Group Assignment #2: Parallel Swap

Jordon Dornbos
Hao Sun

Our model uses a macro to define N, the amount of numbers, and a global array to store the numbers themselves. In addition to this, it also stores a boolean and a count variable. These two variables are used to form a mutex.

For the actual swap process we've created a proctype that creates N threads of the process. The swap process sets i to the PID and then generates a random number and assigns it to j. Next, the process needs to acquire the lock in order to avoid race conditions in swapping the values. In an atomic section the process checks if locked is false. If it is false it sets it to true and increments count. If it's true it will wait until it's false. After this section the process has the lock if it is executing.

To ensure that this critical section is valid we have an assert statement to verify that count equals 1. This means that only one process is in the critical section, as expected. Then we perform the swap of values at positions i and j in the number array. Then the lock is released by decrementing count and setting locked to false.

To check our model we have two LTL expressions. The first LTL expression ensures safety: ![]mutex. In our code we have a macro definition called mutex that ensures count isn't greater than 1. This verifies that only one process is in the critical section at a given time. The LTL checks for the negation of that claim to ensure it is never broken. The second LTL expression we use ensures liveness: !<>cs. When a process enters the critical section it sets a boolean named cs to true. Therefore, the LTL expression <>csp only holds if the process eventually enters its critical section. Once again, our LTL checks for the negation of this claim to ensure it is never broken. We verified these LTL expressions in Spin to ensure that our model is valid.

Extra Credit: After you verify the safety and liveness properties, revise your Promela code such that each proctype performs the swapping several times.

- Are the LTL properties still satisfied? Explain what you experience.

Yes, the LTL properties are still satisfied. The processes still ensure liveness and safety. The swapping runs just like when it runs once, and doesn't result in duplicate numbers.

- Should the proctypes be synchronized after each round of swapping?

Yes, the data is in the shared memory, so after each round of swapping all the processes are in sync. This means that no matter how many times the program swaps, the array won't have duplicate numbers.