Lógica Computacional 117366
Descrição do Projeto
Formalização de Algoritmos para Ordenação por Inserção sobre Sequências Finitas

2 de maio de 2017

Profs. Mauricio Ayala-Rincón & Flávio L. C. de Moura

Observação: Os laboratórios do LINF têm instalado o *software* necessário para o desenvolvimento do projeto (PVS 6.0 com as bibliotecas PVS da NASA).

1 Introdução

Algoritmos de ordenação são fundamentais em ciência da computação. Neste projeto considerar-se-ão algoritmos de ordenção sobre o tipo abstrato de dados `finseq` como especificado no assistente de demonstração PVS.

O objetivo do projeto da disciplina é introduzir os mecanismos básicos de manuseio de tecnologias de verificação e formalização que utilizam técnicas dedutivas lógicas, como as estudadas na disciplina, para garantir que objetos computacionais são logicamente corretos.

2 Descrição do Projeto

Com base na *teoria* `sorting` (arquivos de especificação e prova `sorting.pvs` e `sorting_seq.pvs` e, `sorting.prf` e `sorting_seq.prf`, respectivamente) disponível na página da disciplina, especificada na linguagem do assistente de demonstração PVS (`pvs.csl.sri.com`) executável em plataformas Unix/Linux, os alunos deverão formalizar propriedades de especificações para ordenação por inserção sobre sequências finitas. O arquivo com as questões é denominado `binsertion`.

2.1 Busca em sequências de naturais

O objetivo é demonstrar, formalmente, que a especificação de ordenação por inserção abaixo é correta:

```
fs_insertion_sort(s): RECURSIVE finseq[nat] =
IF length(s) = 0 THEN s ELSE
insertion(first(s), fs_insertion_sort(rest(s)))
ENDIF
MEASURE length(s)
```

Nessa especificação, o mecanismo de inserção utilizado é especificado via a função `insertion` abaixo:

```
insertion (x, s): RECURSIVE finseq[nat] =
IF length(s) = 0 THEN add_first(x,s)
ELSIF x <= first(s) THEN add_first(x,s)
ELSE add_first(first(s), insertion(x,rest(s)))
ENDIF
MEASURE length(s)
```

3 Questões

Concretamente para a função `fs_insertion_sort` deve ser formalizado o seguinte lema:

O resultado de ordenar qualquer sequência finita de naturais utilizando `fs_insertion_sort` *é uma permutação (que preserva repetições) da lista original e que está ordenada não decrescentemente:*

```
fs_insertion_sort_works : LEMMA
FORALL (s):
     is_sorted?(fs_insertion_sort(s)) AND permutations(s, fs_insertion_sort(s))
```

A solução deste problema será dividida em 4 questões auxiliares que abordam propriedades referentes ao mecanismo de inserção utilizado no problema de ordenação.

Questão 01 *Ao inserirmos um natural numa sequência ordenada utilizando o mecanismos de inserção dado pela função* `insertion` *obtemos uma sequência ordenada.*

```
fs_insert_in_sorted_preserves_sort : CONJECTURE
FORALL (s: finseq[nat], x: nat):
   is_sorted?(s) IMPLIES is_sorted?(insertion(x,s))
```

Questão 02 *Ao aplicarmos* `fs_insertion_sort`*, obtemos uma sequência ordenada.*

```
fs_insertionsort_is_sorted: CONJECTURE
FORALL (s : finseq[nat]) : is_sorted?(fs_insertion_sort(s))
```

Questão 03 *Ao inserirmos um natural* `x` *em sequências* `s1` *e* `s2` *que são permutações, respectivamente no início e via o mecanismo de inserção* `insertion`*, obtemos permutações.*

```
fs_ins_and_add_in_perm_is_perm : CONJECTURE
FORALL (s1,
        (s2  | permutations(s1,s2)),
        x: nat) : permutations(add_first(x,s1), insertion(x,s2))
```

Questão 04 *A função* `fs_insertion_sort` *gera uma sequência que é permutação da sequência de entrada.*

```
fs_insertion_sort_is_permutations: CONJECTURE
FORALL (s) : permutations(s, fs_insertion_sort(s))
```

4 Etapas do desenvolvimento do projeto

Os alunos deverão definir grupos de trabalho limitados a **quatro** membros até o dia 8 de Maio. As aulas serão realizadas no LINF a partir do dia 8 de Maio para a turma A. A turma D continuará com aulas no LINF somente nas quintas-feiras.

O projeto será dividido em duas etapas como segue:

- Verificação das Formalizações. Os grupos deverão ter prontas as suas formalizações na linguagem do assistente de demonstração PVS e enviar via e-mail para o professor os arquivos de especificação e de provas desenvolvidos (`binsertionsort.pvs` e `binsertionsort.prf`)

até o dia **12.06.2017**. Na semana de **12-14.06.2017**, durante os dias de aula, realizar-se-á a verificação do trabalho para a qual os grupos deverão, em acordo com o monitor estagiário de docência e professor, determinar um horário (de 30 minutos) no qual todos membros do grupo deverão comparecer.
Avaliação (peso 6.0):

  - Um dos membros, selecionado por sorteio, explicará os detalhes da formalização em máximo 10 minutos.
  - Os quatro membros do grupo poderão complementar a explicação inicial em máximo 10 minutos.
  - A formalização será testada nos seguintes 10 minutos.

- Entrega do Relatório Final.
  Avaliação (peso 4.0): Cada grupo de trabalho devera entregar um Relatório Final inédito, editado em LaTeX, limitado a oito páginas (12 pts, A4, espaçamento simples) do projeto até o dia **23.06.2017** com o seguinte conteúdo:

  - Introdução e contextualização do problema.
  - Explicação da soluções.
  - Especificação do problema e explicação do método de solução.
  - Descrição da formalização.
  - Conclusões.
  - Referências.

Referências

[ARdM17] M. Ayala-Rincón and F.L.C. de Moura. *Applied Logic for Computer Scientists - computational deduction and formal proofs*. UTiCS, Springer, 2017.

[BvG99] S. Baase and A. van Gelder. *Computer Algorithms — Introduction to Design and Analysis*. Addison-Wesley, 1999.

[CLRS09] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Electrical Engineering and Computer Science Series. MIT press, third edition, 2009.