Pedro Paulo Vezzá Campos
Ana Luísa Losnak
Daniel Moraes Huguenin

Implementação de Provador de Teoremas Usando Tableaux Analíticos

São Paulo - SP, Brasil

15 de outubro de 2011

Pedro Paulo Vezzá Campos
Ana Luísa Losnak
Daniel Moraes Huguenin

Implementação de Provador de Teoremas Usando Tableaux Analíticos

Primeiro exercício-programa apresentado para avaliação na disciplina MAC0239, do curso de Bacharelado em Ciência da Computação, turma 45, da Universidade de São Paulo, ministrada pela professora Ana Cristina Vieira de Melo.

DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA
UNIVERSIDADE DE SÃO PAULO

São Paulo - SP, Brasil

15 de outubro de 2011

Introdução

Para o primeiro exercício-programa (EP) de MAC0239 foi proposto pela professora a implementação de um provador de teoremas descritos em lógica proposicional utilizando o método dos tableaux analíticos em linguagem Java ou C. Este relatório apresenta as estruturas e estratégias utilizadas na implementação, além das dificuldades encontradas na concepção do programa.

Este relatório está organizado da seguinte forma: Primeiramente será apresentada uma breve descrição do método dos tableaux analíticos. Depois será apresentada a organização do programa implementado. Em seguida serão comentadas as estruturas de dados utilizadas. Posteriormente serão detalhadas as estratégias de implementação adotadas pelos alunos. Ainda, serão apresentadas as formas de teste aplicadas ao programa. Por fim serão vistas as principais dificuldades enfrentadas pelo alunos durante o projeto e codificação do EP juntamente com uma conclusão do trabalho.

Método dos Tableaux Analíticos

O método dos tableaux analíticos (Ou semânticos) é um sistema de prova baseado em refutação que possui como vantagem o fato de ser um decididor para a validade de um teorema, ou seja, o algoritmo sempre encerrará sua execução em um tempo não necessariamente exponencial retornando como resultado a validade de um dado teorema.

Seu funcionamento é simples e essencialmente algorítmico, permitindo uma fácil implementação computacional.

Organização do Programa

O EP foi implementado utilizando o paradigma de Orientação a Objetos, o que permitiu uma divisão clara de tarefas entre os diferentes objetos que cooperam para o funcionamento do programa. As principais classes implementadas encontram-se descritas abaixo:

main.Main Classe que contém o método `main()` do programa, com a implementação da interface do usuário. É responsável por ler os teoremas (Um por linha) a serem provados ou refutados, tratá-los e enviá-los ao objeto do tipo `ProvadorTeoremas`, exibindo o resultado da prova, “Sequente válido” caso o teorema seja válido ou “Sequente inválido.

Contra-exemplo: ...” caso contrário.

provadorTeoremas.ProvadorTeoremas Classe “Fachada” do provador. Contém o laço principal do programa e é responsável por escolher ramos a expandir além de verificar o fechamento e saturação desses ramos. Por fim, realiza as α-expansões e β-expansões.

provadorTeoremas.Formula Classe responsável por representar uma fórmula da lógica proposicional além de *parsear* uma string para um objeto desse tipo.

provadorTeoremas.FormulaMarcada Classe responsável por encapsular uma `Formula` junto com seu valor booleano. É responsável por retornar se é uma fórmula α ou β, e retornar suas subfórmulas marcadas de acordo com a regra de expansão correspondente.

provadorTeoremas.Ramo Classe responsável por encapsular os dados a serem armazenados na pilha de ramos conforme descrito no exemplo de implementação apresentado pela professora, ou seja, o tamanho do ramo no momento da expansão, a segunda fórmula β da expansão e o vetor de regras β ainda por serem expandidas quando o programa retornar a esse ramo.

provadorTeoremas.Resultado Classe que encapsula o resultado booleano da prova do sequente além de possivelmente um contra-exemplo para um sequente falsificáve.

testesUnitarios.Testes{Formula, FormulaMarcada, ProvadorTeoremas} Classes de testes unitários de suas respectivas classes implementados com a ajuda da biblioteca JUnit. Responsáveis por garantir o bom funcionamento das partes do provador.

Estruturas de Dados Utilizadas

O EP foi implementado utilizando a linguagem Java, o que permitiu aos alunos se isolarem de problemas de implementação e manipulação de estruturas de dados básicas durante a programação. Assim, procedeu-se de maneira geral por utilizar o máximo de ferramentas prontas disponíveis na API do Java.

A classe `Formula` foi implementada como uma árvore binária de operadores. Isso simplificou o código responsável por dividir uma fórmula ao realizar uma α ou β-expansão. Para construí-la partindo de uma *string* completamente parentetizada e bem formada, como definido no enunciado, foi criado um algoritmo localizado no método `parsearFormula` que monta recursivamente a árvore da formula passada.

Já a classe `ProvadorTeoremas` conta com as seguintes estruturas de dados:

`Stack<Ramo> pilhaRamos` Os ramos empilhados após sucessivas β-expansões.

`List<Boolean> betas` O vetor (Implementado com a classe `ArrayList`) de regras β ainda por expandir.

`List<FormulaMarcada> ramo` O vetor (Implementado com a classe `ArrayList`) de objetos `FormulaMarcada` que representam o ramo atual da prova do sequente.

Por fim, o objeto `Resposta` possui o campo `Set<FormulaMarcada> contraExemplo` (Implementado com a classe `HashSet`) para representar o contra-exemplo encontrado durante a prova de um sequente falsificável. O uso da estrutura de dados `Set` garante que não haverá duplicação de átomos no contra-exemplo apresentado ao usuário.

Estratégias de Implementação

Para a expansão do ramo foi utilizada a estratégia de realizar o máximo de α-expansões possível antes de proceder com β-expansões. Isso trouxe como benefício uma menor quantidade de ramos divididos no tableau aumentando a velocidade da prova.

Para a escolha de qual β-expansão aplicar foi escolhida a estratégia ascendente, que opta pela última regra β ainda por expandir no vetor `betas`.

De forma a economizar tempo e espaço computacional o provador remove regras α já expandidas trazendo maior facilidade na análise manual do funcionamento do provador e maior velocidade do programa já que a verificação do fechamento e saturação dos ramos devem iterar menos elementos por vez.

Testes

O bom funcionamento do programa foi verificado através de diversos testes unitários implementados com a biblioteca JUnit. Como consequência, a cada alteração no código fonte foi possível detectar erros de codificação de maneira fácil e rápida.

O *parseamento* de fórmulas foi testado com 24 testes diferentes enquanto o funcionamento do provador com um todo foi verificado com 56 testes distintos. Os testes unitários podem ser vistos no diretório `testesUnitarios` junto com o código fonte do programa.

Dentre os testes produzidos incluem-se: *Modus ponens*, *Modus tollens*, Leis de De Morgan,

contrapositiva, converso, silogismos hipotético e disjuntivo, resolução, adição, conjunção, simplificação, além de diversas tautologias clássicas tais como distributividade e comutatividade.

Por fim, a interface com o usuário também foi testada. As entradas aplicadas podem ser vistas no diretório `testes/Entradas.txt` e as saídas esperadas em `testes/Saidas.txt`.

Dificuldades Encontradas

O *parseamento* de fórmulas foi implementado inicialmente com um algoritmo iterativo que mostrou-se bastante frágil com casos de testes mais complexos. Assim, como trata-se de uma parte essencial ao bom funcionamento do programa optou-se por refazê-lo de maneira recursiva, o que trouxe ganhos de legibilidade de código além de maior robustez.

Ainda, outro problema encontrado foi garantir que o provador estava retornando resultados corretos para os sequentes. Inicialmente o provador foi implementado de maneira que à medida que uma fórmula fosse consumida por uma expansão ela seria removida do ramo. Isso facilitou o *debug* do programa inicialmente e funcionava corretamente para casos de teste menores mas provou-se incorreta para certos sequentes. Para corrigir isso, os alunos perceberam que preservando as fórmulas β no ramo esse problema seria resolvido.

Conclusão

Este primeiro EP de MAC0239 proposto pela professora Ana foi de grande valia por apresentar como proposta a implementação de um provador usando tableaux analíticos visto em aula, fixando conceitos e detalhes computacionais relevantes para futuros profissionais da Ciência da Computação.

Ainda, este trabalho contribuiu ao resumir bem diversos conceitos vistos durante o semestre na área de prova de teoremas até culminar no projeto, desenvolvimento e testes de uma aplicação prática. Por outro lado, como os exercícios não demandaram grandes e tediosos esforços de programação os acadêmicos puderam concentrar-se nas modelagens e abordagens adotadas para resolver o problema, as quais foram apresentadas anteriormente neste relatório.