CS444 Concurrency Assignment 2

Mark Bereza and Morgan Patch

Our solution to the dining philosophers problem is a Java implementation of the Chandy/Misra solution [1]. Put briefly, each philosopher gets an ID (from 0 to 4) and during initialization, forks under contention are given to the philosopher with the smaller ID. Each fork has a property of cleanliness that is binary: forks are either clean or dirty and at the outset all the forks are dirty. After the initialization stage, a clean fork is one that is locked by a philosopher thread and a dirty fork is one that is not locked. During getforks(), philosophers that do not have two forks send requests to their neighbors for their forks. In practice, these requests are simply trylock(). As a rule, if the fork held by a philosopher is clean, the fork is held even if it is requested by another. On the other hand, if the fork is dirty, it is first cleaned (locked) and then given to the philosopher requesting it. After eating, during putforks(), used forks are made dirty. By maintaining state for whether forks were just used to eat, the algorithm gives priority to philosophers who have not recently eaten, helping to prevent starvation.

The solution can be verified simply by running it by using the command "make run". The program prints the state of the forks and the philosophers at every step and additionally prints timestamps for when philosopher threads start/stop eating to a file called timing.txt. After letting it run for about an hour, I could see from the timestamps than multiple philosophers were eating simulatenously since a second philosopher began eating before the firsts' done eating message was printed, and this is common in the output. Additionally, by looking at the stdout output, we were able to verify that the program did not deadlock since it was still printing messages for different philosopher threads eating and giving up forks. Additionally, throughout the output, each philosopher appears to eat regularly, with all five IDs found eating in every 50 or less lines of output, all the way through to when we finally terminated the program. Additionally, the output identifies philosophers by both ID and a name of real philosopher. In this way, we have verified that our program meets all of the critera of the assignment.

REFERENCES

[1] K. M. Chandy and J. Misra, "The drinking philosophers problem," *ACM Transcations on Programming Languages and Systems*, vol. 6, 1984.