5 mostra a estrutura do algoritmo proposto. Supondo que associado a K ? {? } existe um grafo de relevância G (da mesma forma definida por Wassermann), o algoritmo faz uma busca em largura começando pela sentença ? (da mesma forma como foi feito na seção anterior, quando se referenciar uma sentença ? desta forma, está-se na realidade referenciando o vértice de G que representa ?). A busca em largura é implementada através de uma fila Q com as operações padrão de *Enfileirar* e *Desenfileirar* (esta última remove e retorna o último elemento da fila). As subseções seguintes discutem os aspectos mais importantes do algoritmo.

Algoritmo para Inferência Progressiva Baseado em Relevância
Entrada

K: conjunto de sentenças representando uma base de conhecimento.
? : uma sentença a ser provada.
? : função que define a relação de relevância entre as sentenças de K ? {? }.
? : especificação do limite de recursos total a ser utilizado.
?: especificação do limite de recursos a ser utilizado em cada tentativa de inferência.
I: função que implementa um determinado algoritmo de inferência. O retorno de *I* deve ser:
S: sim, indicando que K^i |= ?
N: não, indicando que K^i |? ?
F: falha, o limite de recursos disponíveis definido por ? esgotou (*I* não conseguiu decidir se K^i |= ? ou K^i |? ? com ? recursos).
H: função que define se K^i é tal que está na hora de disparar a *i*-ésima inferência.

Saída

S: sim, indicando que *K* |= ?.
N: não, indicando que *K* |? ?.
F: falha, o limite de recursos disponíveis definido por ? esgotou (o algoritmo não conseguiu decidir se *K* |= ? ou *K* |? ? com ? recursos).

Estruturas de dados

i: um inteiro que, começando com 1, será incrementado a cada tentativa de inferência realizada.
K^i: subconjunto de *K* que será utilizado na i-ésima inferência.
Q: uma fila com as operações padrão *Enfileirar* e *Desenfileirar*.
? : uma estrutura com o objetivo de guardar o estado da inferência no passo atual para ser eventualmente reutilizado nos passos seguintes. A idéia é que *I* seja capaz de armazenar em ? quaisquer estruturas de dados que possibilite que uma eventual execução de *I* no (*i*+1)-ésimo passo seja capaz de reaproveitar o trabalho realizado no *i*-ésimo passo.

1. *i* ? 1
2. K^1 ? ?
3. *Q* ? ?
4. *Enfileirar*(*Q*,?); Marcar ?
5. Enquanto Q ? ? e os recursos utilizados não excederam ?
6. ? ? *Desenfileirar*(*Q*)
7. Para cada sentença ? tal que ? (?, ?)
8. Se ? não está marcada
9. *Enfileirar*(*Q*, ?); Marcar ?
10. K^i ? K^i ? {?}
11. Se *H*(*K*, K^i, *i*, ?, ?)
12. Se *I*(K^i, ?, ? , ?) = *S*, então Devolva *S*
13. *i* ? *i*+1; K^i ? K^{i-1}
14. Se os recursos utilizados não excederam ? , então Devolva *I*(K^i, ?, ? , ?)

Figura 5 - Algoritmo para inferência progressiva baseado em relevância

5.2.1 Implementação da busca em largura

No passo 7 são recuperadas todas as sentenças ? relevantes para a sentença ? que está sendo "expandida" na busca. Se ? ainda não foi marcada, ela entrará para o conjunto K^i e será enfileirada para ser expandida futuramente.

A marcação das sentenças é feita sempre que uma sentença é enfileirada e tem como objetivo garantir que cada sentença será expandida uma única vez. Isto é necessário já que o grafo de relevância pode conter circuitos do tipo $<?_0,?_1 \ldots,?_{n-1},?_n>$ onde $?_n=?_0$ e para todo $j>0$ vale $?(?_{j-1},?_j)$.

5.2.2 Complexidade da recuperação das sentenças relevantes

É possível obter uma implementação da operação de recuperação das sentenças relevantes (passo 7) com baixo custo mantendo-se para cada sentença ? da base de conhecimento uma lista com as sentenças relevantes para ?.[17]

Na realidade, desta forma transfere-se o custo da operação de recuperação para a manutenção destas listas o que vai depender da implementação da relação ? utilizada. No caso considerado nesta dissertação (relevância por sintaxe), este custo pode ser baixo mantendo-se um grafo bi-partido G' no qual os vértices são sentenças ou constantes (de predicado, indivíduo ou função).

Sempre que uma sentença ? for inserida na base, cria-se um conjunto de arestas entre o vértice associado a ? e os vértices associados às constantes que ocorrem em ?. Seja $|?|$ a quantidade de ocorrências de constantes e variáveis na sentença ? e n o número total de constantes da linguagem utilizada. Suponha que as constantes são mantidas em alguma estrutura que permita a operação de busca com complexidade logarítmica (por exemplo em uma árvore balanceada). Então inserir ? na base é $O(|?|*\log n)$.

Vale a pena notar também que, antes de disparar o algoritmo, deve-se inserir a sentença que se deseja provar em G' removendo-a depois de executado o algoritmo. Isto é, durante a execução do algoritmo, o grafo deve conter as sentenças $K \cup \{?\}$.

Por último, percebe-se que, dada uma base de conhecimento K, construir G' é $O(|K|*\log n)$ onde $|K|$ é a somatória das quantidades de ocorrências de constantes e variáveis em todas as sentenças de K.

5.2.3 A função *H*

No passo 11 é chamada a função H que decide se K^i é tal que está na hora de chamar o algoritmo de inferência pela i-ésima vez. Veja que são passados diversos parâmetros para H de forma que sua implementação disponha de informações suficientes para que a decisão seja tomada. Percebe-se que é H quem vai definir a sequência de K^i que será utilizada no decorrer da execução do algoritmo.

---

[17] Que seria a lista *Adjacent* utilizada na algoritmo de Wassermann(2000).

Uma implementação trivial de H seria, por exemplo, fazer com que a cada passo seja acrescentado em K^i aproximadamente 10% da base de conhecimento, como é mostrado abaixo:

Implementação 1 de *H*

$H(K, K^i, i, ?, s)$
  Devolva $|K^i|$? i*$|K|/10$

Onde a notação $|K|$ indica o tamanho da base de conhecimento representada por K (e não necessariamente a quantidade de sentenças de K). Percebe-se que é necessário definir exatamente como medir o tamanho de um conjunto de sentenças. Uma abordagem bem simplista seria tomar a quantidade de sentenças do conjunto como o seu tamanho. Outra abordagem (provavelmente mais realista) seria contar as ocorrências de todas as constantes e variáveis da linguagem em todas as sentenças da base.

A implementação de H pode variar muito de caso a caso. Por exemplo, se K é bem conhecida e não é muito volátil, alguém poderia definir uma sequência fixa de tamanhos para K^i. como no exemplo abaixo:

Implementação 2 de *H*

H(K, K^i, i, ?, s)
  Se i = 1, então Devolva $|K^i|$? 50
  Se i = 2, então Devolva $|K^i|$? 100
  Se i = 3, então Devolva $|K^i|$? 300
  Se i = 4, então Devolva $|K^i|$? 500
  Senão, Devolva$|K^i| = |K|$

Esta sequência poderia ser, por exemplo, "aprendida" estatisticamente executando-se várias sequências (com várias sentenças ?) e escolhendo-se a que apresentar os melhores resultados.

Note que também é a implementação de H que vai definir o número máximo de vezes que o algoritmo de inferência I é disparado. Por exemplo, com a implementação 1 mostrada acima, I seria disparado no máximo 10 vezes, enquanto que com a implementação 2, I poderia ser disparado no máximo 5 vezes.

5.2.4 Guardando o estado da inferência

No passo 12, é disparado o processo de inferência implementado pela função I. A estrutura ? é passada para I com o objetivo de que I consiga reaproveitar o trabalho que já foi realizado nas execuções anteriores de I. Também supõe-se que, se I retornar N ou F, será armazenado em ? informações suficientes para que o trabalho realizado por I possa ser reutilizado em uma eventual próxima execução de I.

5.2.5 Condições de parada do algoritmo

O algoritmo pára (retornando S) se I retorna S no passo 12. Isto é correto já que a lógica considerada aqui é clássica e portanto é monotônica[18], isto é, se $K^i \mid= ?$ então pode-se afirmar que $K \mid= ?$ (já que $K^i ? K$). Entretanto, o algoritmo não pára se I retorna N já que é possível que $K^i \mid? ?$ mas $K \mid= ?$.

Também vale a pena notar que não é garantido que o algoritmo pára se for dado um limite de recursos ($?$ ou $?$) infinito, pois a inferência na lógica considerada é semi-decidível.

5.2.6 Prosseguindo para o próximo passo

No passo 13, o algoritmo decide que será necessário partir para o próximo passo i, acrescentando-se mais sentenças em K^i.

Note que K^i é inicializado com K^{i-1}, de forma que $K^{i-1} ? K^i$, para todo $i>0$. Pode não parecer, mas esta é uma decisão "polêmica", pois é possível ter $?_1 ? K^{i-1}$ e $?_2 ? K^i - K^{i-1}$ tais que $?(?,?_1) = ?(?,?_2)$. Isto é, ir para o próximo passo i não significa considerar sentenças necessariamente menos relevantes do que aquelas que foram consideradas no passo i-1. Na realidade, $?_2$ é tal que $?(?,?_1) ? ?(?,?_2)$. Sendo assim, o algoritmo poderia ter outra estratégia diferente da adotada. Para ver isto, considere que uma parte de K é como está representado no grafo de relevância da figura abaixo:

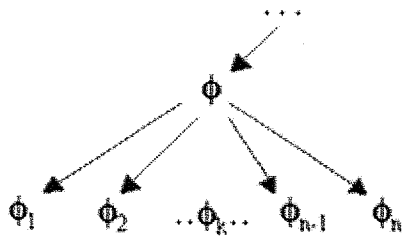

Figura 6 - Um grafo representando parte de uma base de conhecimento

[18] Uma lógica L é monotônica se for verdade que se $A ? B$ então $C(A) ? C(B)$, onde A e B são conjuntos de fórmulas da linguagem de L e C é o operador de fechamento da consequência lógica de L, isto é, $C(X)$ contém todas as fórmulas que são consequência lógica do conjunto de fórmulas X.

Suponha que $?(?,?)=g$ e que $?(?,?_j)=g+1$ para $j=1..n$. Suponha que no i-ésimo passo K^i é como mostrado abaixo:

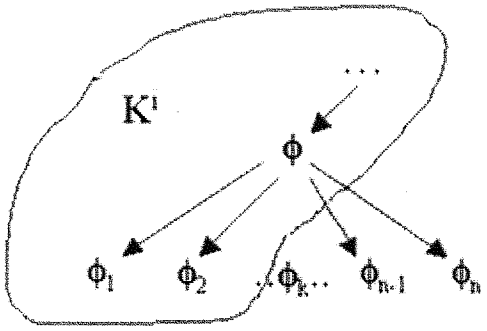

Figura 7 - Uma suposição para K^i no i-ésimo passo

Sendo assim, segundo a implementação dada do algoritmo, K^{i+1} será algo como mostra a figura abaixo:

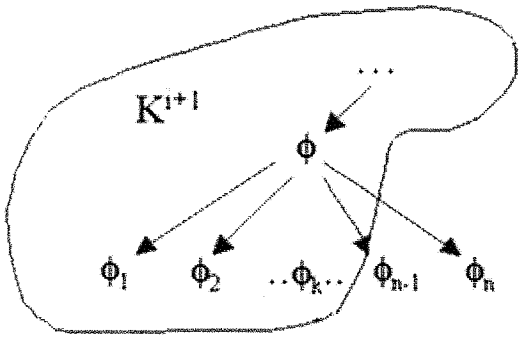

Figura 8 - Uma suposição para K^{i+1} segundo a implementação proposta

Entretanto, como $?(?,?_j)=?(?,?_k)$ para $j=1..n$, então alguém poderia dizer que K^{i+1} poderia ser igual a K'^{i+1} como mostra a figura abaixo:

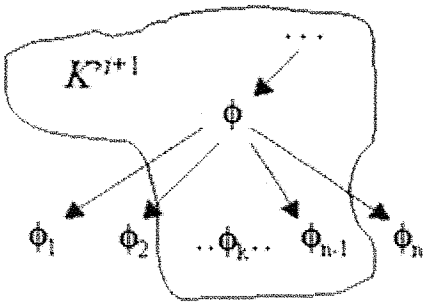

Figura 9 - Uma definição alternativa para K^{i+1} (diferente do que faz o algoritmo dado)

Argumentando que não existe motivo nenhum para se escolher K^{i+1} (Figura 8) ao invés de K'^{i+1}(Figura 9), já que um não possui sentenças menos relevantes do que o outro. Além disto, suponha que $|K'^{i+1}| < |K^{i+1}|$, então seria até preferível tentar K'^{i+1} antes de tentar K^{i+1} já que a hipótese assumida é que existe uma chance de se encontrar uma solução mais rapidamente considerando subconjuntos menores de sentenças. Logo, o passo 13 do algoritmo, que faz com que K^i? K^{i+1}, poderia ser reescrito para implementar outras estratégias. Entretanto, esta possibilidade não será estudada nesta dissertação, o que pode ser objeto de estudo futuro.

Além deste, existe outro aspecto do algoritmo para o qual existe uma alternativa diferente da que foi proposta. Ainda considerando as figuras anteriores, suponha que K^{i-1} seja como é mostrado abaixo:

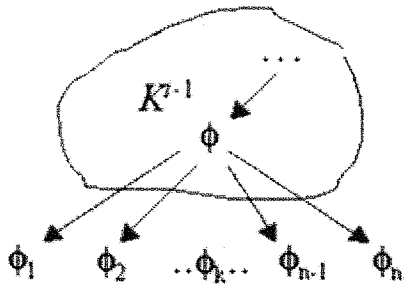

Figura 10 - Uma suposição para K^{i-1}

Como ?(? ,?$_j$) é o mesmo para j=1..n, então porque não fazer K^i igual ao K'^i abaixo (ao invés daquele mostrado na Figura 8):

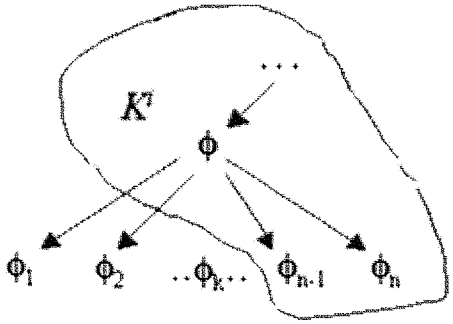

Figura 11 - Uma definição alternativa para K^i (diferente do que faz o algoritmo dado)

Já que, na realidade, não há uma razão específica para que ?$_j$ participe do processo de inferência antes de qualquer ?$_g$ para quaisquer j ? g.

Por último, vale a pena notar que, para manter a completude do algoritmo, qualquer que seja a estratégia que se implemente, esta deve garantir que exista i tal que ? ? K^i para sentença ? tal que ?(? ,?)?? . Isto é, em algum momento, todas as sentenças de K relevantes para ? serão consideradas.

5.2.7 O último passo do algoritmo

Por último, no passo 14, o algoritmo dispara o processo de inferência caso o limite de recursos disponíveis não tenha sido atingido. Este caso ocorrerá quando todas as sentenças de K relevantes para ? já tiverem sido selecionadas (quando a busca em largura terminar, isto é, quando Q=?).

5.2.8 Completude do algoritmo

O algoritmo será completo se as seguintes condições forem satisfeitas:

- ? o procedimento de inferência I é completo;
- ? ambos ? e ? são infinitos;
- ? a função H é tal que existe i tal que K^i=K', onde K' = {? ? K | ?(? ,?) ? ? }. Isto é, em algum momento devem ser consideradas todas as sentenças que têm alguma chance de pertencer a uma prova de ? . É fácil provar que se uma sentença ? é tal que ?(? ,?) = ? , então ? não faz parte de nenhuma prova de ? .

5.2.9 Complexidade do algoritmo

Seja n o tamanho da base de conhecimento K e m a quantidade de sentenças de K. Como já foi discutido anteriormente, pode-se ter uma definição simplista do tamanho de uma base fazendo $n = m$ ou ter uma definição mais "justa" por exemplo fazendo n igual à contagem de todas as ocorrências de constantes e variáveis em todas as sentenças da base. Nesta seção, será assumido que a definição do tamanho da base seja tal que n ? m ou, equivalentemente, que $m = O(n)$.

Suponha agora que a base de conhecimento é representada por um grafo de relevância G=(V,E). Veja que $|V| = m = O(n)$. Ora, em um grafo qualquer, $|E|$? $|V|$*($|V|$-1)/2, logo $|E| = O(m^2) = O(n^2)$. Logo, uma busca em largura em G tem complexidade $O(|V|+|E|) = O(n^2)$.

Seja $g(x)$ uma função tal que a complexidade do algoritmo de inferência I seja $O(g(x))$ em função do tamanho dos conjunto K^i que são passados como parâmetro para I no passo 12.

Suponha também que todos os passos do algoritmo, exceto o 12, são executados em O(1). Então, como o algoritmo executa uma busca em largura no grafo G, pode-se afirmar que a complexidade do mesmo é:

$$O(n^2) + ?\ O(g(|K^i|)).$$

Como $|K^i|$? $|K|$, então pode-se dizer que o algoritmo é

$$O(n^2) + O(g(n)).$$

Isto é, como era esperado, a complexidade do algoritmo depende diretamente da complexidade do algoritmo de inferência I utilizado. No caso desta dissertação, que considera uma base representada em LPPO, sabe-se $g(n)$ é exponencial. Na prática, serão obtidos ganhos nos casos em que a prova de ? é encontrada com um K^i pequeno.

Além disto, note que é possível manter o algoritmo quadrático fazendo com o que o limite de recursos ? seja definido como um limite máximo para K^i, pois assim $O(g(K^i))$ passa a ser limitado por uma constante. Os resultados dos testes realizados (Capítulo 6) sugerem que esta é uma boa estratégia (pois, de fato, grande parte das soluções foram encontradas com um K^i "pequeno").

Também vale a pena fazer uma análise aqui para o caso de se representar a base de conhecimento em uma linguagem menos expressiva tal que $g(n)$ seja menos complexa (por exemplo, sabe-se que existe um algoritmo polinomial para inferência com a linguagem de cláusulas *Horn*).

Neste caso, é interessante notar que, na prática, o primeiro termo da equação da complexidade do algoritmo, $O(n^2)$, não deve causar uma degradação quadrática no desempenho. Para que uma execução do algoritmo gaste um tempo da ordem do quadrado de n seria necessário que todas as sentenças da base fossem relevantes entre si. Ora, as idéias propostas nesta dissertação realmente não são aplicáveis neste caso extremo.

Entretanto, vale a pena notar que se $g(n)$ possui baixa complexidade (por exemplo, polinomial) é bem provável que a heurística apresentada não seja útil e foi exatamente por isto que se escolheu a lógica de predicados completa para os estudos desta dissertação.

Uma outra forma de visualizar a complexidade do algoritmo é especificar o primeiro termo da equação em função da quantidade k de relações de relevância entre as sentenças da base (na realidade, $k = |E|$). Neste caso, o algoritmo seria $O(k) + O(g(n))$.

5.3 Análise da heurística implementada pelo algoritmo

O fundamento teórico da heurística implementada pelo algoritmo apresentado na Figura 5 é que o algoritmo de inferência I, disparado no passo 12, não gera sentenças com grau de irrelevância, em relação a ? , acima do grau máximo de irrelevância encontrado nas sentenças de K^i mais uma unidade. Isto é, ?(? ,?') ? ?(? ,?) para toda sentença ? ? K^i e para toda sentença ?' gerada no espaço de busca[19] de I no i-ésimo passo.

---

[19] Pense no espaço de busca de um algoritmo de inferência I como o conjunto de sentenças que I gera no decorrer de sua execução na busca pela prova desejada. Considere também que se uma sentença ? pertence ao espaço de busca de uma execução de I que teve como entrada uma base K, então $K \models ?$.

Para provar esta afirmação será utilizado o seguinte teorema:

Teorema 2: Seja K um conjunto de sentenças consistente. Seja ? uma sentença e p uma constante de predicado que aparece em ? . Se K |= ? , então existe uma sentença ? ? K tal que p aparece em ? . Isto é, se uma sentença ? é consequência lógica de uma base de conhecimento K, então todas as constantes de predicado que aparecem em ? aparecem em pelo menos uma sentença de K.

Prova: Ora, K |= ? então existe uma prova ? = $<?_0, ..., ?_n>$ de ? a partir de K utilizando um algoritmo que aplica o princípio da resolução de Robinson (ver seção 3.1). O teorema será provado por indução no tamanho de ? .

Se ? = $<?_0>$ então $?_0$? K já que não há como ter $?_0$ como resultado da aplicação de nenhuma regra de inferência. Como ? = $?_0$, então a prova é trivial pois chegou-se a conclusão de que na realidade ? ? K.

Se ? = $<?_0, ..., ?_n>$ com $n > 0$. Se $?_n$? K, a prova é trivial pois novamente chega-se a conclusão de que ? ? K. Caso contrário, $?_n$ é o resultado da aplicação de alguma regra de inferência que tem como premissas um conjunto não vazio de sentenças S ? $\{?_0, ...,?_{n-1}\}$. Observando a regra da resolução (única regra de inferência utilizada para se encontrar ?) percebe-se que existe algum $?_i$? S tal que p ocorre em $?_i$, pois não é possível ter como resultado da aplicação da regra da resolução uma fórmula que contenha alguma constante de predicado que não ocorre em nenhuma das premissas. Se $?_i$? K, então o teorema está provado. Senão, suponha, sem perda de generalidade, que i é máximo no sentido de que não existe $?_j$, $j > i$, tal que p ocorra em $?_j$. Ora ?' = $<?_0, ..., ?_i>$ é uma prova para $?_i$ e pode-se aplicar aqui a hipótese de indução, isto é, supor que se p' ocorre em $?_i$ então existe uma sentença ? ? K tal que p' ocorre em ? . Ora, a elaboração da prova ? certamente não muda este fato, então é verdade que existe ? ? K tal que p ocorre em ? já que p ocorre em $?_i$.

Agora é possível enunciar e provar o seguinte teorema:

> **Teorema 3**: Seja m um inteiro e ? uma sentença tal que, no passo 12 do algoritmo, ? ? K^i, ?(?,?) = m e para todo ?' ? K^i, ?(?',?) ? m. Isto é, m é o grau máximo de irrelevância, em relação a ? , das sentenças de K^i no passo 12 do algoritmo. Então o espaço de busca de qualquer algoritmo de inferência que utilize somente a regra da resolução não vai conter nenhuma sentença ?' tal que ?(?',?) > m+1.
>
> **Prova**: Seja ?' uma sentença pertencente ao espaço de busca do algoritmo de inferência I disparado no passo 12. Seja p uma constante de predicado tal que p ocorre em ?'. Ora, como K^i |= ?', então pelo Teorema **2**, existe ? ? K^i tal que p ocorre em ? . Como ?(? ,?) ? m, então existe um caminho Q = <? , ... ?> de ? a ? de tamanho não maior que m (|Q| ? m). Então, como ?' e ? compartilham a constante de predicado p, Q' = <? , ... ? , ?'> é um caminho de ? a ?' de tamanho não maior que m+1 o que significa que ?(?',?) ? m+1.

O teorema acima prova que o tamanho do espaço de busca do algoritmo de inferência I é controlado através da restrição do grau de irrelevância das sentenças nos conjuntos K^i, já que, da forma como cada K^i é montado, se ? ? K^i então não existe ?' ? K-K^i tal que ?(?',?) < ?(?,?). Isto é, as sentenças em K^i são as mais relevantes para ? .

O próximo capítulo mostra como o algoritmo para inferência progressiva definido neste capítulo foi implementado utilizando o provador de teoremas OTTER como a implementação do algoritmo de inferência I. Além disto, serão apresentados resultados práticos da utilização deste algoritmo resolvendo problemas em uma base de conhecimento montada a partir da biblioteca TPTP.

6 Implementação, Testes e Resultados

Este capítulo descreve a implementação da estratégia apresentada no capítulo anterior e os testes realizados, fazendo uma análise dos resultados obtidos.

6.1 Utilização do OTTER como o algoritmo de inferência

Não foi possível escrever uma implementação em código do algoritmo da Figura 5 chamando o OTTER por dois motivos. Primeiramente, o OTTER não consegue salvar e reaproveitar o estado da inferência entre uma chamada e outra. Em segundo lugar, o limite de tempo de processamento implementado pelo OTTER não é confiável pois ele é verificado somente no laço principal do programa. Isto significa que é possível (e bem provável) que o OTTER ultrapasse o limite de tempo estabelecido porque ele selecionou uma cláusula muito prolífera como a cláusula dada e começou a fazer inferência com a mesma gerando milhares de cláusulas.

Para eliminar estes dois inconvenientes, seria necessário realizar alterações substanciais no código do OTTER. Ao invés disto, optou-se por um subterfúgio simples (mesmo abrindo mão da funcionalidade de guardar o estado da inferência entre uma execução e outra).

A solução encontrada foi alterar o código do OTTER para receber como parâmetro a quantidade de sentenças a ser considerada no processo de inferência. Após realizadas todas as tarefas de leitura dos arquivos de entrada, foi inserida a implementação do algoritmo de seleção das sentenças mais relevantes baseado em um grafo de relevância (de forma semelhante ao algoritmo da Figura 3). O grafo de relevância foi obtido lendo-se as próprias estruturas de dados do OTTER com o acréscimo de algumas listas extras. A utilização das próprias estruturas de dados do OTTER foi um fator bastante positivo na eficiência dos testes pois não foi necessário nenhum esforço redundante de leitura e preparação das informações nos arquivos de entrada.

A versão do OTTER assim modificada será chamada deste ponto em diante de RR-OTTER (Relevant Reasoning OTTER) e seu código fonte está disponível em <http://www.ime.usp.br/~joselyto/mestrado>. O Apêndice A oferece uma breve descrição das estruturas de dados do OTTER bem como das alterações realizadas na implementação do RR-OTTER.

Para resolver um dado problema, basta disparar várias execuções consecutivas do RR-OTTER aumentando-se gradativamente a quantidade de sentenças relevantes a ser considerada em cada execução até que uma solução seja encontrada ou todas as sentenças da base de conhecimento forem utilizadas.

O limite de recursos utilizado foi simplesmente limitar a quantidade de tempo que o RR-OTTER executa em cada passo bem como limitar o tempo total das várias execuções. Para garantir o limite de tempo desejado em cada execução, as chamadas ao RR-OTTER foram feitas através do *shell tcsh*[20] utilizando-se o comando "*limit cputime*" que possibilita abortar a execução de um processo independentemente da situação interna do mesmo.

Outro ponto que vale a pena notar aqui é a configuração das opções do algoritmo do OTTER, pois não basta simplesmente apresentar um conjunto de cláusulas para que o OTTER encontre uma inconsistência no mesmo. É necessário dizer quais regras de inferência serão utilizadas em conjunto com quais técnicas de simplificação. Entretanto, o OTTER possui uma opção, chamada *auto*, que faz uma análise da entrada e decide quais opções utilizar. McCune (2003) sugere que a opção *auto* seja usada como um ponto de partida e que a experiência do usuário com a base de conhecimento é que deverá indicar as melhores configurações para cada caso. Apesar disto, optou-se nesta dissertação por usar a opção *auto* pelos seguintes motivos:

- As opções configuradas no modo *auto* são decididas com base na experiência individual das pessoas que trabalham com o OTTER no *Argonne National Laboratory*. Não se pretende nesta dissertação obter experiência semelhante.
- O modo *auto* é o modo utilizado pelo OTTER na CASC[21] que, por sua vez, é realizada com os problemas da TPTP. Logo, como nesta dissertação também serão utilizados os problemas da TPTP, parece razoável utilizar a opção *auto*.

Entretanto, o modo *auto* do OTTER toma uma decisão inconveniente: com a opção *auto* não é o usuário quem decide a divisão das cláusulas entre os conjuntos *sos* e *usable*. Por exemplo, o OTTER pode decidir colocar em *sos* todas as cláusulas positivas e o restante em *usable*. São dois inconvenientes: em primeiro lugar abre-se mão da completude pois o OTTER não garante que o conjunto *usable* será consistente.[22] Em segundo lugar, agindo desta forma o OTTER desconsidera o fato de que é (bastante) interessante colocar em *sos* as cláusulas que representam a conjectura a ser provada (o que só o usuário pode fazer). Isto é particularmente inconveniente no contexto desta dissertação, já que colocando a conjectura em *sos* estaria-se, na realidade, direcionando o espaço de busca para conter somente as cláusulas cuja derivação tem alguma origem com a conjectura (isto é, esta implementação do conjunto suporte acaba evitando a geração de cláusulas irrelevantes, o que vai ao encontro da hipótese aqui defendida).

---

[20] Os testes foram executados em um ambiente *Linux*.
[21] Competições que acontecem nas conferências CADE (Conference on Automated Deduction).
[22] Como foi dito na seção 3.1, a estratégia do conjunto suporte somente é completa se K-T é consistente, onde K é a base de conhecimento e T? K é o conjunto suporte selecionado (no OTTER *sos*=T e usable=K-T).

Sendo assim, para evitar este comportamento indesejado do modo *auto*, fez-se o seguinte: o OTTER foi executado uma única vez no modo *auto*, coletou-se todas as opções que ele automaticamente gerou e montou-se um arquivo de configurações do OTTER com todas aquelas opções. Este arquivo foi utilizado na execução dos testes ao invés da opção *auto*. É óbvio que, para garantir que as opções decididas fossem boas, a execução inicial do OTTER no modo *auto* foi realizada com um conjunto contendo todas as cláusulas que foram utilizadas nos testes (para que o OTTER tivesse a chance de conhecer todas as cláusulas que seriam utilizadas nos testes).

6.2 Utilização da biblioteca de problemas TPTP como base de conhecimento

A biblioteca de problemas TPTP[23] (Thousands of Problems for Theorem Proving) foi construída a fim de oferecer uma forma padrão para se comparar a eficiência dos programas provadores de teorema que vêm sendo desenvolvidos em diversos institutos de pesquisa do mundo.

A TPTP inclui problemas de diversos domínios como álgebra, geometria, etc. Existem problemas dos mais diversos níveis de dificuldade, inclusive problemas em aberto. Apesar de todas estas características, a TPTP não é uma base de conhecimento e sim uma biblioteca de problemas. Entretanto, após uma busca exaustiva e sem sucesso por uma grande base de conhecimento[24], tomou-se a decisão de utilizar os problemas modelados na TPTP para montar uma base de conhecimento.

Um fato importante de ser notado aqui é que para cada problema da TPTP é dado um conjunto específico de axiomas, hipóteses e conjecturas. Isto é, os provadores de teorema resolvem os problemas da TPTP trabalhando com um conjunto de sentenças possivelmente relevantes umas para as outras. O que quer dizer que a estratégia proposta nesta dissertação não deve ser aplicada diretamente em um problema único da TPTP já que o objetivo é lidar com grandes bases de conhecimento heterogêneas.

Um problema da TPTP é insatisfazível se não existe um modelo que satisfaça a união dos seus axiomas, hipóteses e conjecturas. Neste caso, considerando que os axiomas e hipóteses são consistentes, conclui-se que a conjectura é verdadeira. Caso o contrário, o problema é dito satisfazível e portanto a conjectura é falsa.

---

[23] A versão da TPTP utilizada nos testes desta dissertação foi a v2.5.0. Veja informações em Sutcliffe e Suttner (2004).

[24] Tentou-se por exemplo utilizar a versão aberta do projeto *Cyc* (veja a referência Cycorp), o *OpenCyc*. Entretanto, não se obteve sucesso pois, apesar do *OpenCyc* possuir cerca de 60.000 asserções em sua base de conhecimento, elas são basicamente informações de ontologia de alto nível a partir da qual pode-se criar uma base de conhecimento. Isto é, diferentemente do *Cyc* completo, o *OpenCyc* não é uma base de conhecimento e sim um arcabouço para construção de bases de conhecimento.

Uma forma interessante que foi encontrada para simular o cenário desejado foi tomar a união dos axiomas de diversos problemas da TPTP e montar uma grande base de conhecimento heterogênea. Para resolver um dado problema nesta base de conhecimento, basta tomar as suas hipóteses e conjecturas usando a base completa como o conjunto de axiomas. Sendo assim, percebe-se que não faz sentido comparar os resultados apresentados nesta dissertação com os resultados já publicados do OTTER (ou de qualquer outro provador de teoremas).

Seguindo esta idéia, construiu-se duas bases de conhecimento para os testes conforme está descrito na Tabela 1. Percebe-se que foram selecionados axiomas somente dos problemas insatisfazíveis já que nesta dissertação não será estudado o impacto da estratégia proposta quando a sentença desejada não é consequência lógica da base. Também só foram selecionados os problemas da TPTP expressos na forma normal conjuntiva. Estas bases podem ser obtidas em <http://www.ime.usp.br/~joselyto/mestrado>.

Base	Descrição	Tamanho
Base 1	Axiomas dos problemas insatisfazíveis dos domínios "Teoria dos Conjuntos", "Geometria" e "Teoria das Organizações"	1029 cláusulas
Base 2	Axiomas dos problemas insatisfazíveis dos domínios "Teoria dos Grupos", "Processamento de Linguagem Natural", "Cálculos de Lógica", "Teoria dos Conjuntos", "Geometria" e "Teoria das Organizações"	1781 cláusulas

Tabela 1 - Bases de conhecimento construídas para os testes

Vale a pena notar aqui que se os axiomas dos problemas da TPTP forem simplesmente coletados sem nenhum tratamento obter-se-á bases de conhecimento muito maiores que as que foram apresentadas na Tabela 1. Isto ocorre porque os axiomas se repetem bastante entre os problemas da TPTP. Para evitar isto, foi criado um programa que gera a base de conhecimento sem repetição trivial de axiomas.

Outro ponto importante de ser notado é que as bases foram construídas respeitando a ordem em que os axiomas aparecem nos problemas da TPTP. Isto é, os axiomas foram guardados nas bases na mesma sequência em que eles aparecem na TPTP. Esta característica pode ter impacto nos resultados da aplicação da estratégia proposta, entretanto esta análise não foi realizada nesta dissertação, ficando como possível trabalho futuro (veja discussão na Seção 5.2.6).

Entretanto, argumenta-se aqui que esta sequência implícita na base de conhecimento é um fato que deve ocorrer em geral na construção de bases de conhecimento, pois é mais fácil pensar que os axiomas serão inseridos em uma base de conhecimento seguindo uma certa ordem natural do que aleatoriamente. Isto é, esta sequência possivelmente existente nas bases de conhecimento pode ser explorada positivamente. Percebe-se que se, no passo 7 do algoritmo da Figura 5, as sentenças relevantes forem retornadas na sequência em que elas foram inseridas na base, então as sentenças serão utilizadas pelo algoritmo de inferência nesta sequência. Vale a pena enfatizar que a implementação realizada nesta dissertação se comporta desta forma.

6.3 Descrição dos testes realizados

Para a realização dos testes, foi adotada a abordagem mais simples que considera o tamanho de um conjunto de sentenças como sendo a quantidade de sentenças do mesmo. Além disto, a constante *equal*, que representa o predicado de igualdade nos problemas da TPTP, foi considerada uma constante muito frequente, isto é, ela não foi considerada na definição de relevância por sintaxe adotada.[25]

Foram realizados dois conjuntos de testes conforme descrito na Tabela 2.[26] A coluna "Sequência" indica a sequência de execução do programa RR-OTTER em função da quantidade de sentenças relevantes consideradas em cada passo. Isto é, os números na coluna "Sequência" indicam os tamanhos dos conjuntos K^i utilizados.

A coluna "Limite (s)" indica o limite de tempo utilizado em cada execução do RR-OTTER a partir do qual a execução do programa foi incondicionalmente abortada. Na realidade, deixou-se que o RR-OTTER executasse em cada passo o tempo total (150 segundos) que foi estabelecido para resolver cada problema. O objetivo disto foi conhecer o comportamento do RR-OTTER se lhe fosse dado o tempo total para em cada passo (o que permitiu análises importantes como as que são mostradas na Figura 12 e na Figura 13). Então, o limite de tempo da coluna "Limite (s)" foi aplicado analisando-se o tempo que o RR-OTTER gastou em cada problema: se este tempo excedeu por exemplo 12,5 segundos (caso da massa "Testes 1"), então considerou-se que o RR-OTTER não conseguiu resolver o problema naquele passo.

---

[25] Foram realizados alguns testes considerando o símbolo de igualdade como uma constante qualquer nos quais o uso da estratégia proposta acabou não trazendo benefícios, de onde se tirou a conclusão de que é melhor tratar a igualdade de modo especial (como se ela fosse um conectivo lógico que não interfere na relevância entre as sentenças).

[26] A idéia de ter duas bases de conhecimento e dois conjuntos de testes foi uma tentativa de minimizar o risco de ter-se escolhido um domínio peculiar da TPTP para o qual a estratégia proposta fosse ou coincidentemente muito aderente ou totalmente inaplicável.

Conjunto	Descrição	Base	Tamanho	Sequência	Limite (s)
Testes 1	Problemas insatisfazíveis do domínio "Teoria dos Conjuntos".	Base 1	285 problemas	25, 50, 100, 200, 250, 300, 350, 400, 450, 500, 550 e 600.	12,5
Testes 2	Problemas insatisfazíveis do domínio "Teoria dos Grupos".	Base 2	458 problemas	25, 50, 100, 200, 250, 300, 350, 400, 450 e 500.	15

Tabela 2 - Conjuntos de testes realizados

A fim de obter um parâmetro de comparação, o programa original OTTER foi disparado isoladamente para cada problema em cada um dos conjuntos de testes realizados. Neste caso, o limite de tempo para cada execução foi 150 segundos, que é exatamente igual à quantidade de tempo total que cada problema utilizou com a versão RR-OTTER (para verificar isto, basta somar os tempos que cada execução da sequência utilizou). Desta forma, foi possível traçar uma comparação justa entre os resultados obtidos com o OTTER original contra o que foi conseguido com uma sequência de execuções do RR-OTTER.

Com o propósito único de exemplificar e facilitar o entendimento dos testes realizados, é apresentada a Tabela 3 que lista os resultados de um pequeno subconjunto dos problemas do conjunto "Testes 1".

Problema	Tempo OTTER (s)	Tempo RR-OTTER (s)	Máximo Sentenças Relevantes na Execução do RR-OTTER que Obteve Sucesso
SET003-1	-	13,06	50
SET018-1	-	63,21	300
SET024-6	0,76	12,96	50
SET031-3	0,71	0,45	25
SET183-6	-	98,46	400
SET296-6	0,74	38,08	200

Tabela 3 - Resultados de um pequeno subconjunto dos problemas da TPTP que foram testados

A Tabela 3 mostra, por exemplo, que o programa original OTTER não conseguiu resolver o problema "SET003-1" da TPTP dentro do limite de tempo preestabelecido (150 segundos), enquanto que o programa RR-OTTER conseguiu resolvê-lo em 13.06 segundos. Neste caso, a solução foi encontrada na segunda execução do RR-OTTER quando se tentou encontrá-la considerando as 50 sentenças mais relevantes. Percebe-se que, na realidade, foram consumidos 12,5 segundos tentando-se encontrar a solução com as 25 sentenças mais relevantes, sendo que, ao ultrapassar este limite de tempo preestabelecido, a execução foi abortada, passando-se para a próxima tentativa considerando-se as 50 sentenças mais relevantes. Nesta segunda tentativa, a solução foi encontrada em 0,56 segundos.

Os problemas "SET003-1", "SET018-1" e "SET183-6" são exemplos de casos em que a aplicação da estratégia proposta nesta dissertação foi bastante eficaz. Entretanto, vale a pena notar que existem casos em que acontece o contrário (ver problemas "SET024-6" e "SET296-6"), bem como há casos em que a aplicação da estratégia acaba não influenciando muito o resultado final (ver problema "SET031-3").

6.4 Análise dos resultados obtidos

Como mostra a Tabela 3, a implementação da estratégia de inferência baseada em relevância sintática pode trazer bons resultados em alguns casos, ser indiferente em outros ou até mesmo degradar o desempenho em certos casos. Isto é, como já foi comentado anteriormente, a heurística implementada é do tipo *Quick and Dirty*. Logo, é interessante analisar os resultados médios alcançados em uma grande massa de testes.

Por este motivo, a estratégia foi testada em dois conjuntos de testes relativamente grandes, resolvendo mais de 700 problemas de dois domínios distintos (ver Tabela 1 e Tabela 2).

Os resultados destes testes estão resumidos na Tabela 4. A coluna "Soluções Encontradas" indica a quantidade de soluções que foram encontradas dentro do limite de tempo preestabelecido (150 segundos). O tempo médio exibido na coluna "Tempo Médio 1(s)" é a média dos tempos gastos em todos os problemas, incluindo os problemas para quais a solução não foi encontrada. Já a coluna "Tempo Médio 2(s)" mostra o tempo médio considerando apenas os problemas que a versão original do OTTER conseguiu resolver.

Conjunto	Problemas	Soluções Encontradas		Tempo Médio 1 (s)		Tempo Médio 2 (s)	
		OTTER	RR-OTTER	OTTER	RR-OTTER	OTTER	RR-OTTER
Testes 1	285	111	196	93	61	3,04	6,9
Testes 2	458	212	258	128	138	11,6	23,07

Tabela 4 - Resumo dos resultados dos testes

Percebe-se que a aplicação da estratégia foi mais eficiente no conjunto de testes "Testes 1", onde a versão RR-OTTER conseguir resolver 77% a mais de problemas que a versão original, em um tempo médio 35% menor. Já no conjunto "Testes 2", o RR-OTTER resolveu apenas 22% a mais de problemas em um tempo médio equivalente ao OTTER (apenas 8% menor).

É fácil notar que o tempo médio geral obtido pelo RR-OTTER só não foi menor ainda porque, em média, quando o OTTER encontrou uma solução ele o fez bem mais rápido que o RR-OTTER, como mostra os números na coluna "Tempo Médio 2(s)". Entretanto, isto pode ser explicado, pelo menos em parte, pelo fato de que, nos testes realizados, o estado da inferência não foi armazenado e reaproveitado entre uma execução e outra do mesmo problema. Por exemplo, no caso do problema "SET183-6", cujos resultados estão apresentados na Tabela 3, foram consumidos 87.5 segundos tentando-se encontrar a solução em 7 execuções sem sucesso (considerando gradativamente as 25, 50, 100, 200, 250, 300 e 350 sentenças mais relevantes), até que a solução foi encontrada com as 400 sentenças mais relevantes em 10.96 segundos (veja detalhes na Tabela 6 do Apêndice B).

Ora, é esperado que haja uma perda de tempo entre uma execução e outra do RR-OTTER, pois esta é exatamente a estratégia: tentar encontrar a solução com poucas sentenças e acrescentar sentenças gradativamente até encontrar a solução. Entretanto, é possível minimizar esta perda guardando-se o estado da inferência de uma execução abortada para ser utilizado na próxima tentativa (como sugere o algoritmo da Figura 5). Como o código original do OTTER não ofereceu condições convenientes para se fazer esta implementação, a cada execução do RR-OTTER, todo o processo de inferência é iniciado do ponto zero. Isto explica porque, para os problemas que o OTTER conseguiu resolver, o tempo médio gasto pelo RR-OTTER foi maior que o tempo gasto pela versão original.

Uma análise interessante é saber a quantidade de soluções que foi encontrada em cada etapa da sequência de chamadas do RR-OTTER. Esta informação é mostrada na linha "Novas soluções encontradas" da Tabela 5. Por exemplo, com o conjunto de testes "Testes 1", foram encontradas 196 soluções pelo RR-OTTER. Destas 196 soluções, 86 foram encontradas no primeiro passo (considerando apenas as 25 sentenças mais relevantes), 43 no segundo passo (50) e assim por diante.

A outra linha exibida na Tabela 5, "Soluções encontradas isoladamente", mostra a quantidade de soluções que seriam encontradas executando-se o RR-OTTER isoladamente com cada quantidade de sentenças relevantes consideradas com o limite de tempo de 150 segundos. Por exemplo, executando-se o RR-OTTER utilizando as 200 sentenças mais relevantes, são encontradas 168 soluções no conjunto de testes "Testes 1". É interessante notar que neste caso, o passo com 200 sentenças acrescentou apenas 22 soluções na estratégia gradativa.

Note também que a partir de 450 sentenças, nenhuma solução extra foi encontrada bem como o total de soluções encontradas começou a decrescer (tanto na massa "Testes 1" como na "Testes 2"). Ora, isto comprova a hipótese desta dissertação de que à medida que cresce o conjunto de sentenças a ser considerado, o algoritmo de inferência, apesar de possuir mais informações, não consegue lidar com as mesmas dada a limitação de recursos disponíveis, podendo até mesmo ter seu desempenho degradado.

		Quantidade de Sentenças mais Relevantes Consideradas											
		25	**50**	**100**	**200**	**250**	**300**	**350**	**400**	**450**	**500**	**550**	**600**
Testes 1	Novas soluções encontradas	86	43	21	22	2	5	7	10	0	0	0	0
	Soluções encontradas isoladamente	89	128	150	168	152	143	151	131	115	110	115	113
Testes 2	Novas soluções encontradas	74	46	106	21	25	2	2	2	0	0	-	-
	Soluções encontradas isoladamente	78	100	225	256	252	246	244	258	252	239	-	-

Tabela 5 - Desempenho do RR-OTTER nos diversos estágios

Outro fato interessante a ser notado é que as execuções do RR-OTTER com grandes quantidades de sentenças não acrescentaram muitas soluções no total de soluções que a estratégia gradativa conseguiu gerar. Por exemplo, no caso dos "Testes 1", a partir de 450 sentenças, nenhuma nova solução foi acrescentada (perceba que o RR-OTTER consegue encontrar bastante soluções nestes casos, mas todas elas já haviam sido encontradas nos passos anteriores com menos sentenças sendo consideradas). O gráfico da Figura 12 mostra o percentual de novas soluções encontradas em cada execução do RR-OTTER na massa "Testes 1".

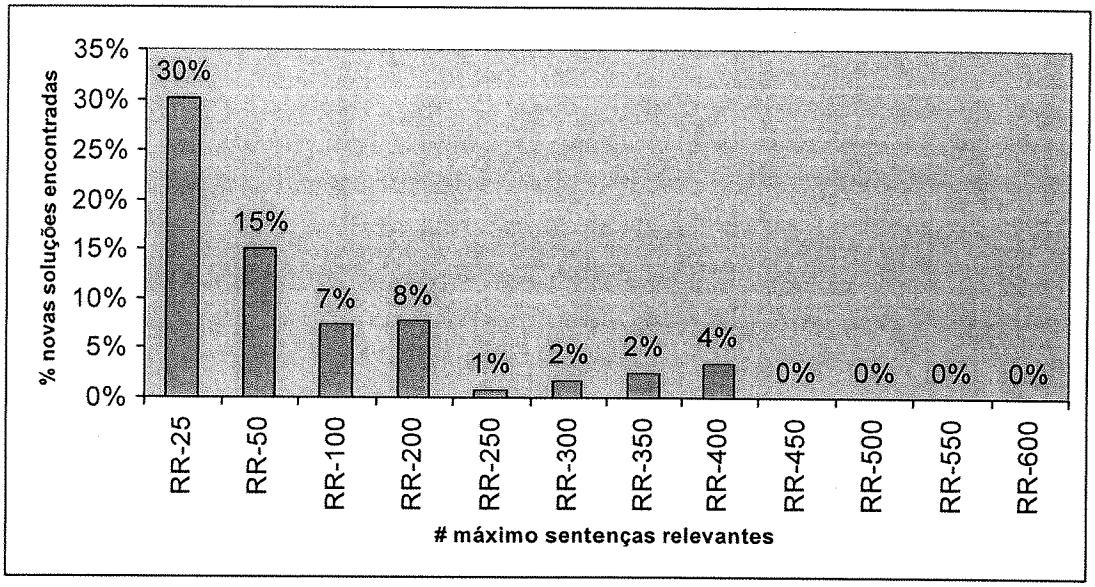

Figura 12 - % de novas soluções encontradas em cada execução do RR-OTTER em relação ao total de soluções encontradas (massa "Testes 1")[27]

A análise do gráfico acima requer bastante cuidado, pois nos testes realizados, o limite de tempo dado (150 segundos) é relativamente curto considerando que podem haver problemas difíceis que talvez fossem solucionados em dias com uma grande de quantidade de sentenças (e que talvez pudessem nunca ser solucionados com uma quantidade menor de sentenças). Então, neste ponto, vale a pena notar que o objetivo desta dissertação é propor uma estratégia que seja eficaz para produzir respostas rápidas para problemas simples, evitando que o mais trivial dos problemas se torne algo complexo por causa do excesso de informação.

O gráfico da Figura 13 compara a taxa de acertos (quantidade de soluções encontradas pelo total de problemas) da massa "Testes 1" das diversas execuções do RR-OTTER. O gráfico também mostra a taxa de acerto do programa original do OTTER e o resultado conseguido com a implementação do algoritmo progressivo.

[27] No eixo "x" do gráfico, RR-*n* indica uma execução do RR-OTTER considerando no máximo *n* cláusulas relevantes.

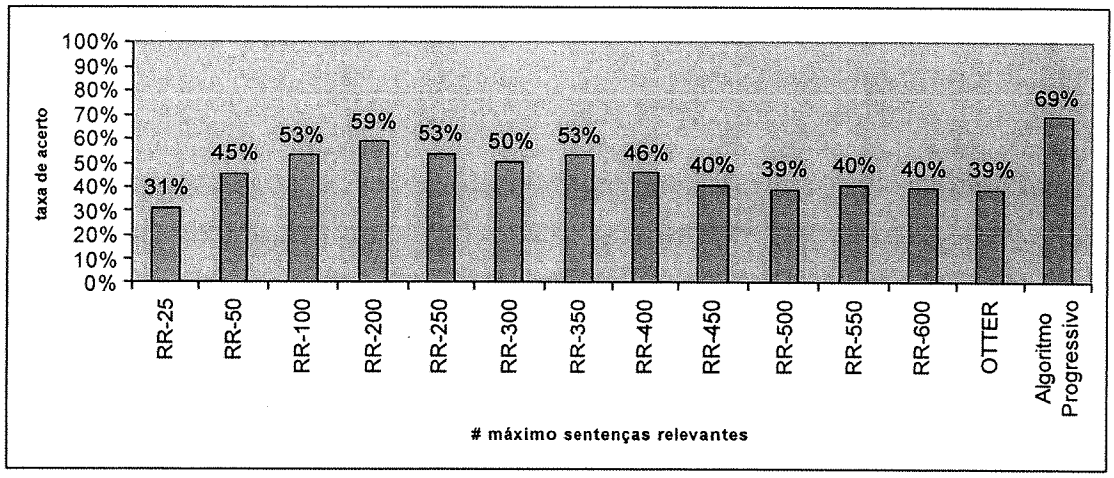

Figura 13 - Comparação da taxa de acerto de cada execução do RR-OTTER, do OTTER (original) e do algoritmo progressivo proposto (massa "Testes 1")

Note que o algoritmo progressivo conseguiu resolver 69% dos problemas mas que nenhuma execução isolada do RR-OTTER atingiu esta patamar. É óbvio que isto ocorre porque não é verdade que o conjunto das soluções encontradas no i-ésimo passo está contido no conjunto das soluções que foram encontradas no $(i+1)$-ésimo passo. Vale a pena notar que o conjunto de soluções encontrado pelo algoritmo progressivo é a união dos conjuntos das soluções encontradas nas diversas execuções do RR-OTTER.

Pode-se comparar também a quantidade de cláusulas geradas no decorrer do processo de inferência em cada execução do RR-OTTER. Esta análise é importante já que um dos benefícios da estratégia proposta é exatamente reduzir a quantidade de cláusulas geradas no processo. O gráfico da Figura 14 mostra evolução da quantidade de cláusulas geradas e tempo gasto à medida que foram acrescentadas cláusulas no conjunto de sentenças relevantes, exibindo também os resultados do OTTER (com o conjunto completo de cláusulas=1029).

O gráfico se refere ao problema SET044-5 da TPTP que ilustra muito bem um caso em que a estratégia progressiva traz benefícios: trata-se de um problema simples cuja solução pode demorar ser encontrada quando a base de conhecimento é muito grande Note a diferença entre a quantidade de cláusulas geradas pelo OTTER (=3.572) e pelo RR-OTTER considerando 25 cláusulas relevantes (=29). Ora, esta diferença (3.543) correspondem a dáusulas geradas desnecessariamente devido ao excesso de cláusulas disponíveis para a inferência. O que esta-se defendendo nesta dissertação é que este deve ser um caso freqüente em se tratando de inferência em grandes bases de conhecimento (e que portanto a heurística apresentada deve trazer benefícios no geral).

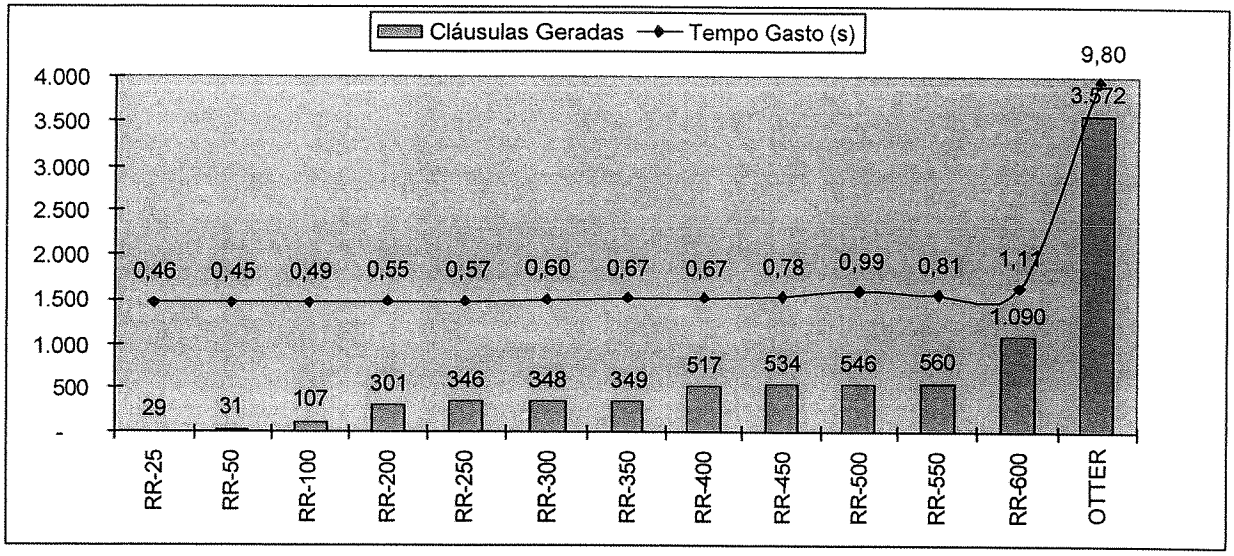

Figura 14 - Comparação da quantidade de cláusulas geradas em cada execução do RR-OTTER e do OTTER (problema SET044-5 da TPTP)

Um fato que pode aumentar o tempo médio gasto pelo RR-OTTER é uma má escolha da sequência de execuções do algoritmo de inferência (isto é, uma implementação ruim da função *H*). Por exemplo, se a sequência começa com um conjunto de sentenças muito pequeno, a probabilidade de falha é maior e, portanto, o RR-OTTER vai gastar mais tempo com tentativas sem sucesso. Para se ter uma idéia deste impacto, se o conjunto de testes "Testes 1" tivesse começado com 50 sentenças ao invés de 25, o tempo médio obtido pelo RR-OTTER seria 3,1 segundos ao invés de 6,9 (veja na linha "Soluções encontradas isoladamente" da Tabela 5 que o passo com 25 sentenças foi o que conseguiu encontrar o menor número de soluções).

Por último, vale a pena ressaltar que em ambos os conjuntos de testes, conseguiu-se aumentar a quantidade de soluções encontradas dentro do limite de tempo preestabelecido, bem como diminuir o tempo médio para encontrar as soluções (em comparação com os resultados obtidos com a versão original do OTTER).

7 Conclusões e Trabalhos Futuros

O objetivo desta dissertação foi verificar a eficácia da aplicação do conceito de relevância entre as sentenças de uma base de conhecimento a fim de melhorar a eficiência dos processos de inferência nesta base. Foi dada uma noção de relevância baseada na sintaxe das fórmulas de uma lógica de predicado de primeira ordem. Esta noção pode ser aplicada diretamente em qualquer base de conhecimento representada em LPPO pois não faz uso de nenhuma estrutura extra-lógica.

Foi apresentado um algoritmo para inferência que implementa uma estratégia de busca na qual procura-se pela solução do problema primeiramente com as sentenças mais relevantes, aumentando-se a quantidade de sentenças consideradas gradativamente até que a solução seja encontrada ou algum limite de recursos seja atingido. Foi demonstrado que, com a noção de relevância sintática apresentada, é garantido que, a cada passo, o algoritmo de inferência não vai gerar sentenças mais irrelevantes do que aquelas que estão sendo consideradas naquele momento. Isto é, provou-se que a heurística proposta realmente restringe o espaço de busca do algoritmo de inferência.

Entretanto, como esperado, a complexidade do algoritmo continua dependendo da complexidade da inferência da linguagem utilizada para representar a base de conhecimento. Então, para comprovar a eficiência da estratégia proposta na prática, foi realizada uma grande quantidade de testes com o provador de teoremas OTTER resolvendo os problemas da biblioteca TPTP.

Um dos resultados interessantes destes testes foi que 88% das soluções encontradas foram obtidas considerando não mais que 20% da base de conhecimento, o que sustenta a hipótese de que, em geral, existe uma boa chance de se encontrar uma solução considerando um conjunto menor de sentenças mais relevantes. De qualquer forma, esta continua sendo uma hipótese que depende muito da base de conhecimento e dos tipos de problemas que se deseja solucionar com a mesma. Não foi apresentada nesta dissertação uma prova de que realmente é mais provável que a solução seja encontrada em meio às sentenças mais relevantes. Apesar disto, a intuição de que este fato é verdadeiro é forte e foi bem sustentada pelos resultados dos testes realizados pois a maior parte das soluções foi encontrada nos primeiros passos da execução do algoritmo.

As seguintes tarefas podem fazer parte de uma eventual continuação do trabalho até aqui realizado:

- ? Alterar o código do OTTER de forma que seja possível reutilizar o "estado" da inferência entre uma execução e outra do mesmo problema. Na realidade, seria interessante fazer isto a partir do trabalho de Ribeiro (2003). Tem-se a forte intuição de que com esta implementação pode-se obter resultados ainda melhores do que os que foram obtidos aqui.
- ? Repetir os testes com outros provadores de teorema. Seria interessante, por exemplo, utilizar o *Vampire* (ver Riazanov(2003)) que tem sido o vencedor das últimas edições do CASC. Também seria interessante utilizar um provador de teorema baseado em outro princípio (diferente da resolução), por exemplo baseado em *Tableaux*.
- ? Realizar testes com outras bases de conhecimento. Por exemplo, seria muito interessante testar as hipóteses desta dissertação com a versão completa do *Cyc* (foi feita uma tentativa sem sucesso com a versão aberta, o *OpenCyc*). Um fato interessante no *Cyc* é que existe uma chance de se obter informações de relevância com base na sua estrutura de microteorias (isto é, com o *Cyc* seria possível obter uma noção de relevância mais "significativa" do que a baseada em sintaxe utilizada nesta dissertação).
- ? O algoritmo apresentado aumenta a quantidade de sentenças relevantes a ser considerada em cada passo. Isto não implica que as sentenças consideradas em um determinado passo são necessariamente mais relevantes do que aquelas consideradas no passo seguinte. Seria interessante testar outras estratégias nas quais o conjunto de sentenças utilizados no i-ésimo passo não esteja necessariamente contido no conjunto utilizado no $(i+1)$-ésimo passo.
- ? Outra alteração que pode ser estudada no algoritmo proposto é a estratégia de seleção entre sentenças com o mesmo grau de relevância. No algoritmo apresentado, as sentenças são selecionadas na ordem em que elas são lidas da base. Outra estratégia seria, por exemplo, selecioná-las de forma aleatória.
- ? Para diminuir a frequência de sentenças com o mesmo grau de relevância (o que causa os efeitos descritos nos dois últimos itens), poderia-se refinar um pouco mais a noção de relevância baseada em sintaxe considerando a quantidade de constantes que as sentenças compartilham. Por exemplo se ? e ? compartilham duas constantes, e ? e ? compartilham uma única constante, poderia-se então considerar ? e ? mais relevantes entre si do ? e ?.

Apêndice A: Documentação da Implementação do RR-OTTER

Este apêndice descreve as alterações que foram realizadas no código fonte do OTTER a partir das quais se obteve o programa chamado nesta dissertação de RR-OTTER (*Relevant Reasoning* OTTER). O objetivo do RR-OTTER é fazer com que o programa receba como parâmetro um inteiro n e faça com sejam utilizadas apenas as n cláusulas mais relevantes para a conjectura que se deseja verificar de acordo com a noção de relevância por sintaxe apresentada nesta dissertação.

O apêndice possui duas seções, a primeira delas apresenta as estruturas de dados e funções do OTTER necessárias para se entender as alterações que foram feitas para se obter o RR-OTTER, enquanto a segunda seção descreve as alterações propriamente ditas.

Como o OTTER foi escrito em C, algumas notações e conceitos comuns nesta linguagem de programação serão mencionados livremente neste apêndice (sem a apresentação de definições).

A.1 Algumas estruturas de dados importantes do OTTER

Nesta seção serão apresentadas as estruturas de dados e funções relevantes para o entendimento das alterações que foram realizadas no código do OTTER. Na realidade, não serão descritos todos os detalhes das estruturas aqui abordadas, somente aqueles que também forem considerados relevantes para o propósito de entender o RR-OTTER. As estruturas do OTTER serão replicadas neste texto grafando as negrito os detalhes de interesse.

Em prol da didática, as explicações apresentadas para as estruturas vão assumir a existência de um determinado objeto instanciado daquela estrutura (este objeto estará indicado no texto logo após a estrutura). Os membros das estruturas são explicados utilizando-se a notação *o.m* onde *m* é um membro da estrutura do objeto *o* (não será feita distinção na notação em relação ao fato do objeto ser um objeto propriamente dito ou um ponteiro para um objeto, neste último caso a notação usual seria *o->m*).

A.1.1 Um diagrama (não completo) das estruturas do OTTER

O diagrama abaixo ilustra uma (pequena) parte das estruturas de dados do OTTER. Os nomes dentro das caixas correspondem a estruturas (*struct* da linguagem C) enquanto os rótulos das setas representam membros existentes na estrutura de onde parte a seta.

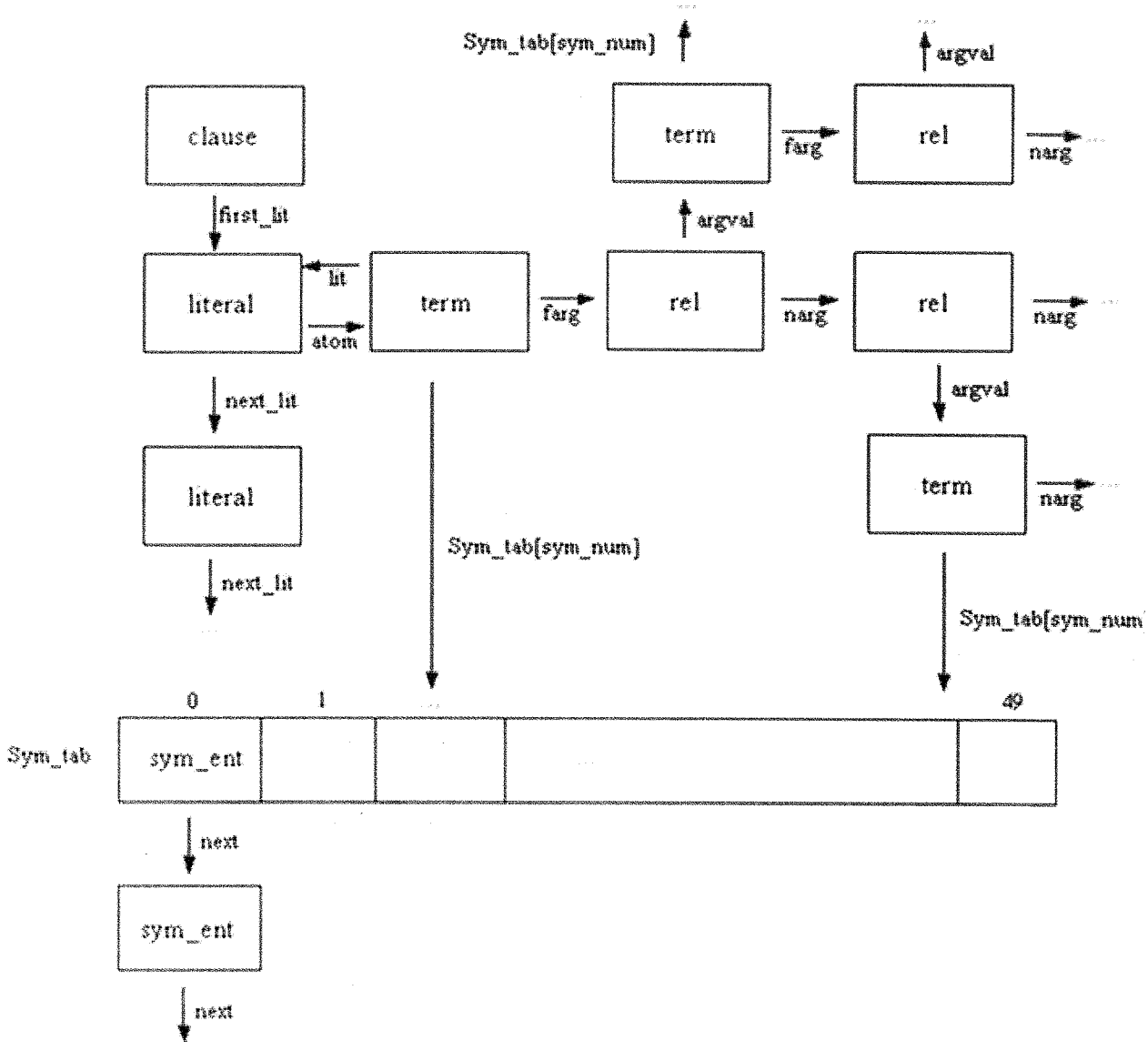

Figura 15- Um diagrama (não completo) das estrutura de dados do OTTER

As subseções seguintes vão discutir um pouco cada das estruturas acima. Entretanto, aqui está uma breve introdução: uma cláusula possui uma lista de literais associados. Cada literal possui um átomo associado. Um átomo pode possuir uma lista de termos associados. Existe uma tabela de símbolos genérica na qual são armazenados todos os símbolos do vocabulário da linguagem utilizada.

A.1.2 As listas de cláusulas do OTTER

O OTTER mantém várias listas de cláusulas (*Usable*, *Sos*, *demodulators*, *hints*, *passive* e *hot*). Para os objetivos propostos aqui, é suficiente que se conheça apenas a *Usable* e a *Sos* e que se pense nas mesmas da seguinte forma: no início da execução do programa *Usable* vai conter as cláusulas que representam a base de conhecimento e *Sos* vai conter as cláusulas que representam a conjectura que se deseja provar.

Na realidade, o OTTER não exige que a *Usable* e a *Sos* sejam conforme descrito acima. O OTTER simplesmente procura por uma inconsistência no conjunto *Usable* ? *Sos*. O usuário é quem deve decidir que cláusulas colocar em *Usable* e *Sos*.

A.1.3 A tabela de símbolos genérica do OTTER

O OTTER armazena todos os símbolos do vocabulário (constantes de predicado, indivíduo, variáveis, funções, conectivos lógicos, etc) da linguagem utilizada em um vetor, declarado globalmente, chamado *Sym_tab*. Os elementos desta tabela sempre são acessados através de um número de identificação, o *sym_num* presente na estrutura *term*.

Sym_tab é um vetor de objetos da seguinte estrutura:

```
struct sym_ent {  /* symbol table entry */
   struct sym_ent *next;
   int sym_num;          /* unique identifier */
   int arity;           /* arity 0 for constants, variables */
   int lex_val;          /* can be used to assign a lexical value */
   int eval_code;        /* identifies evaluable functors ($ symbols) */
   int skolem;           /* identifies Skolem constants and functions */
   int special_unary;    /* identifies special unary symbol for lex check */
   int lex_rpo_status;   /* status for LRPO */
   char name[MAX_NAME];   /* the print symbol */
   int special_op;  /* for infix/prefix/postfix functors */
   int op_type;     /* for infix/prefix/postfix functors */
   int op_prec;     /* for infix/prefix/postfix functors */
   };
sym_ent s;
```

O nome do símbolo *s* é obtido em *s.name*.

Se o símbolo *s* é uma constante de predicado ou função, *s.arity* indica a aridade de *s*.

O vetor *Sym_tab* possui tamanho fixo (=*SYM_TAB_SIZE*) que no código fonte original do OTTER é declarado como 50. O OTTER utiliza o membro *next* da estrutura *sym_ent* para poder acomodar uma quantidade acima de 50 símbolos. Por exemplo, para inserir o 51o. símbolo no vetor *Sym_tab*, por exemplo *s'*, o OTTER faz o seguinte:

```
s'.next = Sym_tab[0]
Sym_tab[0]=s'.
```

A.1.4 Representação das cláusulas

O OTTER trabalha somente com cláusulas, que são disjunções de literais. Uma cláusula representada no OTTER através da seguinte estrutura:

```
struct clause {
   struct ilist *parents;
   struct g2list *multi_parents;  /* for proof-shortening experiment */
   struct list *container;
   struct clause *prev_cl, *next_cl;  /* prev and next clause in list */
   struct literal *first_lit;
   int id;
   int pick_weight;
   struct cl_attribute *attributes;
   short type;          /* for linked inf rules */
   unsigned char bits;  /* for linked inf rules */
   char heat_level;
   };
clause c;
```

As cláusulas do OTTER são armazenas em listas (*Usable*, *Sos*, etc.) sendo que *c.container* aponta para a lista na qual a cláusula *c* está armazenada. Vale a pena notar que a implementação do OTTER assume que uma determinada cláusula está apenas em uma única lista.

c.*prev_cl* e c.*next_cl* apontam respectivamente para as cláusula anterior e seguinte na lista onde *c* está armazenada.

c.first_lit aponta para o primeiro literal que ocorre na cláusula *c*.

A.1.5 Representação dos literais

Os literais são representados através da seguinte estrutura:

```
struct literal {
   struct clause *container;  /* containing clause */
   struct literal *next_lit;
   struct term *atom;
   char sign;
   BOOLEAN target;
   };
literal l;
```

l.container aponta para a cláusula que contém o literal (veja que mesmo que duas cláusulas contenha o mesmo literal, cada uma mantém a sua cópia do mesmo).

l.next_lit aponta para o literal seguinte a *l* na lista de literais da cláusula que contém *l*.

l.atom aponta para uma estrutura que, apesar de se chamar *term* (o que faria alguém pensar se tratar de um termo), representa a fórmula atômica que o literal representa.

l.sign indica se *l* é um literal positivo ou negativo.

A.1.6 Representação de átomos e termos

A estrutura *term*, mostrada abaixo, é utilizada para representar 2 tipos de objetos: um átomo (uma fórmula atômica) ou um termo. Um termo construído a partir de uma constante de função de aridade maior que zero é chamado, na terminologia do OTTER, de termo complexo. Na realidade, o OTTER chama átomos e termos de "termos"[28] o que não será feito no texto deste apêndice (isto é, falar-se-á dos conceitos separadamente).

```
struct term {
   struct rel *farg;        /* subterm list; used for complex only */
   union {           /* term is atom iff (NAME or COMPLEX) && varnum > 0 */
      struct rel *rel;     /* superterm list; used for all except atoms */
    struct literal *lit;  /* containing literal; used for atoms */
    } occ;
   int fpa_id;             /* used to order fpa lists */
   unsigned short sym_num;  /* used for names, complex, and sometimes
vars*/
   VAR_TYPE varnum;         /* used for variables */
   unsigned char type;      /* NAME, VARIABLE, or COMPLEX */
   unsigned char bits;      /* bit flags (see macros.h) */
   };
term a;
```

Suponha que *a* (declarado acima) seja um átomo. Então *a.occ.lit* informa em que literal *a* ocorre (se um mesmo átomo *a* ocorre em dois ou mais literais, cada literal mantém a sua cópia de *a*).

a.type será igual a *COMPLEX* (veja então, de uma forma genérica, que *x.type=COMPLEX* significa que o objeto *x* é um átomo ou um termo complexo).

Pode-se acessar o nome e a aridade da constante de predicado de *a* da seguinte forma: *Sym_tab*[*a.sym_num*].*name* e *Sym_tab*[*a.sym_num*].*arity*.

Os termos que ocorrem em *a* são armazenados em uma lista de elementos da estrutura *rel* mostrada abaixo:

```
struct rel {  /* relations between terms */
   struct term *argval;     /* subterm */
   struct term *argof;      /* superterm */
   struct rel *narg;        /* rest of subterm list */
   struct rel *nocc;        /* rest of superterm list */
   unsigned char path;      /* used in paramod to mark path to into term */
   unsigned char clashable; /* paramodclashability flag */
   };
```

O primeiro termo que ocorre em *a* pode ser acessado através de *a.farg.argval*, o segundo termo através de *a.farg.narg.argval*, o terceiro através de *a.farg.narg.narg.argval* e assim sucessivamente (uma olhada no diagrama da Figura 15 deve ajudar).

---

[28] O construtor do OTTER cometeu este "abuso" de notação aqui muito provavelmente pelo fato da estrutura de um átomo ser bem parecida com a de um termo complexo. Um átomo é uma constante de predicado de aridade *n* seguida de *n* termos enquanto um termo (complexo) é uma constante de função de aridade *n* seguida de *n* termos.

Seja t um termo que ocorre em a. A tabela abaixo lista as possibilidades para t:

t.argval.type	*t* é	**Comentários**
NAME	uma constante de indivíduo	Sym_tab[t.argval.sym_num] retorna o nome da constante de indivíduo.
VARIABLE	uma variável	Sym_tab[t.argval.sym_num] retorna o nome da variável.
COMPLEX	um termo do tipo $f(t')$ onde f é uma constante de função e t' é um outro termo.	? Sym_tab[t.argval.sym_num] retorna o nome da constante de função f. ? t' é armazenado em *t.argval*. Perceba a recursividade implementada aqui já que t' pode ser uma constante de indivíduo, uma variável ou outro termo do tipo $f'(t'')$.

A.2 As alterações realizadas

Nesta seção serão documentas as alterações mais relevantes realizadas no código do OTTER para se obter o programa RR-OTTER. No código fonte, foi inserida uma linha de comentários com a *string* "--Joselyto" antes de cada uma das alterações realizadas.

A.2.1 Algumas alterações simples

Foram realizadas as seguintes alterações simples que não possuem nenhuma relação direta com a implementação do algoritmo proposto neste dissertação:

- ? Foram declaradas no arquivo "header.h", algumas variáveis do tipo *clock_t* com o objetivo de separar o tempo gasto na execução com a preparação das cláusulas e com a inferência propriamente dita.
- ? Foram comentadas algumas linhas de código do OTTER que lançavam mensagens na saída padrão o que estava sobrecarregando a saída padrão desnecessariamente.
- ? Foram lançadas algumas mensagens na saída padrão para indicar, por exemplo, o tamanho das listas construídas (*Usable* e *Sos*), a quantidade de sentenças relevantes considerada e as vezes imprimir o conteúdo de uma lista.

A.2.2 Declaração do parâmetro *MaxRelevantClauses*

O programa modificado, RR-OTTER, recebe um parâmetro extra que é a quantidade máximo de cláusulas "relevantes" que programa deve considerar.

No arquivo "header.h" foi declarado um inteiro, chamado de *MaxRelevantClauses*, cujo o objetivo é armazenar este novo parâmetro (que deve estar em *argv*[1]).

A.2.3 Alterações na estrutura *sym_ent*

Foi acrescentado o seguinte membro na estrutura *sym_ent*:

```
struct glist *ref_cl;
```

Seja *s* um símbolo do tipo *sym_ent*. Se *s* é uma constante (de predicado, indivíduo ou função) então *s.ref_cl* é uma lista com as cláusulas que referenciam *s*.

glist é uma estrutura do OTTER que permite se criar listas de quaisquer objetos.

A.2.4 Alterações na estrutura *clause*

Foram acrescentados os seguintes membros na estrutura *clause*:

```
int HasBeenVisited, HasBeenQueued;
struct glist *ref_const;
```

Os inteiros *HasBeenVisited* e *HasBeenQueued* são utilizados na implementação do algoritmo de seleção das *MaxRelevantClauses* cláusulas mais relevantes.

Se *c* é uma cláusula, então *c.ref_const* é uma lista com as constantes (de predicado, indivíduo e função) que ocorrem em *c*.

A.2.5 Visualizando as estruturas do OTTER como um grafo bipartido

Com as listas que foram acrescentadas às estruturas sym_ent (ref_cl) e clause (ref_const), pode-se pensar nas estruturas do OTTER como um grafo bipartido (ver a Figura 16 abaixo) no qual os vértices são cláusulas ou constantes (de predicado, indivíduo ou função) com as arestas ligando ou uma cláusula às constantes que ocorrem na mesma ou uma constante às cláusulas que a referenciam.

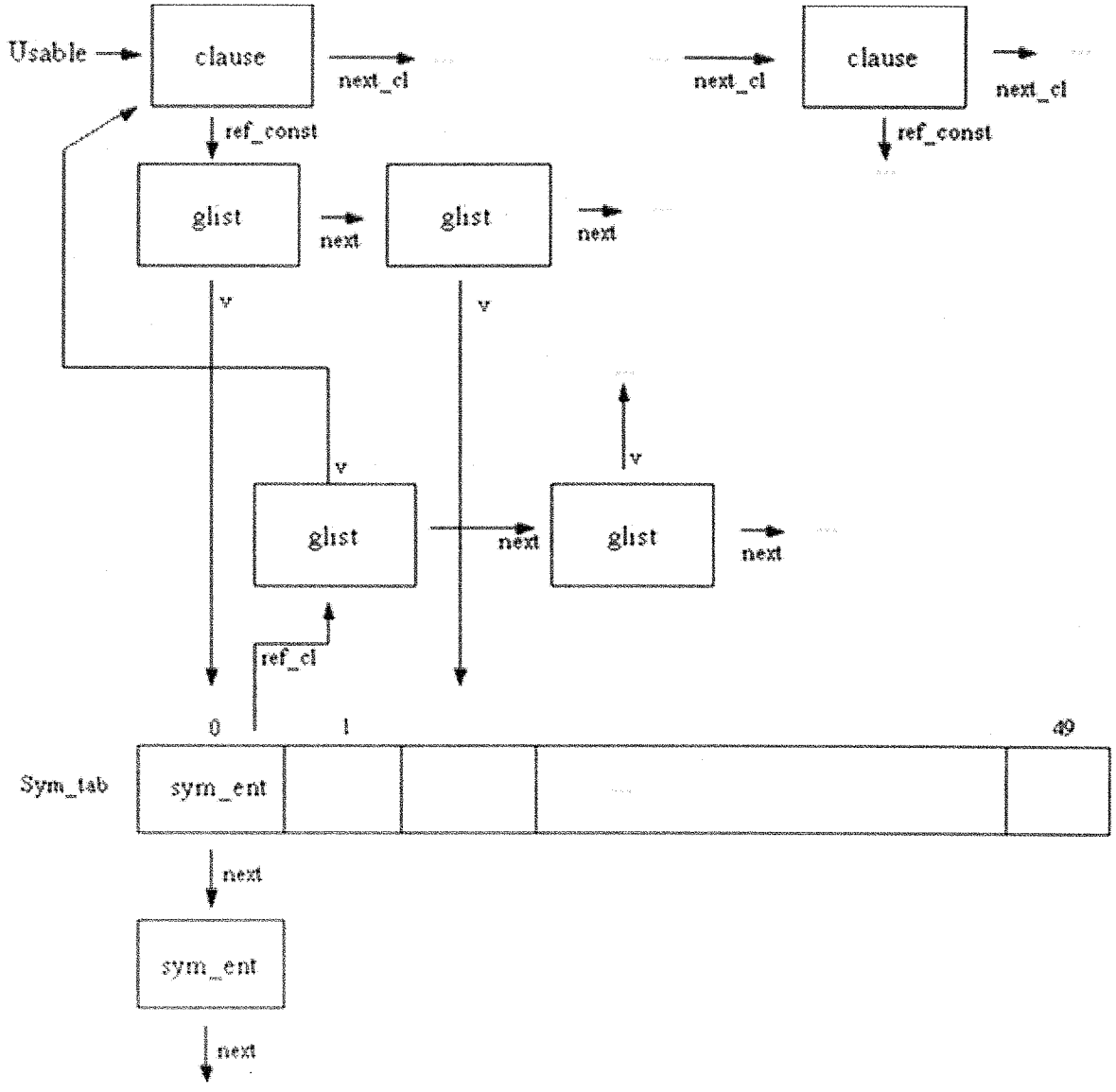

Figura 16 - Alterações nas estruturas *clause* (*ref_const*) e *sym_ent* (*ref_cl*)

A.2.6 A implementação do algoritmo de seleção das cláusulas relevantes

A função *select_relevant_clauses* (arquivo "misc.c") implementa o algoritmo de seleção das cláusulas relevantes. A lista de cláusulas *Usable*, utilizada no laço principal do OTTER, é guardada em uma lista auxiliar chamada de *InputUsableList*. A lista *Usable* é então "esvaziada" e é preenchida novamente com no máximo *MaxRelevantClauses* cláusulas.[29]

A chamada para a *select_relevant_clauses* é feita na função *read_all_input* (arquivo "misc.c") logo após as listas do OTTER (*Usable*, *Sos*, etc.) terem sido inicializadas, isto é, carregadas dos arquivos de entrada para a memória, e antes de qualquer processamento (por exemplo demodulação) que o OTTER eventualmente possa fazer nas cláusulas.

O algoritmo implementado na *select_relevant_clauses* pode ser visto como uma busca em largura em um grafo bipartido formado pelas cláusulas e constantes (de predicado, indivíduo e função) conforme é descrito na subseção anterior. Para a implementação desta busca em largura foi necessário declarar uma estrutura de dados do tipo fila (veja a *struct queue* e as funções *QUEUE** declaradas no próprio arquivo "misc.c").

Tentou-se, sem sucesso, montar estas duas estruturas extras em tempo de inicialização do OTTER, isto é, no momento em que as cláusulas são lidas dos arquivos de entrada para a memória. Logo, a solução adotada foi, antes de entrar no laço da busca em largura, chama-se uma função (a *atualiza_sym_ref_cl*) que preenche os membros *ref_const* e *ref_cl* das cláusulas e constantes. Certamente este fato trouxe uma sobrecarga extra para o algoritmo como um todo o que, no entanto, não influenciou substancialmente os resultados obtidos.

---

[29] Vale a pena lembrar que assume-se aqui que *Usable* contém as cláusulas que representam a base de conhecimento que o OTTER vai utilizar ou , na terminologia mais usual de um provador de teoremas, *Usable* contém as cláusulas que representam os axiomas do problema a ser resolvido.

Por se tratar da principal codificação realizada no RR-OTTER, abaixo esta listado o código fonte da função *select_relevant_clauses*:

```
void select_relevant_clauses() {
  Q q;
  ptrClause cl, cl2;
  struct sym_ent *c;
  struct glist *cl_list, *c_list;
  struct term *t;
  int  bSizeLimitExceeded = 0;

 // Monta as listas ref_cl (e ref_const) que foram acrescentadas à estrutura sym_ent
 // (clause)
 atualiza_sym_ref_cl(Usable);
 atualiza_sym_ref_cl(Sos);

 // Guarda a copia original da lista Usable em InputUsableList e cria uma lista vazia que
 // vai conter as novas clausulas relevantes selecionadas
 InputUsableList = Usable;

 // Inicializa a nova lista Usable
 Usable = get_list();
 Stats[USABLE_SIZE] = 0;

 // Inicializa a fila que vai implementar a busca em largura
 q = QUEUEinit();

 // Enfilara cada uma das cláusulas que representa a conjectura (assume-se que estas
 // estão em Sos)
 for (cl = Sos->first_cl; cl!=NULL; cl = cl->next_cl) {
   QUEUEput(q, cl);
   cl->HasBeenQueued = 1;
 }

 // Laço que implementa a busca em largura
 while (!QUEUEempty(q) && !bSizeLimitExceeded) {
  // Desenfileira uma cláusula cl
  cl = QUEUEremove(q);

  // Para cada constante c em cl
  for (c_list = cl->ref_const; c_list!=NULL && !bSizeLimitExceeded; c_list = c_list->next) {
    c = (struct sym_ent *) c_list->v;

    // Para cada clausula cl2 que referencia c
    for (cl_list = c->ref_cl; cl_list!=NULL && !bSizeLimitExceeded; cl_list=cl_list->next) {
      cl2 = (struct clause *) cl_list->v;

      // Se cl2 ainda não foi enfileirada (marcada)
      if (!cl2->HasBeenQueued) {
        QUEUEput(q, cl2);
        cl2->HasBeenQueued = 1;

        // Acrescenta-se cl2 à nova lista Usable
        prepend_cl(Usable, cl_copy(cl2));

        // Se foi atingido o numero maximo de sentencas relevantes
        if (Stats[USABLE_SIZE]>=MaxRelevantClauses)
          bSizeLimitExceeded = 1;
      } // if
    } // for
  } // for
 } // while

 // Destrói a fila
 QUEUEdump(q);
}
```

Vale pena enfatizar que ao se montar a nova lista *Usable* foi necessário criar cópias das cláusulas pelo fato do OTTER assumir que uma determinada cláusula aparece em apenas uma lista.

Apêndice B: Tabela de Resultados dos Testes

Cada uma das seções seguintes apresenta uma tabela completa dos tempos gastos, em segundos, em cada execução do RR-OTTER e do OTTER original para cada problema das massas "Testes 1" e "Testes 2" respectivamente. RR-*n* indica uma execução do RR--OTTER que considerou no máximo *n* cláusulas relevantes. A versão original do OTTER considerou todo o conjunto de cláusulas ("Base 1"=1.029 e "Base 2"=1.781). As células em branco indicam que aquela execução (do RR-OTTER ou do OTTER) não encontrou uma solução para aquele problema no limite de tempo preestabelecido (150 segundos).

Note que na realidade existem outras informações, além do tempo de execução, que poderiam interessar na análise dos testes (por exemplo a quantidade de cláusulas geradas ou a quantidade de "cláusulas dadas" processadas). Por questão de espaço, optou-se por não listar estas informações aqui, entretanto elas podem ser obtidas em <www.ime.usp.br/~joselyto/mestrado>.

B.1 Massa de Testes 1

Problema	RR-25	RR-50	RR-100	RR-200	RR-250	RR-300	RR-350	RR-400	RR-450	RR-500	RR-550	RR-600	OTTER
SET001-1	0,48	0,48	0,50	0,54	0,56	0,62	0,70	0,68	0,70	0,72	0,71	0,82	1,73
SET002-6													
SET003-1		0,56	0,50	0,85	29,46								
SET004-1		0,57	0,54	0,79	29,74								
SET005-1													
SET006-1		0,46	0,49	0,75	120,55								
SET007-1													
SET008-1													
SET009-1		1,14	1,51	112,03									
SET010-1	26,96	39,62											
SET011-1													
SET012-1													
SET012-2													
SET012-3							5,50		33,59				
SET012-4							5,52		31,85				
SET013-1							5,17						
SET013-2							3,36						
SET013-3													
SET013-4													
SET014-2													
SET014-3													
SET014-4													
SET014-6													

Problema	RR-25	RR-50	RR-100	RR-200	RR-250	RR-300	RR-350	RR-400	RR-450	RR-500	RR-550	RR-600	OTTER
SET015-1													
SET015-2													
SET015-3													
SET015-4													
SET016-1						0,84	0,89	0,89	18,64				
SET016-3	0,73												
SET016-6				3,17	130,15								
SET016-7				3,19	131,04								
SET017-6		0,54	0,88	1,25	3,69								
SET017-7		0,55	0,92	1,30	4,06			0,85	22,18				
SET018-1						0,71	0,83						
SET018-6				3,26	133,37								
SET018-7				2,91	132,01			0,76	0,72				
SET019-3	0,46	0,46	0,49	0,50	0,53	0,56	0,67	0,72	0,68	0,92	0,75	0,91	2,88
SET019-4	0,44	0,47	0,45	0,50	0,51	0,59	0,66	0,71	0,74	0,87	0,77	0,88	2,66
SET020-6	0,45	0,48	0,47	0,58	0,62	0,62	0,76	0,69	0,75	0,85	0,81	1,11	4,58
SET020-7	0,41	0,47	0,44	0,59	0,60	0,69	0,73	0,76	0,80	0,81	0,80	1,11	4,45
SET021-6	0,45	0,46	0,46	0,59	0,63	0,68	0,76	0,73	0,87	0,82	0,82	1,08	4,65
SET021-7	0,43	0,45	0,47	0,59	0,65	0,67	0,73	0,62	0,62	0,83	0,79	1,07	4,52
SET022-3	0,45	0,43	0,51	0,53	0,57	0,61	0,65	0,61	0,69	0,77	0,63	0,74	0,99
SET023-3	0,46	0,47	0,48	0,51	0,56	0,60	0,62	0,66	0,88	0,79	0,66	0,77	0,96
SET024-3	0,46	0,50	0,52	0,56	0,53	0,63	0,71	0,67	0,95	0,85	0,68	0,79	1,00
SET024-4	0,47	0,52	0,53	0,54	0,59	0,62	0,63	0,61	0,75	0,81	0,70	0,77	1,08
SET024-6		0,46	0,44	0,48	0,59	0,55	0,59	0,64	0,71	0,78	0,77	0,76	1,05
SET024-7		0,46	0,45	0,52	0,55	0,59	0,64	0,67	0,79	0,78	0,64	0,77	1,05
SET025-3	0,43	0,48	0,54	0,56	0,55	0,57	0,62	0,64	0,73	0,77	0,68	0,77	1,09
SET025-4	0,47	0,49	0,51	0,56	0,53	0,57	0,62	0,67	0,78	0,82	0,67	0,72	1,06
SET025-6	0,43	0,43	0,48	0,53	0,57	0,58	0,61	0,62	0,67	0,75	0,70	0,80	1,11
SET025-7	0,46	0,45	0,49	0,52	0,62	0,58	0,61	0,65	0,63	0,76	0,73	0,75	1,15
SET025-8	0,45	0,48	0,52	0,55	0,57	0,54	0,62	0,62	0,64	0,79	0,65	0,74	1,10
SET025-9	0,45	0,47	0,50	0,54	0,56	0,60	0,64	0,72	0,92	0,77	0,65	0,75	1,01
SET027-3	0,45	0,48	0,48	0,51	0,58	0,55	0,62	0,68	0,80	0,84	0,72	0,90	2,70
SET027-4	0,43	0,45	0,48	0,47	0,54	0,59	0,63	0,60	0,64	0,93	0,72	0,88	2,65
SET027-6	0,43	0,42	0,48	0,51	0,51	0,53	0,60	0,67	0,64	0,83	0,84	0,89	3,46
SET027-7	0,45	0,47	0,49	0,50	0,50	0,53	0,57	0,63	0,63	0,84	0,85	0,95	2,99
SET031-3	0,45	0,47	0,49	0,52	0,53	0,55	0,57	0,61	0,57	0,76	0,70	0,71	1,00
SET031-4	0,44	0,50	0,46	0,51	0,49	0,54	0,62	0,66	0,67	0,77	0,69	0,75	1,08
SET041-3	0,46	0,49	0,46	0,53	0,55	0,59	0,67	0,63	0,67	0,99	0,83	1,05	3,69
SET041-4	0,46	0,45	0,47	0,52	0,58	0,64	0,70			0,95	0,85	1,09	3,76
SET042-3					1,19	2,53		0,60	0,68				
SET043-5	0,45	0,47	0,48	0,55	0,55	0,52	0,61	0,74	0,91	0,79	0,66	0,70	1,06
SET044-5	0,46	0,45	0,49	0,55	0,57	0,60	0,67	0,67	0,78	0,99	0,81	1,11	9,80
SET045-5	0,50	0,47	0,48	0,50	0,56	0,57	0,56	0,65	0,70	0,81	0,67	0,69	1,07
SET046-5	0,44	0,49	0,48	0,49	0,56	0,56	0,59			0,77	0,68	0,70	1,18
SET047-5	0,48	0,48						0,62	0,64				
SET050-6	0,48	0,48	0,59	0,56	0,61	0,60	0,64	0,61	0,60	0,77	0,70	0,77	0,79
SET051-6	0,47	0,47	0,57	0,57	0,57	0,60	0,64	22,58	0,77	0,76	0,72	0,80	0,78
SET052-6	0,49	0,48	0,60	0,65	1,46	62,72	22,63	22,65	0,71	1,04	0,88	1,28	6,63
SET053-6	0,46	0,45	0,62	0,69	1,46	62,75	22,47	0,66	0,55	1,01	0,88	1,32	6,58
SET054-6	0,45	0,47	0,53	0,50	0,51	0,52	0,61	0,61	0,57	0,81	0,65	0,75	0,89
SET054-7	0,48	0,45	0,45	0,45	0,53	0,53	0,54	0,59	0,56	0,79	0,72	0,70	0,78

Problema	RR-25	RR-50	RR-100	RR-200	RR-250	RR-300	RR-350	RR-400	RR-450	RR-500	RR-550	RR-600	OTTER
SET055-6				0,52	0,57	0,51	0,62	0,64	0,56	0,83	0,73	0,72	0,83
SET055-7				0,53	0,58	0,55	0,56	0,66	0,60	0,82	0,74	0,73	0,81
SET056-6	0,46	0,50	0,50	0,55	0,59	0,52	0,61	0,62	0,59	0,78	0,70	0,82	1,36
SET056-7	0,48	0,49	0,48	0,54	0,57	0,61	0,62	0,63	0,64	0,81	0,73	0,84	1,30
SET057-6	0,50	0,47	0,50	0,53	0,54	0,56	0,61	0,61	0,66	0,80	0,69	0,80	1,38
SET057-7	0,50	0,52	0,48	0,54	0,55	0,56	0,60	0,61	0,60	0,79	0,70	0,84	1,42
SET058-6	0,45	0,46	0,48	0,54	0,54	0,57	0,60	0,64	0,61	0,81	0,70	0,86	1,36
SET058-7	0,49	0,45	0,52	0,52	0,55	0,65	0,63			0,84	0,75	0,86	1,34
SET059-6	0,51	0,49	0,49	5,43	27,89	33,24	34,53						
SET059-7	0,48	0,45	0,51	5,40	27,90	33,26	34,92	0,59	0,62				
SET060-6	0,44	0,49	0,50	0,49	0,55	0,61	0,63	0,64	0,61	0,78	0,69	0,73	0,75
SET060-7	0,51	0,50	0,47	0,53	0,55	0,54	0,61	0,67	0,60	0,83	0,68	0,80	0,83
SET061-6	0,48	0,47	0,53	0,54	0,56	0,55	0,60	0,63	0,58	0,79	0,68	0,77	0,83
SET061-7	0,45	0,45	0,45	0,54	0,55	0,60	0,60	0,63	0,63	0,80	0,66	0,73	0,82
SET062-6	0,51	0,46	0,49	0,48	0,55	0,63	0,57	0,59	0,57	0,79	0,70	0,75	0,78
SET062-7	0,49	0,47	0,50	0,52	0,56	0,59	0,60	0,61	0,61	0,76	0,67	0,77	0,81
SET063-6	0,46	0,51	0,49	0,52	0,51	0,56	0,66	0,64	0,58	0,78	0,73	0,76	0,84
SET063-7	0,47	0,47	0,54	0,52	0,56	0,56	0,62	0,63	0,63	0,77	0,74	0,77	0,92
SET064-6	0,48	0,51	0,50	0,53	0,53	0,56	0,63	0,67	0,66	0,81	0,67	0,79	0,85
SET064-7	0,51	0,47	0,44	0,51	0,59	0,61	0,64	0,64	0,56	0,78	0,71	0,79	0,92
SET065-6		0,47	0,49	0,56	0,58	0,54	0,61	0,57	0,61	0,80	0,68	0,78	0,81
SET065-7		0,50	0,51	0,58	0,59	0,56	0,63	0,63	0,63	0,81	0,68	0,74	0,76
SET067-6	0,47	0,51	0,49	0,51	0,56	0,58	0,65	0,67	0,58	0,78	0,71	0,75	0,86
SET067-7	0,45	0,48	0,50	0,49	0,59	0,58	0,66	0,62	0,60	0,82	0,78	0,77	0,93
SET068-6	0,45	0,47	0,47	0,52	0,59	0,61	0,67	0,61	0,65	0,87	0,74	0,78	0,81
SET068-7	0,49	0,47	0,50	0,51	0,54	0,59	0,66	1,23	40,78	0,83	0,77	0,76	0,85
SET071-6	0,50	0,46	0,50	0,83	16,37	1,14	1,21	1,21	40,86	1,25	1,02	1,95	22,63
SET071-7	0,48	0,48	0,51	0,84	16,42	1,12	1,20			1,23	1,00	1,93	22,86
SET072-6		0,59	4,32	29,68									
SET072-7		0,59	4,38	29,65				10,11					
SET073-6		0,48	0,53	0,67	15,69	16,94	9,91	10,12					
SET073-7		0,53	0,51	0,68	15,64	16,81	9,86	10,12					
SET074-6		0,50	0,54	0,66	15,68	19,15	9,87	10,18					
SET074-7		0,54	0,51	0,68	18,30	17,75	9,84	0,72	0,62				
SET075-6	0,46	0,52	0,57	0,54	0,58	0,63	0,71	0,66	0,60	0,70	0,70	0,78	0,76
SET075-7	0,54	0,47	0,63	0,63	0,62	0,57	0,66	6,34		0,71	0,71	0,80	0,81
SET076-6			0,50	1,10	1,19	1,23	1,12	6,36					
SET076-7			0,50	1,08	1,25	1,23	1,16	0,65	0,64				
SET077-6		0,57	0,52	0,57	0,56	0,54	0,62	0,64	0,64	0,73	0,63	0,80	0,76
SET077-7		0,48	0,52	0,54	0,59	0,57	0,62	0,66	0,61	0,71	0,65	0,78	0,80
SET078-6		0,53	0,50	0,58	0,58	0,63	0,62	0,61	0,62	0,69	0,60	0,82	0,77
SET078-7		0,55	0,50	0,56	0,54	0,56	0,59	51,68		0,71	0,61	0,77	0,79
SET079-6		0,57	0,50	0,55	0,59	0,60	51,49	51,70					
SET079-7		0,51	0,53	0,55	0,59	0,65	51,61	0,62	0,67				
SET080-6	0,49	0,54	0,54	0,56	0,55	0,51	0,64	0,62	0,66	0,67	0,63	0,77	0,75
SET080-7	0,50	0,52	0,53	0,56	0,57	0,59	0,66	0,65	0,64	0,67	0,64	0,77	0,75
SET081-6		0,55	0,53	0,61	0,60	0,62	0,63	0,65	0,67	0,67	0,64	0,84	1,48
SET081-7		0,52	0,54	0,59	0,53	0,55	0,65	0,63	0,64	0,68	0,63	0,92	1,41
SET082-6		0,57	0,53	0,59	0,54	0,55	0,65	0,63	0,61	0,69	0,64	0,79	0,78
SET082-7		0,57	0,52	0,62	0,61	0,60	0,62			0,77	0,60	0,83	0,83
SET083-6		0,51	0,67	3,02									

Problema	RR-25	RR-50	RR-100	RR-200	RR-250	RR-300	RR-350	RR-400	RR-450	RR-500	RR-550	RR-600	OTTER
SET083-7		0,60	0,62	2,95									
SET084-6		0,50	0,58	1,26									
SET084-7		0,57	0,61	1,26									
SET085-6			0,52	1,71									
SET085-7			0,52	1,73				0,64	0,72				
SET090-7		0,45	0,48	0,49	0,57	0,58	0,68	0,72	42,19	0,73	0,71	0,76	0,78
SET093-6	0,44	0,46	0,51	0,52	0,56	0,65	0,76	0,69	42,34	0,85	0,85	1,23	6,12
SET093-7	0,48	0,44	0,54	0,50	0,58	0,65	0,73			0,87	0,80	1,19	6,11
SET094-6	0,45	0,47	0,65	7,52									
SET094-7	0,46	0,50	0,65	7,47				0,61	0,63				
SET095-6		0,45	0,48	0,50	0,57	0,61	0,60	0,70	0,69	0,69	0,64	0,73	0,77
SET095-7		0,48	0,45	0,52	0,62	0,59	0,59	101,72		0,68	0,62	0,72	0,83
SET096-6	0,45	0,51	0,43	0,51	0,58	0,62	0,73	101,24					
SET096-7	0,46	0,42	0,50	0,52	0,59	0,58	0,72	0,60	0,67				
SET097-6	148,99	0,49	0,48	0,49	0,57	0,59	0,61	0,63	0,67	0,69	0,60	0,74	0,77
SET097-7	149,07	0,47	0,44	0,51	0,59	0,60	0,60			0,72	0,68	0,75	0,77
SET098-6													
SET098-7									57,14				
SET099-7	0,65			3,52	10,68	128,15		0,65	0,62	0,80	0,66	0,74	0,83
SET101-6		0,51	0,53	0,60	0,59	0,62	0,68	0,64	0,58	0,80	0,64	0,74	0,79
SET101-7		0,52	0,52	0,56	0,62	0,67	0,63	0,62	0,57	0,83	0,66	0,77	0,75
SET102-6		0,52	0,57	0,60	0,61	0,61	0,63	0,64	0,63	0,85	0,67	0,80	0,72
SET102-7		0,52	0,58	0,62	0,62	0,60	0,65	0,65	0,61	0,84	0,66	0,78	0,76
SET103-6		0,50	0,56	0,58	0,68	0,61	0,59	0,65	0,61	0,80	0,71	0,75	0,77
SET103-7		0,52	0,54	0,59	0,61	0,65	0,62	26,94	23,62	0,78	0,69	0,73	0,77
SET104-7				0,89	1,54	70,32	26,77	27,55	23,85		0,84	1,32	6,22
SET105-7		3,41	25,38	1,19	2,10	60,75	27,22	0,70	0,63		1,01	1,94	21,16
SET108-6	0,49	0,55	0,52	0,58	0,68	0,69	0,77	0,68	0,63	0,65	0,74	0,80	0,80
SET108-7	0,48	0,57	0,51	0,59	0,65	0,70	0,71	25,04		0,69	0,73	0,83	0,79
SET117-6	0,52	0,49	0,64	1,16	1,22	69,27	25,53	24,35			0,87	1,27	6,30
SET117-7	0,55	0,49	0,67	1,16	1,21	69,34	25,09	0,95	19,37		0,85	1,28	6,50
SET118-6	0,49	0,51	0,54	0,67	1,92	0,94	0,95	0,92	19,39	0,80	0,88	1,31	6,20
SET118-7	0,51	0,53	0,51	0,65	1,92	0,97	0,87	0,62	0,61	0,79	0,83	1,28	6,47
SET119-7	0,49	0,55	0,50	0,59	0,61	0,64	0,58	0,61	0,60	0,62	0,73	0,76	0,72
SET120-7	0,49	0,53	0,52	0,58	0,61	0,63	0,61	19,73	19,36	0,67	0,68	0,72	0,80
SET121-7				0,74	2,07	20,86	19,64	19,71	19,37	0,79	0,90	1,33	6,18
SET122-7				0,72	2,08	20,89	19,69			0,82	0,84	1,28	6,43
SET123-6								2,05					
SET124-6						1,42	2,07						
SET125-6						33,90	31,54						
SET126-6													
SET128-6								2,07					
SET130-6						2,09	2,06						
SET131-6													
SET132-6													
SET134-6													
SET135-6													
SET136-6													
SET137-6								10,64					
SET138-6						12,89	10,70						
SET144-6													

Problema	RR-25	RR-50	RR-100	RR-200	RR-250	RR-300	RR-350	RR-400	RR-450	RR-500	RR-550	RR-600	OTTER
SET146-6			0,63							1,53	2,02	8,77	
SET147-6													
SET148-6													
SET149-6													
SET150-6							128,88						
SET151-6						30,99		2,93	106,94				
SET152-6				99,32			2,85	0,88	0,79				
SET153-6		0,52	0,55	0,59	0,87	0,90	0,87			0,66	0,76	0,81	1,75
SET154-6													
SET155-6													
SET156-6							5,76						
SET157-6													
SET158-6				1,33	1,38	1,46	1,75						
SET160-6													
SET162-6													
SET163-6													
SET165-6								63,48					
SET166-6	0,50	0,46	0,52	0,60	8,67	11,90	14,40	0,85	0,61				
SET167-6				0,58	0,62	0,69	0,84	0,85	0,62	0,65	1,00	0,80	0,89
SET168-6				0,59	0,65	0,62	0,80			0,66	0,93	0,76	0,86
SET173-6													
SET175-6													
SET183-6								10,96					
SET184-6			0,51	0,59	0,65	2,03	3,64						
SET185-6													
SET186-6			0,63	0,65	0,72	1,22	5,12						
SET187-6				1,97	3,32	7,23	7,97						
SET188-6				4,76									
SET189-6				5,80									
SET190-6				16,60	21,33								
SET191-6													
SET192-6			1,44	2,69	6,93								
SET193-6			0,52	0,53	0,69	1,00	9,22						
SET194-6			27,43	11,98									
SET195-6			26,67	11,51									
SET196-6			0,61	0,59	0,87		12,17						
SET197-6			0,61	0,63	0,79		12,18						
SET199-6			39,90	43,96	88,35		110,98						
SET201-6													
SET202-6													
SET203-6						7,57							
SET204-6			0,67	0,83	1,09								
SET205-6			0,61							1,61	1,77	7,88	
SET206-6			0,60							1,64	1,84	8,20	
SET208-6													
SET209-6													
SET214-6													
SET215-6													
SET216-6													
SET217-6													
SET223-6													

Problema	RR-25	RR-50	RR-100	RR-200	RR-250	RR-300	RR-350	RR-400	RR-450	RR-500	RR-550	RR-600	OTTER
SET230-6													
SET231-6			0,60	16,52	18,09	18,17	22,24						
SET232-6			0,59										
SET233-6			0,63										
SET234-6			0,51	1,05	1,31								
SET235-6				0,91	6,06								
SET236-6				1,00	7,15								
SET238-6													
SET239-6													
SET240-6													
SET241-6													
SET242-6						35,43	34,95						
SET243-6													
SET244-6													
SET245-6													
SET246-6													
SET247-6													
SET252-6													
SET253-6													
SET254-6													
SET257-6													
SET260-6								17,30	131,13				
SET261-6			0,52	1,36	10,52	10,10	11,80	99,59		1,00	1,06	2,76	52,12
SET280-6							98,53						
SET287-6					3,49	7,63	5,52	0,51	0,57				
SET296-6				0,58	0,60	0,58	0,54			0,63	0,68	0,74	0,79
SET347-6													
SET382-6													
SET386-6								21,14	20,83				
SET411-6				1,65	70,59	77,65	21,14				0,94	1,52	7,39
SET442-6													
SET451-6													
SET454-6													
SET468-6			0,57	10,89									
SET473-6								4,87					
SET479-6				4,06	107,53	19,17	4,82						
SET497-6													
SET503-6													
SET504-6													
SET505-6								0,57	0,66				
SET506-6	0,48	0,47	0,47	0,54	0,65	0,62	0,59	0,53	0,66	0,69	0,65	0,88	0,77
SET507-6	0,49	0,48	0,49	0,52	0,58	0,59	0,56			0,64	0,65	0,85	0,79
SET510-6													
SET511-6							40,01						
SET515-6													
SET516-6													
SET517-6													
SET518-6													
SET553-6								0,68	0,73				
SET558-6	0,48	0,47	0,51	0,50	0,68	0,68	0,68			0,75	0,98	64,33	3,53
SET559-6													

Problema	RR-25	RR-50	RR-100	RR-200	RR-250	RR-300	RR-350	RR-400	RR-450	RR-500	RR-550	RR-600	OTTER
SET561-6				19,20									
SET562-6								0,65	0,80				
SET563-6	0,53	0,55	0,56	0,61	0,73	0,77	0,65			0,80	1,60		7,23
SET564-6			7,59										
SET565-6													
SET566-6								1,45	1,49				
SET786-1	0,56	0,52	0,57	0,84	1,44	1,54	1,43			1,47	1,55		

Tabela 6 - Lista completa dos resultados da massa "Testes 1"

B.2 Massa de Testes 2

Problema	RR-25	RR-50	RR-100	RR-200	RR-250	RR-300	RR-350	RR-400	RR-450	RR-500	OTTER
GRP001-1			1,46	1,38	1,53	1,40	1,44	1,47	1,44	1,65	1,30
GRP001-2	1,15	1,11	1,17	1,36	1,22	1,19	1,16	1,55	1,37	1,32	1,50
GRP001-4	1,15	1,07	1,13	1,39	1,20	1,22	1,24	1,18	1,19	1,38	1,28
GRP001-5			1,30	1,25	1,35	1,39	1,39	1,48	1,33	1,67	1,34
GRP002-1			1,38	1,67	1,69	3,26	3,10	2,87	2,93	3,11	
GRP002-2	1,68	1,21	1,15	1,21	1,26	1,45	1,26	1,53	1,62	1,58	1,75
GRP002-3		1,17	1,00	1,00	1,05	1,10	1,09	1,10	1,11	1,16	1,73
GRP002-4		1,14	1,00	1,02	1,09	1,09	1,06	1,12	1,09	1,13	1,76
GRP003-1		0,90	0,97	1,02	1,09	1,04	1,09	1,08	1,09	1,12	1,72
GRP003-2		0,96	1,02	1,02	1,03	1,03	1,10	1,10	1,13	1,15	1,78
GRP004-1		1,02	0,98	1,04	1,05	1,07	1,04	1,08	1,13	1,18	1,74
GRP004-2		0,98	1,05	1,07	1,11	1,08	1,06	1,16	1,17	1,05	1,86
GRP005-1					2,84	3,56	3,05	3,02	4,20	4,22	
GRP006-1				1,02	1,06	1,05	1,12	1,13	1,12	1,13	2,47
GRP007-1	0,94	0,96	0,97	0,97	1,07	1,05	1,08	1,13	1,17	1,13	2,61
GRP008-1											
GRP009-1			1,06	1,13	1,14	1,17	1,17	1,18	1,26	1,31	7,56
GRP010-1			1,09	1,18	1,17	1,17	1,20	1,22	1,30	1,27	10,48
GRP010-4	1,06	1,02	1,13	1,16	1,15	1,24	1,17	1,24	1,24	1,24	1,93
GRP011-4	1,07	1,10	1,08	1,44	1,48	1,42	1,31	1,36	1,42	2,72	
GRP012-1		1,17	1,16	1,15	1,12	1,19	1,22	1,22	1,26	1,79	15,32
GRP012-2		1,12	1,17	1,15	1,30	1,23	1,29	1,35	1,69	1,68	104,49
GRP012-3		1,11	1,05	1,15	1,15	1,17	1,22	1,27	1,20	1,30	1,71
GRP012-4		1,10	1,11	1,17	1,15	1,12	1,18	1,27	1,18	1,19	1,73
GRP013-1			1,17	1,24	1,22	1,25	1,25	1,35	1,31	1,31	1,77
GRP014-1			1,15	1,13	1,17	1,09	1,23	1,21	1,27	1,25	1,71
GRP017-1			1,16	1,11	1,17	1,22	1,27	1,21	1,32	1,33	15,70
GRP018-1		1,08	1,10	1,21	1,16	1,18	1,19	1,22	1,21	1,28	2,00
GRP019-1		1,12	1,15	1,22	1,17	1,23	1,24	1,26	1,25	1,33	1,79
GRP020-1		1,15	1,13	1,22	1,11	1,26	1,19	1,24	1,20	1,25	1,71
GRP021-1		1,12	1,15	1,19	1,20	1,20	1,19	1,24	1,28	1,26	1,76
GRP022-1			1,13	1,18	1,16	1,16	1,23	1,36	1,32	1,34	1,76
GRP022-2			1,17	1,23	1,18	1,24	1,17	1,21	1,23	1,25	1,79
GRP023-1		1,05	1,13	1,21	1,17	1,17	1,20	1,22	1,32	1,26	1,72
GRP023-2		1,06	1,15	1,18	1,18	1,30	1,34	1,23	1,36	1,33	1,78
GRP024-5	1,06	1,14	1,17	1,14	1,28	1,21	1,23	1,16	1,26	1,22	1,83
GRP025-1											
GRP025-3								5,44	5,45	9,90	
GRP026-1											

Problema	RR-25	RR-50	RR-100	RR-200	RR-250	RR-300	RR-350	RR-400	RR-450	RR-500	OTTER
GRP026-2											
GRP026-3								12,76	13,07	23,80	
GRP026-4											
GRP027-2								3,97	4,17	7,05	
GRP028-1		1,13	1,11	1,14	1,11	1,18	1,17	1,20	1,17	1,15	1,67
GRP028-2		1,07	1,13	1,10	1,14	1,10	1,19	1,23	1,17	1,17	1,78
GRP028-3		1,09	1,07	1,15	1,13	1,11	1,14	1,24	1,22	1,24	1,71
GRP029-1		1,05	1,06	1,14	1,15	1,07	1,17	1,18	1,25	1,24	1,76
GRP029-2		1,06	1,13	1,02	1,08	1,22	1,19	1,18	1,26	1,22	1,83
GRP030-1		1,11	1,11	1,09	1,19	1,21	1,19	1,24	1,28	1,22	1,74
GRP031-1	1,04	1,09	1,17	1,17	1,18	1,17	1,28	1,25	1,23	1,29	1,85
GRP031-2	1,12	1,13	1,14	1,26	1,28	1,16	1,24	1,24	1,22	1,30	1,88
GRP032-3	1,11	1,09	1,13	1,12	1,19	1,16	1,26	1,21	1,27	1,34	1,82
GRP033-3											
GRP033-4											
GRP034-3	1,04	1,11	1,13	1,30	1,32	1,28	1,33	1,30	1,34	1,37	1,85
GRP034-4	1,03	1,09	1,17	1,26	1,31	1,32	1,28	1,34	1,42	1,41	1,80
GRP035-3	1,03	1,12	2,54	2,69	2,65	2,75	2,75	2,75	2,75	2,91	19,42
GRP036-3		14,59	24,32	24,15	24,21	24,38	25,47	93,11			
GRP037-3											
GRP038-3	1,15	2,82	10,47	29,53	29,23	29,12	29,44	33,52	33,86	39,80	
GRP039-1				10,72	10,79	10,86	10,79	12,60	12,22	14,49	154,56
GRP039-2				9,57	9,66	9,77	9,77	11,34	11,21	13,56	
GRP039-3				10,66	11,14	10,70	11,02	12,30	12,59	14,73	155,52
GRP039-4				10,82	11,09	11,11	11,13	12,52	12,55	14,84	154,40
GRP039-5				10,17	10,15	10,17	10,05	11,86	11,98	14,00	
GRP039-6				11,18	11,45	11,39	11,33	12,84	13,10	15,20	148,80
GRP039-7				10,51	10,39	10,58	10,42	12,02	12,05	14,15	
GRP040-3				10,07	10,05	10,21	10,06	11,75	11,57	13,99	
GRP040-4				9,51	9,34	9,66	9,39	10,84	11,11	12,91	136,80
GRP041-2			0,94	1,03	1,00	1,05	1,03	1,04	1,11	1,13	1,23
GRP042-2			0,96	0,97	1,05	1,00	1,02	1,03	1,12	1,13	1,32
GRP043-2			1,00	1,05	1,04	0,98	1,03	1,07	1,13	1,13	2,87
GRP044-2			0,98	0,99	1,03	1,04	1,10	1,10	1,18	1,20	9,54
GRP045-2			0,96	1,09	1,07	1,00	1,06	1,09	1,21	1,15	9,53
GRP046-2			1,02	1,03	1,04	1,05	1,02	1,10	1,12	1,05	1,31
GRP047-2			0,95	1,06	1,04	1,06	1,06	1,08	1,11	1,09	1,33
GRP048-2			0,99	1,01	1,02	1,10	1,06	1,13	1,13	1,12	1,29
GRP049-1			1,04	1,02	1,05	1,10	1,04	1,09	1,06	1,14	1,23
GRP050-1			1,03	0,98	1,03	1,12	1,05	1,13	1,14	1,12	1,32
GRP051-1			1,03	1,05	1,07	1,04	1,11	1,15	1,19	1,09	1,23
GRP052-1			1,03	1,09	1,06	1,10	1,12	1,09	1,15	1,24	1,32
GRP053-1			1,12	1,03	1,03	1,17	1,14	1,15	1,17	1,22	1,32
GRP054-1			1,11	1,09	1,12	1,10	1,16	1,15	1,20	1,20	1,32
GRP055-1			1,14	1,15	1,13	1,19	1,18	1,16	1,20	1,20	1,25
GRP056-1			1,07	1,17	1,11	1,20	1,20	1,23	1,22	1,23	1,30
GRP057-1			1,15	1,12	1,09	1,15	1,16	1,24	1,21	1,26	1,39
GRP058-1			1,18	1,14	1,21	1,17	1,12	1,19	1,16	1,19	1,29
GRP059-1			1,14	1,13	1,06	1,24	1,16	1,22	1,25	1,20	1,28
GRP060-1			1,14	1,24	1,07	1,16	1,21	1,24	1,19	1,19	1,39
GRP061-1			1,01	1,06	1,08	1,00	1,07	1,11	1,15	1,09	1,46

Problema	RR-25	RR-50	RR-100	RR-200	RR-250	RR-300	RR-350	RR-400	RR-450	RR-500	OTTER
GRP062-1			1,03	1,06	1,08	1,05	1,05	1,09	1,16	1,11	1,31
GRP063-1			1,03	1,07	1,01	1,07	1,08	1,12	1,19	1,10	1,20
GRP064-1			1,10	1,14	1,18	1,16	1,15	1,12	1,27	1,22	1,29
GRP065-1			0,98	1,12	1,13	1,03	1,14	1,08	1,16	1,16	1,33
GRP066-1			1,07	1,06	1,06	1,08	1,09	1,11	1,09	1,11	1,34
GRP067-1			1,07	1,12	1,05	1,13	1,15	1,07	1,12	1,21	1,36
GRP068-1			1,01	1,11	1,13	1,14	1,15	1,10	1,18	1,13	1,42
GRP069-1			1,10	1,12	1,09	1,12	1,12	1,16	1,29	1,19	1,29
GRP070-1			1,17	1,10	1,11	1,17	1,16	1,19	1,18	1,24	1,27
GRP071-1			1,09	1,15	1,13	1,15	1,17	1,21	1,24	1,19	1,24
GRP072-1			1,17	1,12	1,11	1,16	1,15	1,22	1,15	1,22	1,45
GRP073-1			1,15	1,13	1,10	1,20	1,17	1,21	1,21	1,25	1,33
GRP074-1			1,14	1,18	1,19	1,23	1,19	1,20	1,27	1,24	1,33
GRP075-1			1,14	1,12	1,17	1,20	1,12	1,21	1,30	1,28	1,22
GRP076-1			1,03	1,05	1,01	1,02	1,05	1,05	1,11	1,17	1,32
GRP077-1			1,02	1,00	1,05	1,07	1,05	1,14	1,10	1,10	1,31
GRP078-1			0,99	0,97	1,04	1,08	1,07	1,12	1,13	1,15	1,37
GRP079-1			1,06	1,06	1,06	1,00	1,13	1,16	1,10	1,16	1,42
GRP080-1			1,05	1,06	1,05	1,02	1,09	1,12	1,12	1,10	1,32
GRP082-1			1,06	0,99	1,05	1,11	1,11	1,12	1,14	1,16	1,39
GRP083-1			1,06	1,09	1,04	1,11	1,15	1,06	1,17	1,15	1,36
GRP084-1			1,08	1,05	1,08	1,09	1,13	1,15	1,14	1,22	1,38
GRP085-1			1,05	1,09	1,11	1,11	1,11	1,18	1,21	1,18	1,31
GRP086-1			1,05	1,06	1,08	1,10	1,15	1,25	1,15	1,16	1,31
GRP087-1			1,09	1,12	1,13	1,15	1,15	1,22	1,20	1,17	1,36
GRP088-1			1,11	1,08	1,09	1,12	1,15	1,17	1,23	1,29	1,25
GRP089-1			1,16	1,19	1,16	1,15	1,14	1,21	1,22	1,27	1,37
GRP090-1			1,14	1,21	1,18	1,12	1,21	1,24	1,21	1,23	1,38
GRP091-1			1,16	1,17	1,20	1,14	1,22	1,22	1,20	1,20	1,33
GRP092-1			1,14	1,16	1,17	1,13	1,21	1,22	1,22	1,28	1,38
GRP093-1			1,19	1,22	1,25	1,19	1,25	1,15	1,28	1,26	1,36
GRP094-1			1,03	1,04	1,05	1,04	1,09	1,09	1,09	1,10	1,35
GRP095-1			0,99	0,99	1,03	1,02	1,07	1,11	1,13	1,10	1,42
GRP096-1			1,02	1,01	1,00	1,06	1,08	1,08	1,06	1,19	1,44
GRP097-1			1,04	1,08	1,03	1,06	1,14	1,10	1,12	1,18	1,41
GRP098-1			1,04	1,07	1,07	1,08	1,09	1,13	1,19	1,14	1,43
GRP099-1			1,05	1,04	1,14	1,07	1,09	1,10	1,16	1,16	1,31
GRP100-1			1,09	1,09	1,15	1,04	1,10	1,16	1,14	1,21	1,35
GRP101-1			1,02	1,14	1,10	1,09	1,14	1,20	1,17	1,22	1,40
GRP102-1			1,04	1,11	1,18	1,14	1,17	1,19	1,18	1,18	1,47
GRP103-1			1,05	1,08	1,14	1,13	1,19	1,14	1,16	1,19	1,42
GRP104-1			1,18	1,11	1,14	1,20	1,16	1,23	1,15	1,21	1,31
GRP105-1			1,14	1,16	1,14	1,18	1,16	1,19	1,15	1,18	1,45
GRP106-1			1,10	1,20	1,14	1,21	1,15	1,21	1,22	1,26	1,37
GRP107-1			1,12	1,17	1,20	1,20	1,20	1,20	1,24	1,24	1,45
GRP108-1			1,03	1,06	1,07	1,07	1,06	1,06	1,09	1,10	1,41
GRP109-1			1,04	1,11	1,07	1,13	1,08	1,05	1,08	1,07	1,35
GRP110-1			1,10	1,12	1,12	1,14	1,03	1,14	1,26	1,19	1,32
GRP111-1			1,09	1,08	1,10	1,08	1,10	1,17	1,13	1,17	1,56
GRP113-1			17,69		3,04	3,85	2,65	2,68	3,42	3,63	
GRP114-1											

Problema	RR-25	RR-50	RR-100	RR-200	RR-250	RR-300	RR-350	RR-400	RR-450	RR-500	OTTER
GRP115-1					22,79	42,27	14,68	14,28	20,80	21,24	
GRP116-1		0,98	1,03	1,04	1,10	1,07	1,19	1,13	1,18	1,12	1,28
GRP117-1		1,04	1,10	1,04	1,09	1,11	1,16	1,12	1,20	1,16	1,48
GRP118-1			1,05	1,06	1,03	1,14	1,05	1,17	1,12	1,20	1,46
GRP119-1							127,04	128,21			
GRP120-1		1,00	1,04	1,06	1,15	1,22	1,27	1,14	1,24	1,26	1,30
GRP121-1		1,08	1,09	1,09	1,17	1,17	1,18	1,20	1,18	1,19	1,29
GRP122-1			1,02	1,08	1,00	1,07	1,12	1,13	1,15	1,16	1,36
GRP123-1					1,06	1,15	1,18	1,13	1,18	1,18	1,30
GRP123-2					1,07	0,98	0,99	1,17	1,17	1,18	1,48
GRP123-3					1,01	1,03	1,11	1,14	1,14	1,18	1,51
GRP123-4					1,19	1,22	1,27	1,16	1,13	1,13	1,50
GRP123-6											
GRP135-2		0,99	1,02	1,00	1,08	1,07	1,97	1,16	1,33	1,48	1,49
GRP136-1	1,12	1,01	1,27	1,06	1,28	1,09	1,11	1,12	1,15	1,19	1,35
GRP137-1	0,97	0,99	1,03	1,09	1,09	1,20	1,12	1,10	1,18	1,13	1,64
GRP138-1	1,02	1,00	1,08	1,03	1,23						
GRP139-1	0,93	0,99	1,07	1,07	1,11	1,13	1,12	1,11	1,17	1,15	1,33
GRP140-1	1,01	1,01	1,02	1,14							
GRP141-1	0,89	0,92	0,95	0,98	1,06	1,10	1,20	1,27	1,26	1,33	147,90
GRP142-1	0,94	0,94	0,94	0,98	1,05	1,06	1,09	1,11	1,13	1,13	2,43
GRP143-1	0,90	0,94	0,97	1,00	1,06	1,09	1,13	1,11	1,12	1,13	2,42
GRP144-1	0,93	0,90	0,99	1,05	1,06	1,07	1,16	1,19	1,17	1,19	
GRP145-1			0,99	1,03	1,04	1,12	1,02	1,10	1,10	1,11	1,36
GRP146-1	0,92	0,97	0,94	1,02	1,01	1,03	1,12	1,06	1,05	1,11	1,33
GRP147-1	1,00	1,03	1,02	1,36	6,04						
GRP148-1	0,99	1,05	1,08	1,17							
GRP149-1	1,04	1,07	1,08	1,33	3,23						
GRP150-1	1,01	0,99	1,04	1,08	1,20	1,14	1,27	1,18	1,23	1,32	
GRP151-1	1,01	1,02	1,05	1,09	1,11	1,19	1,14	1,21	1,18	1,21	1,26
GRP152-1	0,97	1,02	1,05	1,14	1,16	1,14	1,16	1,21	1,19	1,22	2,39
GRP153-1	1,01	1,03	1,12	1,07	1,10	1,19	1,12	1,26	1,19	1,26	1,95
GRP154-1	0,95	1,03	1,01	1,08	1,08	1,07	1,13	1,15	1,13	1,21	1,19
GRP155-1	1,01	8,64	1,07	1,06	1,07	1,08	1,06	1,14	1,12	1,19	1,25
GRP156-1	1,02	1,13	1,09	1,02	1,07	1,05	1,03	1,15	1,22	1,16	4,52
GRP157-1	1,01	1,00	1,04	1,04	1,16	1,11	1,10	1,11	1,15	1,24	4,62
GRP158-1	1,03	8,45	1,05	1,03	1,09	1,12	1,10	1,10	1,13	1,19	1,30
GRP159-1		1,49	1,17	1,20	58,17		1,48	1,53	2,09		
GRP160-1	0,90	0,92	0,92	0,98	1,00	1,02	1,06	1,08	1,09	1,06	1,28
GRP161-1		0,98	0,92	1,24	1,02	0,94	0,95	1,02	1,09	1,14	1,28
GRP162-1	0,83	0,87	0,82	1,17	79,70						
GRP163-1	0,83	0,80	0,87	0,93	1,05	46,67					
GRP164-1	0,87	0,76	0,91	0,88	0,91	0,95	0,95	0,88	0,95	1,00	1,25
GRP164-2	0,85	0,90	0,83	0,94	0,90	0,93	0,96	0,99	0,99	1,03	1,31
GRP165-1			3,38	5,72	9,14	61,08	93,87	101,20	104,72		19,24
GRP165-2			4,93	6,17	12,16	61,11	114,31	121,43			1,37
GRP166-1			3,77	9,00	13,83	108,64	102,77	110,85	114,97		76,40
GRP166-2			5,74	7,23	14,37	59,48	122,13	130,36			1,47
GRP166-3			3,80	9,03	14,13	109,73	104,37	112,32	116,39		75,70
GRP166-4			5,85	7,39	14,64	60,38	123,18	131,71			1,30
GRP167-1											

Problema	RR-25	RR-50	RR-100	RR-200	RR-250	RR-300	RR-350	RR-400	RR-450	RR-500	OTTER
GRP167-2											
GRP167-3											
GRP167-4											
GRP167-5											
GRP168-1	0,82	0,91	2,25	4,07							
GRP168-2	0,84	8,28	0,93	0,91	0,94	0,91	0,92	0,89	1,00	0,97	1,33
GRP169-1			0,90	1,00	1,13	1,15	1,13	1,19	1,18	17,58	
GRP169-2			0,86	10,21							
GRP170-1	16,16		11,10	10,45							
GRP170-2	15,12	89,22	86,01	11,95							
GRP170-3	6,21		7,95	9,97							
GRP170-4	5,66	1,00	8,03	10,48							
GRP171-1			3,66	8,77	13,57	106,08	100,58	108,37	112,09		48,50
GRP171-2			7,91	16,34	22,38						48,31
GRP172-1			8,02	10,39	22,11	81,79					1,30
GRP172-2			3,65	4,62	8,47	38,81	91,02	98,95	131,51	141,76	1,29
GRP173-1			0,88	0,96	0,97	0,96	0,95	0,99	1,00	0,99	1,91
GRP174-1			0,92	13,58			1,86	1,90	1,87	2,23	
GRP175-1		1,00	3,61	4,94	0,93	1,00	0,98	1,00	0,99	1,03	16,86
GRP175-2		1,06	3,52	5,25	0,96	0,95	0,91	0,92	1,03	1,03	1,26
GRP175-3			3,58	4,91	0,93	0,94	0,97	0,94	1,02	0,98	16,54
GRP175-4			3,57	5,29	14,17	60,02	125,65	133,50	137,50	140,85	1,27
GRP176-1		0,91	0,95								
GRP176-2				31,45	57,95	58,20	57,88	58,47			
GRP177-2			3,69	4,79	9,03	40,00	92,23	99,42	132,18	141,74	1,27
GRP178-1			5,42	18,96	42,83		107,78	115,46	119,94		80,69
GRP178-2			3,67	4,72	8,68	38,98	91,49	99,36	131,77	141,55	1,31
GRP179-1		0,91	0,88	0,92	0,92	0,93	0,94	0,95	0,96	0,96	1,27
GRP179-2							11,49	11,63	11,72	16,11	
GRP179-3							11,39	11,48	11,55	16,05	
GRP180-1											
GRP180-2											
GRP181-1	1,91	1,92	1,70	3,48							
GRP181-2											
GRP181-3	2,29	1,79	1,81	3,61	27,24	27,69	27,80	28,47	35,06		
GRP181-4											
GRP182-1			0,87	0,90	0,97	1,00	0,96	0,98	1,05	1,08	2,08
GRP182-2				0,92	0,96	0,99	1,03	0,99	1,07	1,02	2,17
GRP182-3			0,97	0,97	1,04	0,93	1,04	1,03	1,01	1,04	1,95
GRP182-4				1,05	1,02	0,98	0,97	1,03	1,01	0,99	1,94
GRP183-1											
GRP183-2										95,79	
GRP183-3											
GRP183-4											
GRP184-1											
GRP184-2	5,19										
GRP184-3											
GRP184-4	10,00	1,45	2,10	1,57	1,51	1,51	1,52	1,62	1,60	1,61	
GRP185-1											
GRP185-2											
GRP185-3											

Problema	RR-25	RR-50	RR-100	RR-200	RR-250	RR-300	RR-350	RR-400	RR-450	RR-500	OTTER
GRP185-4											
GRP186-1											
GRP186-2											
GRP186-3											
GRP186-4											
GRP187-1				11,74	1,35	1,46	1,49	1,46	1,40	1,44	12,02
GRP188-1	1,24	1,29	1,24	1,32	1,35	1,34	1,40	1,45	1,57	1,43	2,07
GRP188-2				1,28	1,37	1,37	1,39	1,43	1,42	1,39	2,04
GRP189-1	1,18	1,17	1,26	1,28	1,37	1,33	1,37	1,47	1,41	1,47	1,80
GRP189-2				1,38	1,34	1,33	1,39	1,47	1,44	1,40	1,87
GRP190-1	1,32	1,32	1,37	1,95	4,83	4,75	4,81	4,83	5,59		
GRP190-2	1,25	1,22	1,30	1,30	1,27	1,29	1,34	1,37	1,37	1,40	4,11
GRP191-1	1,24	1,25	1,36	1,34	1,52	1,54	1,53	1,58	1,67	24,96	
GRP191-2				2,17	6,08	6,03	6,13	6,17	7,13		
GRP192-1		1,24	1,26	1,24	1,36	1,39	1,38	1,46	1,47	1,50	5,72
GRP193-1			8,47	29,37	65,92						77,45
GRP193-2			5,57	7,20	13,73	61,31	141,10				1,20
GRP195-1	1,52	3,92	1,86	1,61	2,53	2,57	2,94	2,51	2,51	2,65	29,79
GRP196-1	1,54	4,42	1,68	1,72	2,71	2,59	2,62	2,75	2,88	2,98	1,46
GRP197-1	1,57	2,31	1,46	1,57	2,53	2,52	2,57	2,55	2,74	2,72	131,51
GRP198-1	1,76	4,45	2,08	2,39	3,67	3,71	3,77	3,86	3,99	4,06	
GRP199-1	1,40	58,17	1,74		76,35	27,25	27,40	35,29	4,25	4,37	135,18
GRP200-1		1,09	1,22	1,14	1,14	1,19	1,22	1,26	1,19	1,21	1,21
GRP201-1		1,08	1,15	1,15	1,18	1,23	1,22	1,21	1,23	1,26	1,20
GRP202-1		1,14	1,17	1,16	1,18	1,20	1,16	1,18	1,21	1,28	1,24
GRP203-1		1,10	1,23	1,19	1,19	1,20	1,18	1,21	1,26	1,34	1,26
GRP205-1	1,11	1,12	1,06	1,15	1,21	1,24	1,12	1,28	1,19	1,21	1,22
GRP206-1		1,09	1,15	1,19	1,20	1,20	1,17	1,21	1,26	1,24	1,25
GRP208-1											
GRP209-1											
GRP210-1											
GRP211-1											
GRP212-1											
GRP213-1											
GRP214-1											
GRP215-1											
GRP216-1				183,02	148,83	153,18					
GRP217-1											
GRP218-1											
GRP219-1											
GRP220-1											
GRP221-1				148,35	169,16	169,91					
GRP222-1											
GRP223-1											
GRP224-1											
GRP225-1											
GRP226-1											
GRP227-1				154,55	165,92	170,55					
GRP228-1				157,88	169,61	174,28					
GRP229-1				155,02	168,46	173,19					
GRP230-1											

Problema	RR-25	RR-50	RR-100	RR-200	RR-250	RR-300	RR-350	RR-400	RR-450	RR-500	OTTER
GRP231-1				28,46	31,60	31,56	41,42	43,99	60,13	96,33	
GRP232-1				30,66	33,90	34,26	44,11	52,02	58,82	108,11	
GRP233-1				29,06	31,85	32,30	41,84	42,08	51,54	82,84	
GRP234-1				29,38	32,52	32,51	42,47	42,73	52,00	84,01	
GRP235-1				30,07	33,19	33,28	42,90	41,28	48,63	78,45	
GRP236-1				1,14	1,21	1,24	1,17	1,18	1,20	1,27	64,78
GRP237-1											
GRP238-1											
GRP239-1				48,22	51,74	52,61	66,96	67,23	83,35	134,99	
GRP240-1											
GRP241-1											
GRP242-1											
GRP243-1											
GRP244-1											
GRP245-1											
GRP246-1											
GRP247-1											
GRP248-1											
GRP249-1											
GRP250-1											
GRP251-1											
GRP252-1											
GRP253-1											
GRP254-1											
GRP255-1											
GRP256-1											
GRP257-1											
GRP258-1											
GRP259-1											
GRP260-1											
GRP261-1											
GRP262-1											
GRP263-1											
GRP264-1											
GRP265-1											
GRP266-1											
GRP267-1											
GRP268-1											
GRP269-1											
GRP270-1											
GRP271-1											
GRP272-1											
GRP273-1											
GRP274-1											
GRP275-1											
GRP276-1											
GRP277-1											
GRP278-1											
GRP279-1											
GRP280-1											
GRP281-1											

Problema	RR-25	RR-50	RR-100	RR-200	RR-250	RR-300	RR-350	RR-400	RR-450	RR-500	OTTER
GRP282-1											
GRP283-1											
GRP284-1											
GRP285-1											
GRP286-1											
GRP287-1											
GRP288-1											
GRP289-1											
GRP290-1											
GRP291-1											
GRP292-1											
GRP293-1											
GRP294-1											
GRP295-1											
GRP296-1											
GRP297-1											
GRP298-1											
GRP299-1											
GRP300-1											
GRP301-1											
GRP302-1	3,77		3,12	2,17	2,19	1,82	1,81	1,46	1,44	1,47	35,96
GRP303-1				1,38	1,39	1,40	1,37	1,38	1,44	1,43	30,67
GRP304-1	3,24		3,12	2,21	2,14	1,76	1,86	1,46	1,43	1,51	36,99
GRP305-1											
GRP306-1	3,42		3,37	2,36	2,27	1,95	2,07	1,57	1,51	1,67	36,94
GRP307-1											
GRP308-1											
GRP309-1	6,00		2,50	2,09	2,13	1,81	1,85	1,49	1,49	1,50	35,53
GRP310-1	4,10		3,19	2,32	2,18	1,82	1,83	1,43	1,50	1,50	36,29
GRP311-1											
GRP312-1											
GRP313-1											
GRP314-1	3,83		3,24	2,38	2,23	1,94	1,95	1,39	1,47	1,53	29,23
GRP315-1	2,52		3,27	2,18	2,17	1,79	1,81	1,35	1,39	1,38	29,48
GRP316-1											
GRP317-1											
GRP318-1											
GRP319-1											
GRP320-1											
GRP321-1											
GRP322-1											
GRP323-1											
GRP324-1											
GRP325-1	21,62		34,66	33,18	38,73	38,96	38,66	31,30	30,81	31,41	64,72
GRP326-1	3,36		3,09	2,23	2,12	1,80	1,79	1,41	1,47	1,47	35,08
GRP327-1	5,12		2,90	2,09	2,07	1,72	1,69	1,25	1,29	1,30	30,37
GRP328-1											
GRP329-1				135,45	145,22	149,15					
GRP330-1											
GRP331-1											
GRP332-1											

Problema	RR-25	RR-50	RR-100	RR-200	RR-250	RR-300	RR-350	RR-400	RR-450	RR-500	OTTER
GRP333-1	2,44		2,90	2,09	2,06	1,75	1,77	1,35	1,32	1,35	30,55
GRP334-1	4,95		2,95	2,02	2,09	1,71	1,72	1,32	1,35	1,33	30,97
GRP335-1											
GRP336-1											
GRP337-1											
GRP338-1											
GRP339-1	35,05		10,79	9,05	9,60	9,29	9,29	6,76	6,85	6,88	38,65
GRP340-1											
GRP341-1											
GRP342-1											
GRP343-1	7,50		8,00	6,63	6,93	6,54	6,62	4,94	4,95	4,97	36,92
GRP344-1	2,56		3,23	2,13	2,12	1,79	1,85	1,46	1,56	1,51	56,06
GRP345-1											
GRP346-1											
GRP347-1											
GRP348-1											
GRP349-1								73,92	74,55	111,70	
GRP350-1											
GRP351-1											
GRP352-1											
GRP353-1											
GRP354-1											
GRP355-1											
GRP356-1											
GRP357-1								134,98	125,46		
GRP358-1											
GRP359-1								96,61	96,53	146,57	
GRP360-1											
GRP361-1											
GRP362-1				1,30	1,26	1,25	1,31	1,36	1,44	1,53	170,59
GRP363-1								90,03	90,40	134,21	
GRP364-1											
GRP365-1											
GRP366-1											
GRP367-1											
GRP368-1											
GRP369-1											
GRP370-1											
GRP371-1								132,60	132,97		
GRP372-1								114,98	114,91	169,81	
GRP373-1											
GRP374-1								112,19	113,01	166,32	
GRP375-1								106,18	106,12		
GRP376-1											
GRP377-1								104,09	104,63		
GRP378-1								100,65	100,48	148,57	
GRP379-1											
GRP380-1											
GRP381-1											
GRP382-1											
GRP383-1											

Problema	RR-25	RR-50	RR-100	RR-200	RR-250	RR-300	RR-350	RR-400	RR-450	RR-500	OTTER
GRP384-1								114,19	114,92		
GRP385-1				1,44	1,41	1,44	1,52	1,52	1,53	1,64	94,13
GRP386-1											
GRP387-1								105,36	105,96	158,74	
GRP388-1				1,40	1,36	1,39	1,37	1,43	1,41	1,49	98,48
GRP389-1											
GRP390-1											
GRP391-1											
GRP400-1			1,13	1,24	1,49	1,52	1,19	1,26	1,26	1,43	1,25
GRP401-1	1,25	1,38	1,54	1,53	1,36	1,19	1,46	1,54	1,53	1,51	1,25
GRP402-1			1,13	1,36	1,20	1,38	1,29	1,22	1,63	1,39	1,17

Tabela 7 - Lista completa dos resultados da massa "Testes 2"

Bibliografia

Brachman, R. J.; Levesque, H. J. A fundamental tradeoff in knowledge representation and reasoning. In: ------. **Readings in knowledge representation**. Morgan Kaufmann, 1985. Cap. 4, p. 41-70.

Cormen T. H.; Leiserson C. E.; Rivest R.L.; Stein C. **Introduction to Algorithms**. Second Edition. MIT Press. 2001.

Cycorp. **Informações sobre a Cycorp e seus produtos**. Disponível em: <http://cyc.com>. Aceso em: 23 de fevereiro de 2004.

Enderton, H. **A Mathematical Introduction to Logic**. London: Academic Press. 1972. 295 p.

Epstein, Richard L. **Propositional Logics:** The Semantic Foundations of Logics. Second Edition. Wadsworth Publishing, 2001. 525 p.

Gamut, L.T.F. **Logic, Language and Meaning**: Volume 1 – Introduction to Logic. The University of Chicago Press, 1991.

Greiner R.; Darken C.; Santoso N.I. Efficient Reasoning. **ACM Computing Surveys**, v. 33, i. 1, p.1-30, mar. 2001.

Krajewski S. Relatedness Logic. **Reports on Mathematical Logics**, v. 20, p. 7-14. 1986.

McCune W. **OTTER 3.3 Reference Manual**. 2003. Disponível em: <http://www-unix.mcs.anl.gov/AR/otter/otter33.pdf>. Acesso em: 23 de fev. de 2004.

Overbeek, R. A new class of automated theorem-proving algorithms. **Journal of the ACM,** v.21, i. 2, p. 191-200, abr. 1974.

Riazanov, A. **Implementing an Efficient Theorem Prover**. 2003. Tese de doutorado em ciência da computação submetida para avaliação no departamento de ciência da computação da Universidade de Manchester em dezembro de 2003. Disponível em: <http://www.cs.man.ac.uk/~riazanoa/Vampire/thesis.ps>. Acesso em: 23 de fev. de 2004.

Ribeiro F. **otterlib - a C library for theorm proving**. 2002. Disponível em: <http://www.ime.usp.br/~fr/otterlib/otterlib-techreport.ps.gz>. Acesso em: 23 de fev. de 2004.

Robinson G.A.; Wos L.T. Paramodulation and theorem proving in first order theories with equality. In: Meltzer B.; Michie D. **Machine Intelligence**. v. 4. Edinburgh. 1969. p. 135-150.

Robinson J. A. A Machine-Oriented Logic Based on the Resolution Principle. **Journal of the ACM,** v.12, i. 1, p. 23-41, jan. 1965.

Russell S.; Norvig P. **Artificial Intelligence**: a Modern Aproach. Prentice Hall. 1995. 932 p.

Sutcliffe, G.; Suttner, C. **The TPTP Problem Library for Automated Theorem Proving**. Disponível em: <http://www.cs.miami.edu/~tptp>. Acesso em: 23 de fev. de 2004.

Wassermann, R. **Resource-Bounded Belief Revision**. 2000. 160 f. Tese (Doutorado em Ciência da Computação) - Institute for Logic, Language and Computation, Universidade de Amsterdam, Amsterdam, 2000.

Wos L.; Robinson G.; Carson D. Efficiency and completeness of the set-of-support strategy in theorem proving. **Journal of the ACM,** v. 12, i. 4, p. 536-541, out. 1965.