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

24 de maio de 2013

Prof. Mauricio Ayala-Rincón

A estagiária de docência Ariane Alves Almeida (`arianealvesalmeida@gmail.com`) dará suporte aos alunos no desenvolvimento do projeto. Laboratórios do LINFO tem instalado o software necessário (PVS 6.0 com as livrarias 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 diminuição do tempo na 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 lógicamente corretos.

2 Descrição do Projeto

Com base na *teoria* PVS `sorting` (arquivos de especificação e prova `sorting.pvs` e `sorting.prf`) 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 em listas de naturais.

2.1 Busca em listas de naturais

Para a especificação de um algoritmo de ordenação por seleção do máximo, que utiliza intercâmbio e reversão de listas, abaixo

```
sorting_min(l : list[nat]) : RECURSIVE list[nat] =
 IF l`length < 2 THEN l
 ELSE LET rev_sw_min = reverse(switching_min(l)) IN
       cons(car(rev_sw_min), sorting_min(cdr(rev_sw_min)))
 ENDIF
MEASURE length(l)
```

a teoria inclui a formalização da sua correção expressa no seguinte lemma:

```
sorting_min_work  : LEMMA
FORALL (l: list[nat]): is_sorted?(sorting_min(l)) AND
permutations(l, sorting_min(l))
```

Para obter essa formalização são necessários diversos lemmas auxiliares também demonstrados integralmente na teoria `sorting`. Esses elementos serão 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 uma especificação de ordenação por inserção abaixo é correta.

```
insertion_sort(l): RECURSIVE list[nat] =
IF null?(l) THEN null ELSE
insert(car(l), insertion_sort(cdr(l)))
ENDIF
MEASURE length(l)
```

Nessa especificação, o mecanismo de inserção utilizado é especificado via a função `insert` abaixo.

```
insert (x, l): RECURSIVE list[nat] =
IF null?(l) THEN cons(x,null)
ELSIF x<= car(l) THEN cons(x,l)
ELSE cons(car(l), insert(x,cdr(l)))
ENDIF
MEASURE length(l)
```

3 Questões

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

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

Como questões auxiliares para conseguir tal formalização, será necessário provar as seguintes propriedades referentes aos mecanismos de inserção utilizados na abordagem de ordenação.
Questão 01 *Ao inserirmos um natural numa lista, o comprimento da lista é incrementado em um.*

```
insert_size: CONJECTURE
FORALL (l: list[nat], x: nat):
       insert(x,l)`length = l`length + 1
```

Questão 02 *Ao inserirmos, utilizando a função* `insert`*, um elemento numa lista ordenada, obtemos uma lista ordenada.*

```
insert_in_sorted_preserves_sort : CONJECTURE
FORALL (l: list[nat], x: nat):
   is_sorted?(l) IMPLIES is_sorted?(insert(x,l))
```

4 Etapas do desenvolvimento do projeto

Os alunos deverão definir grupos de trabalho limitados a **quatro** membros até o dia 12 de Junho. Excepto pelo dia da segunda prova, 1 de Julho de 2013, as aulas serão realizadas no LINFO desde o dia 17 de Junho.

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 (`sorting.pvs` e `sorting.prf`) até o dia **10.07.2013**. Na semana de **15-17.07.2013**, durante os dias 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 **17.07.2013** 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.