Goals of the lecture: Self-stabilization

- Fault-tolerance
- Definition of self-stabilizing
- Algorithm with $K$-state Machines
  - Proof
- Algorithm with 3-state Machines
  - Proof

References: Dijkstra 74, Dijkstra 86
Fault-tolerance

- systems which recover from faults
- self-stabilization: highly fault-tolerant
  - a fault can change any data
- system viewed as consisting of legal and illegal states
- self-stabilization: should reach a legal state in finite moves
Terminology

- Underlying topology: connection graph
- neighbors
- privilege: boolean function of
  - own state,
  - states of its neighbors
- legal state: application dependent
Requirements on legal state

- In each legal state, one or more privileges
- any move from a legal state leads to a legal state
- each privilege present in at least one legal state
- for any pair of legal states, there exist a sequence of transferring moves

Definition of self-stabilization: Regardless of initial state, and privilege selected each time, the system is guaranteed to reach a legal state after a finite number of moves.
Example: Mutual Exclusion

legal state: exactly one privilege

- $N+1$ machines numbered $0..N$
- $L,S,R$: states of left, self, right
- bottom machine: machine 0
- format:
  if privilege then corresponding move fi
**Algorithm I: K-state machine** \((K > N)\)

**Bottom:**

\[
\text{if } (L = S) \text{ then } S := S + 1 \text{ mod } K \text{ fi}
\]

**For other machines:**

\[
\text{if } (L \neq S) \text{ then } S := L \text{ fi}
\]
Example
Proof

Lemma 0: If the system is in a legal state, then it will stay legal.

Lemma 1: A sequence of moves in which Bottom does not move is finite.

Lemma 2: Given any configuration, either
(1) no other machine has the same state as the bottom, or
(2) there exists a value which is different from all machines.

Lemma 3: With in a finite number of moves, part one of Lemma 2 will be true.

Theorem 1: Within finite number of moves, the system will reach a legal state.
Algorithm II: 3-state machine

Ring of at least 3 machines
Bottom: B, Normal: N, Top: T
configuration viewed as a string of 0,1,2

Bottom:
if \((B + 1 = R)\) then \(B := B + 2\);

Normal:
if \((L = S + 1)\) or \((R = S + 1)\) then \(S := S + 1\);

Top:
if \((L = B)\) and \((T \neq B + 1)\) then \(T := B + 1\)
Viewing the string with arrows

\[ y = \# \text{ of left-pointing} + 2\# \text{ of right-pointing} \]

Bottom:
(0) \( B \leftarrow R \) to \( B \rightarrow R \) \( \Delta y = +1 \)

Normal Machine:
(1) \( L \rightarrow S \rightarrow R \) to \( L \leftarrow S \rightarrow R \) \( \Delta y = 0 \)
(2) \( L \leftarrow S \rightarrow R \) to \( L \leftarrow S \rightarrow R \) \( \Delta y = 0 \)
(3) \( L \rightarrow S \leftarrow R \) to \( L \leftarrow S \rightarrow R \) \( \Delta y = -3 \)
(4) \( L \rightarrow S \rightarrow R \) to \( L \leftarrow S \leftarrow R \) \( \Delta y = -3 \)
(5) \( L \leftarrow S \leftarrow R \) to \( L \rightarrow S \leftarrow R \) \( \Delta y = 0 \)

Top Machine (privilege also depends on B):
(6) \( L \rightarrow T \) to \( L \leftarrow T \) \( \Delta y = +1 \)
(7) \( L \rightarrow T \) to \( L \leftarrow T \) \( \Delta y = +1 \)
Example

1 → 0 → 2 ← 0

B
1 1 ← 2 ← 0

B
1 ← 2 2 ← 0

B
0 → 2 2 ← 0

B
0 0 → 2 ← 0

B
0 0 0 0 0

B
0 0 0 0 ← 1

B
T

T

T

T

T

T
Proof

Claim: Single arrow implies it stays that way.
Claim: string free from arrow creates one in a single move.

Now show that if multiple arrow then y will be decreased in finite moves.
Proof contd

Lemma 0: Between two successive moves of Top at least one move of Bottom takes place.

Lemma 1: A sequence of moves in which Bottom does not move is finite. Proof: sufficient to consider normal machines. (3),(4),(5) decrease number of arrows. (1) and (2) moves finite due to topology.

Theorem: Within finite moves, there is one arrow in the string. Proof: between successive moves of bottom, falsification of “left-most arrow exists and points to the right” happen in (3), (4), or (6). If (6) then done. If (3) or (4), y decreases by 3. y can increase by at most 2 per move of Bottom, thus y is decreased by 1.