TRABALHO 0: UM PROVADOR BASEADO EM RESOLUção DRAFT 0

RAUL H.C. LOPES

1. Introdução

O objetivo fundamental deste trabalho consiste em exercitar a construção de programas que operam sobre classes indutivas. Para tal você construirá uma série de programas que operam sobre estruturas de dados clássicas, cuja natureza indutiva está muito clara: você implementará um provador de teoremas para lógica de primeira ordem. Neste trabalho, seu provador estará restrito ao cálculo de proposições da lógica clássica. Em outro trabalho, isso será ampliado para atingir toda a lógica de primeira ordem.

Você deverá implementar um módulo (ou conjunto de módulo, dependendo de como você projetar sua solução), contendo os tipos que definem sua lógica, tradutores da linguagem externa para os tipos internos do seu provador, as regras de inferência. Para efeitos deste documento, tudo isso será aqui identificado como seu **provador** ou seu **sistema**

2. Seu provador e o mundo

Seu provador entenderá teorias apresentadas usando o cálculo de proposições e provar teoremas por contradição. Uma teoria será sempre uma conjunção de disjunções de literais. Um literal representará uma verdade atômica (via variável proposicional) ou a negação de uma verdade atômica. Por exemplo:

$$\begin{aligned} p &\text{ verdade atômica, afirmando } p \\ -p &\text{ a negação de } p \\ p|q &\text{ a afirmação de que } p \text{ ou } q \text{ valem} \\ p|-q &\text{ a afirmação de que } p \text{ ou a negação de } q \text{valem} \end{aligned}$$

Observe o uso de:

- | para representar a disjunção;

- $-$ para representar a negação;
- & para representar a conjunção.

Seu provador seguirá essas convenções. Ele entederá uma teoria como um conjunto de axiomas e interpretará a mesma como a conjunção dos axiomas que a compõem: ou seja, a conjunção dos axiomas da teoria estará implícita. Cada axioma será uma disjunção de literais.

Observação (e dica): uma teoria pode ser representada internamente como uma seqüência de axiomas e um axioma, como uma seqüência de literais.

2.1. **A linguagem do seu provador.**
A linguagem do seu provador está exposta acima:

- teorias serão conjunções de axiomas, que são, por sua vez, disjunções de literais;
- um teorema a provar é qualquer fórmula do cálculo de proposições, envolvendo conjunçõese disjunções de literais.

2.2. **Input/output do seu provador.**
Seu provador aceitará na linha de comando um string definindo o teorema a provar e lerá do *stdin* uma seqüência de contendo exatamente um axioma da teoria por linha.

Ele produzirá como output:

- **teorema**: Se conseguir provar o teorema em questão.
- **falha**: Se não conseguir provar o teorema.

2.3. **A lógica do seu provador.**
Seu provador construirá provas por contradição. Lida a teoria Γ e o teorema a provar α ele tentará obter uma contradição da conjunção de Γ com a negação de α. Provar:

$$\Gamma \& -\alpha \vdash \perp$$

Ele deverá realizar a seguinte seqüência de passos:

1. Ler teoria;
2. converter teoria para representação interna;
3. ler teorema a provar;
4. converter teorema para representação interna;
5. negar teorema (já na sua representação interna);
6. tentar derivar a contradição.

Na tentativa de derivar a contradição seu provador usará as seguintes regras.

- **resolução binaria**

Dadas as formulas

$$p|C_0 \text{ e}$$
$$-p|C_1 \text{ derive}$$
$$C_0|C_1$$

- **fatoração**
  Dada a fórmula

$$C_0|p|C_1|p|C_2 \text{ derive}$$
$$C_0|C_1|C_2$$

Uma fórmula vazia (sem literais) representa a contradição. Ela é derivada, por exemplo, de p e $-p$ por **resolução** binária.

3. A execução trabalho

3.1. **As etapas de construção.** Decomponha seu trabalho de acordo com as etapas abaixo.

3.1.1. *Os tipos de dados.* Defina as classes indutivas e os tipos de dados que as representarão na linguagem de implementação.

3.1.2. *Lendo um axioma.* Defina e implemente e procedimento para converter um string representando um axioma (ou uma conjectura) para a representação interna do seu sistema.

3.1.3. *Lendo uma teoria.* Defina e implemente procedimento para converter um conjunto de strings representando uma teoria para a representação interna da mesma no seu sistema.

3.1.4. *Negando uma conjectura.* Implementa procedimento que transforma a negação de uma conjectura em uma conjunção de disjunções.

3.1.5. *Apresentando output.* Defina e implemente procedimentos que permitam converter qualquer representação interna de fórmula para um string em acordo com a linguagem externa do provador.

3.1.6. *As regras de inferência.* Defina e implementas as regras de inferência acima.

3.1.7. *Estratégias.* Defina e implemente estratégias para melhorar o desempenho do seu provador.

3.2. **Linguagem de programação.** Implemente em Haskell ou C. Se você, adicionalmente, implementar em Isabelle e provar propriedade de correção da sua implementação você receberá mais créditos.

3.3. O que entregar. Você deverá entregar *tarball* contendo:

- A implementação do provador;
- Módulo de testes;
- Documentação em LaTeX;
- Makefile

Estando no dirtório de trabalho, compacte seus arquivos com o comando:

```
tar -zhcf tp0.tar.gz *
```

Teste seu *tarball*, executando a seguinte seqüência de comandos em um diretório temporário **vazio**:

```
tar -zxf tp0.tar.gz
make all
make test
```

Essa seqüência

3.4. Makefile. Seu *Makefile* conterá dois *targets*:

- **all**: que ativará a compilação de todos os programas bem como a geração de documentação em **PDF** a partir dos fontes em LaTeX.
- **test**: que ativará o módulo de testes.

3.5. Como entregar. Submeta seu trabalho por e-mail para

`raulh@inf.ufes.br`

O subject será:

ed:0

A mensagem conterá attachment com o arquivo do trabalho.

3.6. O módulo de testes. O módulo de testes consistirá neste trabalho simplesmente de um conjunto de aquivos contendo os axiomas de teoremas e não teoremas usados para testar seu provador. Além disso, seu *Makefile* conterá um *target* **test** que acionará a prova de cada um teoremas e não teoremas que constituem o teste.

3.7. A avaliação. A nota será atribuída de acordo com a tabela 1. Note que um item com pontuação $*0.8$ indica que a nota final do trabalho será multiplicada por 0.8.

Plágios anulam o trabalho atual e são remetidos imediatamente para direção do CT, que define punição que pode resultar em expulsão da universidade.

Provador	até 8.0
Módulo de testes	até 1.0
Documentação	até 1.0
Trabalho individual	$*1.2$
Trabalho em grupo de 2	$*1.0$
Trabalho em grupo de 3	$*0.8$
Trabalho em grupos de 4 ou mais	$*0$

Tabela 1. Pontuação do trabalho