Modelação e Validação de Sistemas Concorrentes

Mini Projeto

Pedro Valente
Nº 50355

Rafael Gameiro
Nº 50677

Introdução

Este trabalho consistiu na implementação de um buffer de números inteiros partilhados com capacidade limitada. O acesso ao buffer foi feito através de 4 threads, 2 threads produtoras e outras 2 consumidoras. Os produtores colocavam valores no fim do buffer a fim de os consumidores os acederem. O restante documento apresentará: quais as propriedades de concorrência que o algoritmo implementado respeita; uma simples descrição da implementação feita, sendo esta acompanhada pelo código em anexo.

Propriedades

Exclusão mútua

Exclusão mútua ocorre quando o acesso a uma zona de código, denominada região critica, é feita apenas por um único processo mesmo que vários processos tencionem aceder a essa região. O nosso algoritmo cumpre esta propriedade uma vez que o acesso é região critica é feito por um único processo de cada vez, através de mecanismos de controlo de acesso chamados *mutexes.* Os *mutexes* permitem que a secção de código em que se encontram seja apenas acessível pelo processo que adquirir o seu *lock.* Desta forma, um processo que queira aceder a região critica e não tenha o acesso ao *lock* terá de aguardar que o processo que o detém liberte-o.

Deadlock freedom

Deadlock acontece numa situação em que dois processos que dependem um do outro fiquem parados durante o seu momento de execução à espera um do outro levando a um bloqueio geral. Neste caso, *Deadlock freedom* ocorre quando vários processos pretendem aceder à região critica, é garantido que um deles acabará por aceder. Na nossa implementação essa propriedade é conseguida uma vez que todos os processos estão em competição para aceder à região critica, mas o nosso algoritmo permite controlar a ordem em que esses processos a acedem. Desta forma a seu tempo, cada processo consegue aceder à região critica.

Starvation freedom

Starvation acontece quando um processo nunca é executado devido a outros processos com maior prioridade o impedirem de ser executado. *Starvation freedom* possibilita que a ordem de chegada dos processos que pretendem aceder a uma região critica seja respeitada. De forma a respeitar a ordem de chegada dos vários processos, na nossa implementação utilizámos uma adaptação do algoritmo de

bakery. Uma vez que o algoritmo de *bakery* cumpre a propriedade de *starvation freedom* o mesmo acontece com a nossa implementação.

Implementação

A implementação do algoritmo foi feita na linguagem Rust. O buffer usado consistiu numa *struct* que continha posições para cabeça e cauda, assim como o número atual de elementos a processar. O acesso ao buffer foi feito exclusivamente na região critica e com o uso da biblioteca *Mutex*. Para gerir a ordem de execução de cada thread e possibilitar *fairness* e progresso, recorreu-se ao algoritmo de *bakery.* O algoritmo necessitou de uma estrutura para guardar a ordem de execução de cada thread, em que o acesso a estrutura foi também por meio de *mutexes*.