RAMDE

ISEP - DEI - MESCC

Dinning Philosophers Problem

Carlos Rijo - 1101626

[Introduction 3](#_Toc1)

[Requirements 3](#_Toc2)

[Development 4](#_Toc3)

[Forks State Definition 5](#_Toc4)

[State Definition 6](#_Toc5)

[Variable Definition 6](#_Toc6)

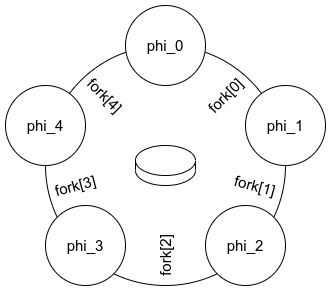
[Conclusion 7](#_Toc7)

## Introduction

In this assignment it was proposed to implement the Dinning Philosophers Problem in NuSMV.

The dining philosophers problem describes a scenario where five philosophers sit at a round table, with a big bowl of spaghetti and with only five forks.

Each philosopher has access to a fork at his left and at his right. A visual representation of the table where the philosophers are sitting is presented below.



Each philosopher has two main actions: he can either think or eat.

When a philosopher is thinking he remains silent and does not interact with the environment. However when a philosopher gets hungry he will try to eat. Since he needs both forks to eat, first he will need to pick up its left fork (if/when available) ant then its right fork (if/when available).

When this requirements are met, the philosopher can start eating (not being limited by the remaining amounts of spaghetti or stomach space). When he becomes full the philosopher puts both the forks in the table (first left fork and then right fork) and gets back to thinking.

The main goal of this problem is how to design and implement a solution in which no philosopher will starve. Starvation can happen when all philosophers pick up the left fork and wait for the right fork to be available, assuming that no philosopher can know when others may want to eat or think.

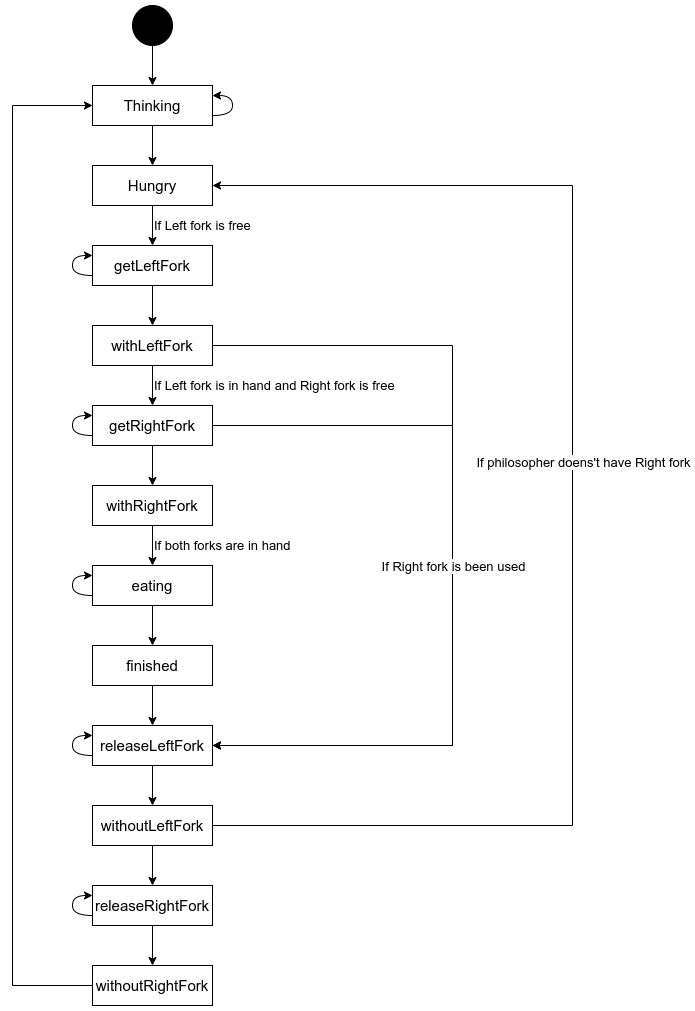
### Requirements

This problem has the following requirements:

* The philosopher shall have both forks to eat
* The philosopher shall be able to eat
* While philosopher is hungry and when the left fork is available, the philosopher shall pick the left fork first
* While philosopher is hungry and when the right fork is available, the philosopher shall pick the right fork
* The philosopher shall pick the right fork after getting the left fork
* The solution shall avoid starvation
* The philosophers shall not know what the other philosophers are doing or planing to do.

## Development

This solution has five philosophers (with its own id and with a left fork and right fork assigned), five forks (that can be free or assigned to a philosopher) and the following states (described in the next table and diagram):



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **State Name** | **Transition from state** | **Transition to state** | **Transition event** | **Description** |
| Thinking | withoutRightFork | Hungry | - | Philosopher is thinking |
| Hungry | Thinking | getLeftFork | If left fork is free | Philosopher is hungry |
| Eating | withRightFork | Finished | - | Philosopher is eating |
| Finished | Eating | releaseLeftFork | - | Philosopher has finished eating |
| getLeftFork | Hungry | withLeftFork | - | Philosopher will try to pick the left fork |
| getRightFork | withLeftFork | withRightFork | - | Philosopher will try to pick the right fork |
| releaseLeftFork | If Right fork is been used |
| withLeftFork | getLeftFork | getRightFork | If Left fork is in hand and Right fork is free | Philosopher has left fork |
| releaseLeftFork | If Right fork is been used |
| withRightFork | getRightFork | Eating | - | Philosopher has right fork |
| releaseLeftFork | Finished | withoutLeftFork | - | Philosopher will release the left fork |
| releaseRightFork | withoutLeftFork | withouRightFork | - | Philosopher will release the right fork |
| withoutLeftFork | releaseLeftFork | releaseRightFork | - | Philosopher does not have left fork |
| Hungry | If philosopher does not have Right fork |
| withouRightFork | releaseRightFork | Thinking | - | Philosopher does not have right fork |

At the beginning all philosophers will start thinking and so all forks will be free.

When a philosopher is hungry we will try to pick up its left fork. If the fork is available we will get the left fork, but if not he will stay in a hungry state and wait until the fork is free.

After picking up the left fork we will try to pick the right fork and repeating the same process as before.

Having both forks, the philosopher will eat for a undetermined set of time.

The philosopher, when finished eating, will release the left fork, then release the right fork and start thinking again.

In this design the philosopher can not pick up or release the fork automatically. He can get/release the fork in his hand in the next step or in any future step.

### Forks State Definition

As stated before, the forks can be free or assigned to a philosopher.

Any fork will be free when the philosopher is in the state after of release (withoutLeftFork and withoutRightFork).

As for the assignment of the philosopher ID, the left fork will be assigned if the left fork is free and when the philosopher has the fork in its hand.

The right fork, will be assigned if the left fork is in the philosophers hand, if the right fork is free and when the philosopher has the fork in its hand.

The fork will not change its ID in any other situation.

### State Definition

Following the requirements, it was defined the transitions:

* while the philosopher is thinking, he can opt by continuing to do so, or he can get hungry
* if the philosopher is hungry and both forks are free get Left Fork.
* while getting the left fork, the philosopher can continuing to do so, or pick the left fork
* if the philosopher has left fork and the right fork is free get right Fork.
* while getting the right fork, the philosopher can continuing to do so, or pick the right fork
* if the philosopher has both left and right fork he can start eating.
* while eating, the philosopher may decide to continue eating, or he is full and done
* if releasing the left fork, the philosopher can continuing to do so, or leave the left fork
* if the philosopher does not have left fork he can release the right fork.
* if releasing the right fork, the philosopher can continuing to do so, or leave the right fork

To avoid starvation, it was implemented the idea of an altruist philosopher. When a philosopher picks the last available fork and they reach a deadlock, in this scenario the philosopher that picked the last fork will have an altruist behavior and he will release the fork giving the opportunity to other philosophers eat.

With this in mind, the next conditions were added:

* if the philosopher has left fork and the right fork is not free release Left Fork
* if the philosopher does not have left and right fork go to hungry.

### Variable Definition

To help verify the system the next variables were defined:

* DEADLOCK
  + All the philosophers have the left fork
* any\_philosopher\_eating
  + One or more philosophers are eating
* all\_pickup
  + All forks are being used
* total\_phil\_eating
  + Number of philosophers that are currently eating

## Conclusion

To test the requirements described above, it was implemented verification within NuSMV.

The next table will demonstrate the test made and its results.

|  |  |  |
| --- | --- | --- |
| **To Test** | **Verification** | **Result** |
| A philosopher needs both forks to eat | CTLSPEC  AG (state = eating) -> ((leftFork = id) & (rightFork = id)) | **True** |
| If a philosopher is hungry and wants to eat, then he will eventually eat | CTLSPEC  AG (state = hungry) -> AF (state = eating) | **True** |
| If a philosophers is eating, its neighbors will not eat at the same time. | LTLSPEC  G (phi\_0.state = eating) -> F !(phi\_1.state = eating)  .  .  . | **True** |
| If a philosophers is eating, its neighbors can not eat at the same time. | LTLSPEC  G !(phi\_0.state = eating & phi\_1.state = eating)  .  .  . | **True** |
| No more than two philosophers can eat at the same time. | LTLSPEC  G (total\_phil\_eating <=2) | **True** |
| If every philosopher holds his left fork, sooner or later somebody will get the opportunity to eat. Fix Deadlock | LTLSPEC  G (all\_pickup -> F any\_philosopher\_eating) | **False** |
| LTLSPEC  !( F G all\_pickup ) | **True** |

Note: The result of the simulation can be verified in the attached file (result.txt).

Even thought the solution implemented avoids starvation, it does not avoid deadlock. Instead it resolves the deadlock by the concept of altruist philosopher.

But after a deadlock occurs the verification failed to verify that any philosopher is able to eat (as it is possible to see in the result table or in the result file).

I think the altruist solution can resolve the deadlock but in all simulations i could not force a situation were any philosopher could eat. Since all will release the fork in their hands. For example:

1. Philosopher 0 picks up left fork
2. Philosopher 1 picks up left fork
3. Philosopher 2 picks up left fork
4. Philosopher 3 picks up left fork
5. Philosopher 4 picks up left fork (DEADLOCK)
6. Philosopher 4 releases left fork
7. Philosopher 3 releases left fork
8. Philosopher 2 releases left fork
9. Philosopher 1 releases left fork
10. Philosopher 0 releases left fork

In this example, even though Philosopher 0 has now its right fork free he will release the left fork.

Overall, the solution implemented meets the requirements and is able to avoid starvation