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 xS and upS

- For the bottom machine:

upS=true

- For the top machine:

upS=false

Four-state machines

- The bottom machine

if xS=xR and non upR then xS:=non xS fi

- The top machine

If xS!=xL then xS:=non xS fi

- Others

If xS!=xL then xS:=non xS; upS:=true fi;

If xS=xR 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!=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?