INF214 - Assignment 1

Viet Hoang Nguyen

September 22, 2018

Task 1

In this task, we are to identify two sets of variables; $\mathcal{V}$, the set of shared variables that is being read from, and $\mathcal{W}$, the set of shared variables that is being written to.

Enqueue

The variables that are being read from by the enqueue function are *rear* and $\_size$, giving us $\mathcal{V} = \{rear, \_size\}$. The variables being written to by the function are $rear \rightarrow next$, $\_size$ and *head*. This gives us $\mathcal{W} = \{head, rear, \_size\}$.

The inference rules gives ut $\mathcal{V}_1 \cap \mathcal{W}_2 = \{rear, \_size\}$, telling us that the processes would interfere. Deriving from this we should not call enqueue from multiple threads concurrently, as this can cause overwriting.

Dequeue

The variables that are being read from by the dequeue are *head* and $\_size$, giving us $\mathcal{V} = \{head, \_size\}$. The variables being written to by the function are *head*, *rear* and $\_size$, giving us $\mathcal{W} = \{head, rear, \_size\}$.

Looking at the interference rules again, we get the following interference:$\mathcal{V}_1 \cap \mathcal{W}_2 = \{rear, \_size\}$. Calling dequeue from multiple threads would not be wise here either.

Node Iterator Operators

The iterator have two different operations, the ++ operator and the * operator. For both operations, the only thing that can be considered as a shared variable here is the node that *it* is pointing to.

For the ++ operator, the node which *it* points to can be deleted, leaving us with a dangling pointer. This gives us $\mathcal{V} = \{it \rightarrow next\}$ and $\mathcal{W} = \emptyset$

Looking at the * operator, having multiple iterators could problematic in the sense that data races could arise as a result of multiple iterators accessing and modifying the same node concurrently. This gives us $\mathcal{V} = \{it \rightarrow data\}$ and $\mathcal{W} = \emptyset$

Concurrency

We have already seen that doing multiple calls of either enqueue or dequeue concurrently will not lead to any good results. Since enqueue and dequeue write and read from the same variables, it is safe to assume that calling boht of these functions concurrently is not a good idea either.

In terms of the node iterator operations, it should be safe to call on these operators as long they are not shared. Calling either enqueue or dequeue and the ++ operator concurrently could lead to errors because enqueue or dequeue could make the ++ operator return the wrong reference. It would probably be the same for the * operator as it would result in the return of or modification to the wrong node.