4yp

Thomas Aston

May 16, 2024

Contents

- Change figure linkings
-

1 Modelling and analysing implementations of locks

1.0.1 Starvation Freedom

Starvation freedom is a liveness property that states that every thread that attempts to acquire the lock eventually succeeds [1]. It requires that any thread attempting to gain the lock must can only be bypassed by other threads a finite number of times.

One common approach to checking infinite properties in CSP is to hide some (in this case internal) channels and then check that the model does not diverge. This approach does not work here: consider a starvation-free lock which uses busy waiting (repeatedly testing if the lock is available). We have that hiding the internal communications results in a divergence, however the lock is, by definition, starvation-free; an example of this is the Peterson lock $\boxed{\text{link to model}}$.

Roscoe and Gibson-Robinson showed that every infinite traces property that can be captured by CSP refinement can also be captured by a finite-traces refinement check when combined with the satisfaction of a deterministic Büchi automaton [2]. However, we can show that such an automaton must be non-deterministic.

Let us without loss of generality consider 2-threaded locks.

We can first conclude that the Büchi automaton has at least one accepting state reachable while T.0 holds the lock and T.1 attempts to acquire the lock. By the definition of starvation freedom, the automaton should be satisfied if T.0 never releases the lock. The automaton therefore must have at least one accepting state where T.0 holds the lock and the repeated actions of the threads cause (a subset of) the accepting states to be visited infinitely often.

Now we consider the TAS lock from before. This is clearly not starvation free as the first thread to communicate on getAndSet once the lock becomes available will acquire it. As a result T.0 can bypass T.1 an infinite number of times, hence the TAS lock is not starvation free. We have from the above that there must exist some accepting state in the automaton which is reachable when T.0 holds the lock and T.1 is attempting to obtain the lock. Since T.1 only has one available state when T.0 holds the lock (communicating a getAndSet.L.0.T.1.false.false), we have that this accepting state must be reachable whenever T.0 bypasses T.1. As a result, we have that T.0 can obtain the lock an infinite number of times, and each of these bypasses can result in a visit to an accepting state of the automaton. We therefore have that the Büchi automaton accepts this run despite the both the run and the TAS lock not being starvation-free. This is a contradication and hence we have that no such deterministic automaton can exist.

We hence have that no deterministic Büchi automaton can accurately capture starvation freedom and hence we cannot test directly for starvation-freedom through a standard FDR refinement check.

1.1 First-come-first-served

References

[1] Hanno Nickau and Gavin Lowe. *Concurrent Algorithms and Data Structures Lecture Notes*. 2023. URL: https://www.cs.ox.ac.uk/teaching/courses/2023-2024/cads/.

[2] A. W. Roscoe and Thomas Gibson-Robinson. *The relationship between CSP, FDR and Büchi automata*. 2016. URL: https://www.cs.ox.ac.uk/files/8301/infinitec.pdf.