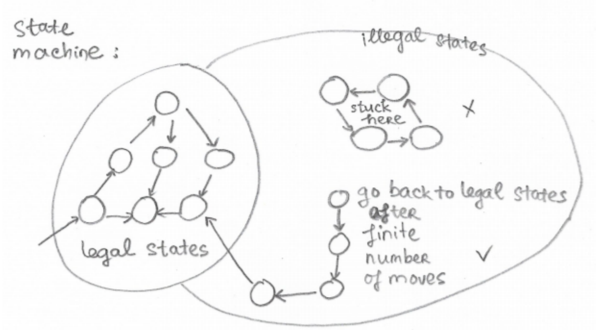
EE 382V: Parallel Algorithms **Fall 2017**

Lecture 18: November 4

Lecturer: Vijay Garg *Scribe: My Luc*

18.1 Introduction

A compute system has two parts: program and data. For data, when a bit gets corrupted, it will behave badly. The question is if it will continue to behave badly or come back to normal/legal state. A self-stabilizing system is guaranteed to come back to a normal/legal state after a finite number of moves.

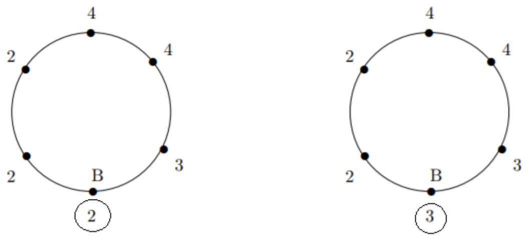


18.2 Mutual Exclusion with K-state Machines

A machine can enter critical section only if it has privilege. The goal of the self-stabilizing mutual exclusion algorithm is to determine who has the privilege and how the privileges move in the network. Self-stabilizing algorithm ensures that sooner or later the system will be in a configuration in which only 1 process has the privilege to enter the critical section.

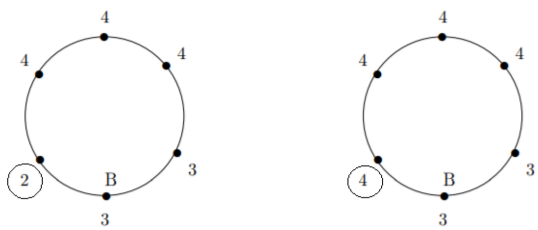
Assume:
N: number of machines 0…N-1. Each machine is a K-state machine.
S: state of the machine 0…K-1 (K $\geq$ N)
K: number of labels using in the system (K $\geq$ N)
L: left neighbor

Bottom machine: if (L = S) then S := S+1 mod K (The bottom machine has privilege if L = S. When it is done with the critical section, it updates S := S+1 mod K)



A move by bottom machine: since S = 2 and its left L = 2, it enters the CS.
It updates S = 3 when it is done.

Normal machine: if (L $\neq$ S) then S := L (Normal machine has privilege if L $\neq$ S. When it is done with the critical section, it updates S := L)

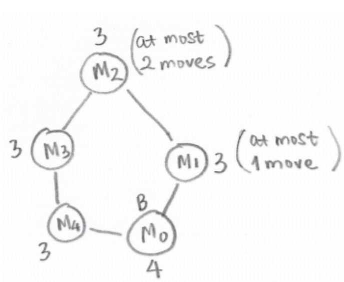


A move by normal machine: since S = 2 and its left L = 4, it enters the CS.
It updates S = 4 when it is done.

There are 2 legal states:
1. All states are the same: in this case the bottom machine has the privilege
2. Some consecutive states are the same to a certain point m and then different from that point on: in this case the normal machine m+1 has the privilege.

18.2.1 Claim 1

Any sequence of moves in which the bottom machine does not move is finite.



Normal machine i can move at most i times.
Total number of moves is finite: 1 + 2 + 3 + ... + N-1 = O(N^2)

18.2.2 Claim 2

In any configuration, either:
1. Bottom machine has unique label or
2. There is a label missing from the network.

There are N machines, so at least N labels are used. If a label is not unique, then some other machine must have the same label. Therefore, some labels must be missing.

18.2.3 Claim 3

Within finite number of move, "Bottom machine has unique label" must be true.

If (L = S) then S := S+1 mod K. The bottom machine is cycling labels from 0, 1, 2, ..., K-1, 0, 1, 2, ..., K-1, 0, 1, ... The bottom machine is the only machine that generates labels. Normal machines copy label from its left.
Within a finite number of moves, the bottom machine has to move, as soon as the bottom machine gets to the missing label, it has unique label.

18.2.4 Theorem

For all configurations, the system will get into a legal state in a finite number of moves.

18.3 Application

Routing table gets corrupted: a package is looping in a circle. Routing table needs to be designed in a way that eventually the package reaches its destination.