Self-stabilizing Systems in Spite of Distributed Control

Edsger W. Dijkstra
Burroughs Corporation
Overview

• Motivation

• Complications

• Concepts

• Solutions
Motivation

• Parallel, distributed, multi-threaded computing etc. needs synchronization

• How to guarantee the correctness?

• How to recover from the incorrect system?
Motivation

• Correctness

• The system is in a legitimate state

• Recovery

• The system will reach a legitimate state
Classification

• System state is saved in common place

• No common store for synchronization
Concepts

• Neighbors
• Distributed Control
• Privileges
• Self-stabilizing
Neighbors

• Processes with direct communication capability

• Worse: a small subset of processes
Distributed Control

• No common store for mutually exclusive access

• The system state is saved in variables across various processes
Privileges

• For machines

• Boolean function of states (L,S,R)

• True -> Present -> Move -> New state

• Nondeterministic
Legitimate System from Privilege

• One or more privileges are present
• Legitimate state moves to another legitimate state
• Each privilege must present in at least one legitimate state
• There is a path between any legitimate states
Self-stabilizing

• Regardless of initial state and privilege selected

• At least one privilege present

• The system moves into a legitimate state after a finite number of steps
Solutions

• Assumptions: N+1 machines in a ring
• Nr. 0 is the bottom machine
• Nr. N is the top machine
• Solutions:
  K-state machines
  Four-state machines
  Three-state machines
K-state Machines

• For the bottom machine:
  If L=S then S:=(S+1) mod K fi

• For others
  If L!=S then S:=L fi

Where L, R are states of left, right neighbor
And 0 <= S < K
Four-state machines

• Each state has $x_S$ and $u_p S$

• For the bottom machine:
  $u_p S = \text{true}$

• For the top machine:
  $u_p S = \text{false}$
Four-state machines

• The bottom machine
  if \( x_S = x_R \) and non upR then \( x_S := \text{non} \ x_S \) fi

• The top machine
  If \( x_S \neq x_L \) then \( x_S := \text{non} \ x_S \) fi

• Others
  If \( x_S \neq x_L \) then \( x_S := \text{non} \ x_S \); upS:=true fi;
  If \( x_S = x_R \) and upS and non upR then upS:=false fi
Three-state Machines

• The bottom machine
  If \((S+1) \mod 3 = R\) then \(S:= (S-1) \mod 3\) fi

• The top machine
  If \(L=R\) and \((L+1) \mod 3 \neq S\) then \(S:= (L+1) \mod 3\) fi

• Others
  If \((S+1) \mod 3 = L\) then \(S:= L\) fi;
  If \((S+1) \mod 3 = R\) then \(S:= R\) fi
Question 1

- Are all systems “self-stabilizing”? Are all fault-tolerant systems “self-stabilizing”?
Question 2

• The daemon selects privilege(s) for next move, so in real implementation, is the selection arbitrary? Or otherwise?