Lógica Computacional 117366
Descrição do Projeto
Formalização de Algoritmos para Ordenação Rápida (QuickSort)

24 de setembro de 2014

Prof. Mauricio Ayala-Rincón
Prof. Flávio L. C. de Moura

A estagiária de docência Ariane Alves Almeida (`arianealvesalmeida@gmail.com`) dará suporte aos alunos no desenvolvimento do projeto. Laboratórios do LINF têm instalado o *software* necessário (PVS 6.0 com as bibliotecas PVS da NASA).

1 Introdução

Algoritmos de busca e ordenação são fundamentais em Ciência da Computação. Busca é um mecanismo essencial em estruturas de dados e ordenação é relevante para diminuir o tempo de busca em diversas estruturas de dados. Neste projeto considerar-se-ão algoritmos de ordenção sobre o tipo abstrato de dados `list` 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`, composta de formalizações para auxiliar na prova de correção de diferentes algoritmos de ordenação e provas auxiliares, este projeto trata dos arquivos de especificação e prova do algoritmo de ordenação rápida, que são, respectivamente, `quicksort.pvs` e `quicksort.prf`. Os arquivos estão disponível na página da disciplina, especificados 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 rápida em listas de naturais.

2.1 Ordenação rápida: *Quicksort*

Diversas noções e lemas auxiliares, demonstrados integralmente na teoria `sorting`, são necessários. Esses elementos são a base para o desenvolvimento de uma fromalização da correção de outros algoritmos de ordenação.

Em particular, neste projeto, o objetivo é demonstrar, formalmente, que a especificação de ordenação rápida abaixo é correta:

```
quick_sort (l): RECURSIVE list[nat] =
    CASES l OF
      null: null,
      cons (x, r): append
       (quick_sort (leq_elements (r,x)),
         cons (x, quick_sort (g_elements (r, x))))
```

```
    ENDCASES
  MEASURE length (l)
```

Nessa especificação, o mecanismo de ordenação se dá ao concatenar uma sublista com os elementos da lista original menores ou iguais a um pivô, este pivô, e a sublista de elementos maiores que ele. A construção dessas sublistas é dada através das funções `leq_elements` e `g_elements` especificadas abaixo:

```
 leq_elements (l,p): RECURSIVE list[nat] =
    CASES l OF
      null: null,
      cons (x, r): if x<= p then
                    cons (x, leq_elements (r, p))
                   else
                    leq_elements (r, p)
                   endif
    ENDCASES
  MEASURE length (l)

 g_elements (l,p): RECURSIVE list[nat] =
    CASES l OF
      null: null,
      cons (x, r): if x > p then
                    cons (x, g_elements (r, p))
                   else
                    g_elements (r, p)
                   endif
    ENDCASES
  MEASURE length (l)
```

3 Questões

Concretamente para a função `quick_sort` deve ser formalizada a seguinte conjectura:
Questão 03 *O resultado de ordenar qualquer lista de naturais utilizando* `quick_sort` *é uma permutação (que preserva repetições) da lista original que está ordenada não decrescentemente:*

```
quick_sort_works : CONJECTURE
FORALL (l: list[nat]):
     is_sorted?(quick_sort(l)) AND permutations(l, quick_sort(l))
```

Como questões auxiliares para conseguir tal formalização, será necessário provar as seguintes propriedades referentes ao mecanismo de seleção de elementos menores ou iguais e maiores utilizados na abordagem de ordenação.
Questão 01 *O tamanho da lista que contém os elementos e uma lista menores ou iguais a um pivô, tem no máximo o tamanho da lista original. O mesmo vale para a lista com elementos maiores que o pivô.*

```
leq_elements_size : LEMMA
FORALL (l1 : list[nat]) :
   length(leq_elements(l1,x)) <= length(l1)

g_elements_size : LEMMA
FORALL (l1 : list[nat]) :
   length(g_elements(l1,x)) <= length(l1)
```

Questão 02 *Quando procuramos por um elemento menor ou igual ao pivô selecionado na lista de elementos menores iguais a ele, a quantidade de ocorrências deste elemento nesta lista será a mesma da lista original. E quando procuramos por um elemento maior que o pivô, a quantidade de ocorrências é zero nessa lista. O mesmo ocorre ao procurarmos por um elemento maior que o pivô na lista que contém os elementos maiores que ele*

```
same_occurrence_leq: LEMMA
FORALL (l1:list[nat], x,p:nat):
  x <= p IMPLIES
  ((occurrence(l1)(x) = occurrence (leq_elements(l1,p))(x))
  AND
  (occurrence (g_elements(l1,p))(x) = 0))

same_occurrence_g: LEMMA
FORALL (l1:list[nat], x,p:nat):
  x > p IMPLIES
  ((occurrence(l1)(x) = occurrence (g_elements(l1,p))(x))
  AND
  (occurrence (leq_elements(l1,p))(x) = 0))
```

4 Etapas do desenvolvimento do projeto

Os alunos deverão definir os grupos de trabalho limitados a **três** membros até o dia 29 de Setembro. Exceto pelo dia da segunda prova, 3 de Dezembro de 2014, as aulas serão realizadas no LINF a partir do dia 24 de Setembro.

O projeto será dividido em duas etapas como segue:

- A primeira etapa do projeto e a de Verificação das Formalizações. Os grupos deverão ter prontas as suas formalizações na linguagem do assistente de demonstração PVS e enviar via e-mail à estagiária com cópia para o professor os arquivos de especificação e de provas desenvolvidos (`quicksort.pvs` e `quicksort.prf`) até o dia **10.11.2014**. No dia **12.11.2014**, durante o horário de aula, realizar-se-á a verificação do trabalho para a qual os grupos deverão, em acordo com o monitor e professor, determinar um horário (de uma hora) 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 20 minutos.
  - Os quatro membros do grupo poderão complementar a explicação inicial em máximo 10 minutos.

  - A formalização será testada nos seguintes 30 minutos.

- A segunda etapa do projeto consiste da apresentação dos resultados finais e conclusões do estudo do problema.
  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 **17.11.2014** 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.