Enunciado do Segundo EP

Marcelo Finger

MAC239 — Segundo Semestre de 2013 — BCC/IME
Entrega: 02/12/2013

1 Objetivos

Neste Exercício-Programa, vamos instanciar um conjunto de fórmulas de Lógica de Primeira Ordem, transformando-as em fórmulas proposicionais, e assim resolver problemas submetendo seu resultado a um SAT-Solver.

Como a lógica de primeira ordem é mais expressiva que a lógica proposicional, vamos fazer isso apenas com um fragmento da lógica de primeira ordem. Este fragmento não possui quantificadores explícitos, ou seja, podemos admitir que os quantificadores existenciais foram todos *skolemizados*, e que as demais variáves são implicitamente quantificadas universalmente. Além disso, vamos assumir que os domínios de quantificação são finitos, o que permitirá transformar as fórmulas de primeira ordem em fórmulas de lógica proposicional.

Por fim, iremos tratar de uma linguagem *puramente relacional*, em que os termos são apenas variáveis ou constantes. Desta forma, não há necessidade de tratar de símbolos funcionais.

2 Regras do Trabalho

1. Espera-se que este trabalho seja feito individualmente ou em duplas.

2. Para fins desta disciplina, as duplas contém exatamente 2 alunos, não existindo instâncias de duplas com número superior a 2 elementos ou número fracionário de elementos.

3. Cada indivíduo ou dupla entregará um relatório através do moodle no endereço `http://paca.ime.usp.br` pela conta de um dos membros da dupla. No caso das duplas, ambos devem estar inscritos no Paca e a nota será a mesma para ambos os membros da dupla de 2.

4. Deve ser entregue um único arquivo compactado (formato zip ou tgz), contendo o programa gerador das entradas para o resolvedor SAT e um relatório com tudo que é necessário para roda o programa e quaisquer observações. Não inclua arquivos executáveis.

5. O programa deve ser escrito em uma das seguintes linguagens: C, C++, Java, Scala, Perl e Python. Como neste EP haverá bastante processamento de texto, linguagens dinâmicas são recomendáveis (mas não obrigatórias).

6. Descontar-se-á por erros ortográficos, gramaticais, desvios de estilo e outros impropérios à língua.

3 Instanciação

A entrada do programa é um conjunto de cláusulas na Lógica Primeira Ordem sem quantificadores. Assume-se que todos os quantificadores existenciais já foram *skolemizados* e, implicitamente, todas as variáveis estão quantificadas universalmente. Assume-se que o domínio de quantificação é finito, o que será expresso pelo intervalo de quantificação de cada variável.

A entrada será expressa na seguinte linguagem de fórmulas. Uma fórmula será composta por

(a) Declaração de domínio de variáveis.

(b) Uma clásula contendo as variáveis declaradas e mais nenhuma.

(c) Uma lista de restrições de domínio. Esta lista pode estar vazia.

A sintaxe de entrada das fórmulas é a seguinte:

- As *variáveis* são expressas por palavras em letra maiúscula. Por exemplo, `X`, `RAINHA`, etc.

- *Declarações do domínio de variáveis* são da forma `<VARIÁVEL>: <NÚMERO> <NÚMERO>.` Por exemplo, `RAINHA: 1 8.` e `X: 0 10`.

  Nota: Para fins deste EP, não é necessário tratar de símbolos funcionais nem constantes. Uma constante pode ser representada por uma variável com domínio de quantificação unário. Por exemplo, a constante "pedro" pode ser representada pela variável PEDRO e com a restrição de domínio: `PEDRO: 1 1.`

- Os *predicados* são expressos por palavras em letras minúscula, opcionalmente seguidos de uma lista de argumentos entre parênteses, que podem ser apenas variáveis. Por exemplo, `pessoa(Y)`, `mae(PEDRO, X)`, etc.

- Uma *cláusula* é a disjunção de um ou mais predicados, que podem estar *negados*. Ela é expressa por uma lista de predicados separados por espaço e terminada por um ponto final. Um termo negado possui um `-` no início. Por exemplo: `-pessoa(X) mae(EVA,X).`

  Após o ponto final, as variáveis da cláusula podem estar restritas por uma *lista de restrições de domínio*, que consiste em zero ou mais *restrições de domínio* separadas por espaço. As restrições de domínio, se houver, são terminadas por um ponto final.

- Finalmente, *restrições de domínio* são expressões matemáticas, formadas pelos operados `+`, `-`, `=`, `>`, `<` e `!=`, sobre variáveis. Por exemplo, `X < Y` e `X-Y = W-Z`. Restrições devem possuir um valor `verdadeiro` ou `falso`.

Em cada linha da entrada, pode-se ter uma *declaração do domínio de variáveis* ou uma *cláusula* seguida por uma *restrição de domínio*. Todas as *declarações do domínio de variáveis* devem anteceder as *cláusulas*.

A *instanciação* desta entrada consiste em substituir as variáveis de uma cláusula por todos os valores possíveis na *declaração do domínio de variáveis*, desde que esses valores satisfação a *restrição de domínio*. Considere os exemplos a seguir:

1. Considere um tabuleiro em que queremos posicionar duas rainhas. O tabuleiro possui 3 linhas e 3 colunas. Uma fórmula representando que uma rainha posicionada num tabuleiro onde a linha e a coluna são iguais é uma rainha na diagonal principal é dada por:

   ```
   LINHA: 1 3.
   COLUNA: 1 3.
   RAINHA: 1 2.
   diagonalprincipal(RAINHA) -linha(RAINHA, LINHA) -coluna(RAINHA, COLUNA). LINHA = COLUNA
   ```

   A instanciação desta entrada é:

   ```
   diagonalprincipal(1) -linha(1, 1) -coluna(1, 1).
   diagonalprincipal(1) -linha(1, 2) -coluna(1, 2).
   diagonalprincipal(1) -linha(1, 3) -coluna(1, 3).
   diagonalprincipal(2) -linha(2, 1) -coluna(2, 1).
   diagonalprincipal(2) -linha(2, 2) -coluna(2, 2).
   diagonalprincipal(2) -linha(2, 3) -coluna(2, 3).
   ```

2. Considere a propriedade “foo” que pode ser satisfeita por 5 valores distintos, linearmente ordenados, e a propriedade “bar” que também pode ser satisfeita pelos mesmos 5 valores. A fórmula a seguir expressa que todos os elementos tem propriedade “foo” e que se um elemento possui a propriedade “foo” então os elementos posteriores na ordem devem ter a propriedade “bar”, é a seguinte:

   ```
   X: 1 5.
   Y: 1 5.
   foo(X).
   bar(Y) -foo(X). X < Y
   ```

   A instanciação desta entrada é:

```
foo(1).
foo(2).
foo(3).
foo(4).
foo(5).
bar(2) -foo(1).
bar(3) -foo(1).
bar(4) -foo(1).
bar(5) -foo(1).
bar(3) -foo(2).
bar(4) -foo(2).
bar(4) -foo(3).
bar(5) -foo(3).
bar(5) -foo(4).
```

Para cada cláusula, deve-se passar por todas as possíveis combinações dos valores das variáveis e avaliar as restrições de domínio para esses valores. Se as restrições forem satisfeitas para estes valores, deve-se criar uma nova instância da cláusula com o valor das variáveis substituído.

4 Implementação

O seu programa deve receber como entrada um conjunto de fórmulas contendo cláusulas e declarações de variáveis como definido na seção anterior e gerar um arquivo com a entrada instanciada.

Caso seja passada a opção `-c` ao seu programa, a saída deve estar no formato `cnf` compatível com os SAT Solvers utilizados no EP1. Ou seja, se esta opção for passada, você deverá fazer um pós-processamento numerando cada termo instanciado conforme os padrões cnf.

Para facilitar a implementação, você pode utilizar ferramentas geradores de *parsers* como o YACC ou o GNU Bison, ou ainda bibliotecas de tratamento de texto, como o Boost.Xpressive[1].

4.1 Bônus

Utilizando variáveis é possível escrever programas muito mais curtos para se resolver problemas. Valendo pontos extras, escreva as regras utilizadas no EP1 para resolver o problema do Sudoku utilizando a linguagem deste EP. Se for necessário, adicione novos operadores às *restrições de domínio*.

Deste modo, deve ser possível resolver um Sudoku passando estas regras pelo seu programa e concatenando-se ao resultado uma série de cláusulas que definem a posição dos números fixo do problema.

[1] `http://www.boost.org/doc/libs/1_38_0/doc/html/xpressive.html`