Alex, Carter, Nima
Mar 27th, 2024

# Solution

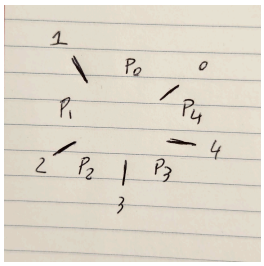
We approached all the problems by coding SMV files (Symbolic Model Verification file) (using test driven development testing our solution against NuSMV). The zip contains:

- `phil.smv`
- `phil.extended.smv`

Please check the `README.md` file on how to run all these files.

## Approach (phil.smv)

Before diving in, we first drew a diagram of the problem to better visualize. Here is the diagram:




We utilized test driven development to solve phil.smv

1. We first moved the SPECs from the phil module to the main module
2. We then developed the SPEC for chopsticks setting that each chopstick can be used by one philosopher at a time.
3. We then tested the solution against the NuSMV
4. We then developed the SPC for each philosopher can eat eventually
5. We then tested the solution against the NuSMV
6. We then traced the failure path outputted by NuSMV to develop the fairness constraints.
7. We then developed a sushi object which creates a polling solution in which "*a plate of sushi is passed around by the philosophers to see in order if they want to eat or not*"
8. We then iteratively achieved our final solution using NuSMV
9. After the change to the assignment, we moved the SPECs back into the phil module. We also decreased the program's reliance on FAIRNESS.

## Approach (phil.extended.smv)

We utilized test driven development to solve phil.extended.smv

1. We first developed the SPEC for non-blocking first creating that `philosopher0` can always request to eat.
2. We tested the solution against NuSMV creating the rest of the non-blocking SPEC for the rest of the philosophers.
3. We then implemented the no strict sequencing for `philosopher0` against the rest of the philosophers.

4. We tested the solution against NuSMV creating the rest of the no strict sequencing SPEC for the rest of the philosophers.
5. After the change to the assignment, we moved some SPECs back into the phil module, and changed others.