

Trabalho Final

- ◆ Definição, modelação e verificação de um sistema concorrente de pequena complexidade:
 - CCS: Workflow, Protocolo Comunicação (tcp, ...).
 - PI: Handover com k-dispositivos e n-células.
 - SPI: Algoritmo de distribuição de chaves.
- ◆ Estrutura do trabalho
 - Descrição informal do problema proposto
 - Formalização em CCS/PI/SPI
 - Verificação de algumas propriedades ilustrativas
- ◆ 20 páginas A4 Max.

Enunciado Base

- ◆ O uso generalizado de processadores multicore está a abrir cada vez mais perspectivas ao uso de algoritmos paralelos / concorrentes
- ◆ A questão fundamental da programação concorrente é como disciplinar o uso de recursos (controle de concorrência)
- ◆ Tradicionalmente, o controle de concorrência assenta da definição de secções críticas
 - fragmentos de código que se sabe serem executados apenas por um thread de control
 - uso de locks / semáforos / synchronized / atomic, etc

Enunciado Base

- ◆ O uso de locks é muitas vezes considerado ineficiente porque: discutir ?
- ◆ Herlihy e outros propuseram a adoção de algoritmos “lock-free”.
- ◆ Relacionados com implementações de baixo nível (spin-locks) e com transações otimistas
- ◆ Suportados por instruções CAS, executadas em user mode, em vez de locks, executados em system mode

Enunciado Base

- ◆ Propriedades básicas de um algoritmo lock-free
- ◆ lock freedom
 - “An algorithm is **lock-free** if every step taken achieves global progress (for some sensible definition of progress)”
- ◆ wait freedom (\Rightarrow lock-freedom)
 - “An algorithm is **wait-free** if every operation has a bound on the number of steps it will take before completing”

Enunciado Base

- ◆ Os algoritmos lock-free oferecem em geral uma maior performance, mas são muito subtis e difíceis de garantir correctos (daring concurrency) !
- ◆ Essência dos mecanismos (cf. optimistic ...)
 - preparar para executar a operação
 - tentar executar a operação (e.g., actualizar dados)
 - se não houver sucesso, tentar de novo :) !
- ◆ Abordagens à verificação
 - debugging (impossível)
 - model-checking
 - verificação formal (difícil...)

Enunciado Base

- ◆ Considere um algoritmo lock free para definir um buffer duplo, usando uma lista ligada.
 - Pesquise bibliografia, para encontrar uma solução
 - Programe a sua solução em Java, usando a primitiva CAS (CompareAndSwap)
- ◆ Construa um modelo em cálculo PI (porquê) do seu algoritmo. O modelo terá que abstrair alguns aspectos, e a fila terá que ser limitada ...
- ◆ Exprima e verifique (SLMC) propriedades de correcção, incluído wait-freedom, e lock-freedom
- ◆ Justifique que o seu programa está correcto.