# Lógica Computacional 117366
# Descrição do Projeto
# Formalização de Algoritmos para Ordenação por Fusão de Listas

24 de setembro de 2013

Professores Mauricio Ayala-Rincón e Flávio L.C. de Moura

A estagiária de docência Ana Cristina Rocha Oliveira Valverde (anacrismarie@gmail.com) e o monitor Thiago Mendonça Ferreira Ramos (thiagomendoncaferreiraramos@yahoo.com.br) darão 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 a 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 para o 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* PVS `sorting` (os arquivos de especificação e prova são, respectivamente, `mergesort.pvs` e `mergesort.prf`, mas outros arquivos estão também disponíveis nessa teoria `sorting`) disponível na página da disciplina, especificada na linguagem do assistente de demonstração PVS (`pvs.csl.sri.com`) executável em plataformas compatíveis com Unix/Linux os alunos deverão formalizar propriedades de especificações para ordenação por mescla ou fusão de listas de naturais (*merge sort*).

### 2.1 Ordenação em listas de naturais

Para a especificação de um algoritmo de ordenação por seleção do elemento mínimo de uma lista que utiliza intercâmbio de elementos e a reversão de listas, dada 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 via o seguinte lema:

```
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 lemas auxiliares também demonstrados integralmente na teoria `sorting`. Esses elementos serão a base para o desenvolvimento de uma formalização da correção de outros algoritmos de ordenação.

Em particular, neste projeto o objetivo é demonstrar formalmente, que a especificação de ordenação por mescla abaixo é correta.

```
merge_sort(l) : RECURSIVE list[nat] =
 IF length(l) <= 1 THEN l
 ELSE merge(merge_sort(prefix(l,floor(length(l) / 2))),
                merge_sort(sufix(l,floor(length(l)/2) + 1)))
 ENDIF
MEASURE length(l)
```

Nessa especificação, o mecanismo de mescla utilizado é especificado via a função `merge` abaixo.

```
merge(l1, l2 : list[nat]) : RECURSIVE list[nat] =
IF null?(l1) OR null?(l2) THEN append(l1, l2)
ELSIF car(l1) <= car(l2) THEN cons(car(l1), merge(cdr(l1),l2))
ELSE cons(car(l2), merge(l1, cdr(l2)))
ENDIF
MEASURE length(l1) + length(l2)
```

# 3 Questões

Concretamente para a função `merge_sort` devem ser formalizadas as seguintes conjecturas:
**Questão 03** *O resultado de ordenar qualquer lista de naturais utilizando* `merge_sort` *é uma permutação (que preserva repetições) da lista original e que está ordenada não decrescentemente:*

```
- merge_sort_works: CONJECTURE
  FORALL (l : list[nat]) :
    permutations(merge_sort(l), l) AND  is_sorted?(merge_sort(l))
```

Como questões auxiliares para conseguir tal formalização, será necessário provar as seguintes propriedades referentes aos mecanismos de mescla utilizados na abordagem de ordenação.
**Questão 01** *Ao aplicarmos o algoritmo de mescla/fusão* `merge` *a duas listas, a lista resultante preserva o comprimento das listas originais.*

```
- merge_size: CONJECTURE
  FORALL (l : list[nat]) :
    length(merge(l1,l2)) = length(l1) + length(l2)
```

Esse resultado deverá ser utilizado para concluir que `merge` gera uma permutação das listas originais.
**Questão 02** *Ao mesclarmos duas listas ordenadas utilizando a função* `merge`*, obtemos uma lista ordenada.*

```
- merge_of_sorted_is_sorted : CONJECTURE
  FORALL (l1, l2 : list[nat]) :
    is_sorted?(l1) AND is_sorted?(l2) IMPLIES is_sorted?(merge(l1,l2))
```

# 4 Etapas do desenvolvimento do projeto

Os alunos deverão definir grupos de trabalho limitados a **quatro** membros até o dia 7 de outubro. Algumas aulas serão realizadas no LINF a partir do dia 23 de setembro.

O projeto será dividido em duas etapas de avaliação como segue:

- A primeira etapa do projeto é 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 **11 de Novembro de 2013, antes da aula**. Na semana de **11-14 de Novembro de 2013**, durante os dias de aula, incluindo também outros horários adicionais, realizar-se-á a verificação do trabalho para a qual os grupos deverão, em acordo com o monitor e professor, determinar um horário (de 40 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 no máximo 20 minutos.
  - Os quatro membros do grupo poderão complementar a explicação inicial em no máximo 10 minutos.
  - A formalização será testada durante a apresentação.
- 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 deverá entregar um Relatório Final inédito do projeto, editado em LaTeX, limitado a oito páginas (12 pts, A4, espaçamento simples) até o dia **25 de Novembro de 2013** com o seguinte conteúdo:
  - Introdução e contextualização do problema.
  - Explicação das soluções.
  - Especificação do problema e explicação do método de solução.
  - Descrição da formalização.
  - Conclusões.
  - Referências.