Chapter 9

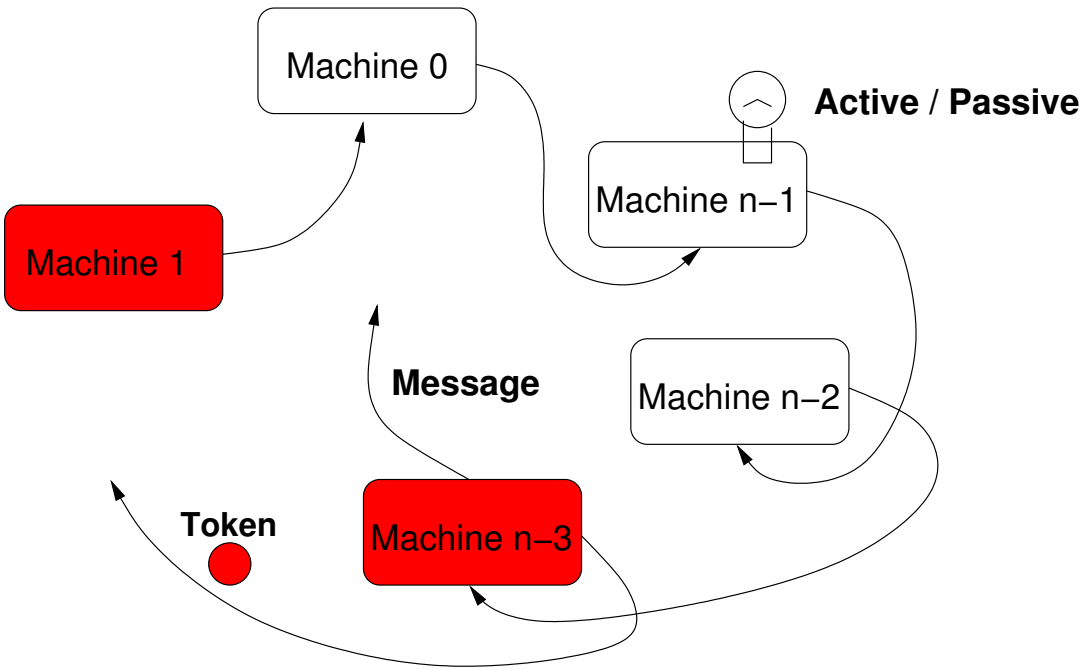
Application: Verification of distributed systems

Distributed Termination Detection : Dijkstra

Example 9.1. Implement the following termination detection protocol:

A passive machine becomes active, iff it receives a message from another machine.

Only active machines can send messages.



Edsger W. Dijkstra, W. H. J. Feijen, and A.J.M. van Gasteren. Derivation of a Termination Detection Algorithm for Distributed Computations. IPL 16 (1983).

Assumptions for distributed termination detection

Rules for a probe

Rule 0 When active, $Machine_{i+1}$ keeps the token; when passive, it hands over the token to $Machine_i$.

Rule 1 A machine sending a message makes itself red.

Rule 2 When $Machine_{i+1}$ propagates the probe, it hands over a red token to $Machine_i$ when it is red itself, whereas while being white it leaves the color of the token unchanged.

Rule 3 After the completion of an unsuccessful probe, $Machine_0$ initiates a next probe.

Rule 4 $Machine_0$ initiates a probe by making itself white and sending to $Machine_{n-1}$ a white token.

Rule 5 Upon transmission of the token to $Machine_i$, $Machine_{i+1}$ becomes white. (Notice that the original color of $Machine_{i+1}$ may have affected the color of the token).

Correctness of the abstract version: Dijkstra

Assumptions

The machines constitute a closed system, i.e. messages can only be dispatched among each other (no outside messages). The system in the initial state can have any color and several machines can be active. The token is located in the 0'th. machine.
The given rules describe the transfer of the token and the coloration of the machines upon certain activities.
The task is to determine a state in which all the machines are passive (not active). This is a stable state of the system, because only active machines can dispatch messages and passive machines can only become active by receiving a message.
The invariant: Let t be the position on which the token is, then following invariant holds:
$(\forall i : t < i < n \;\; Machine_i$ is passive$) \vee (\exists j : 0 \leq j \leq t \;\; Machine_j$ is red$) \vee$ ($Token$ is red)

Distributed Termination Detection: Correctness

$(\forall i : t < i < n \;\; Machine_i \text{ is passive}) \vee (\exists j : 0 \leq j \leq t \;\; Machine_j \text{ is red}) \vee$
$(Token \text{ is red})$

Correctness argument
When the token reaches $Machine_o$, $t = 0$ and the invariant holds.
If
$(Machine_o \text{ is passive}) \wedge (Machine_o \text{ is white}) \wedge (Token \text{ is white})$
then
$(\forall i : 0 < i < n \;\; Machine_i \text{ is passive})$ must hold, i.e. termination.

Proof of the invariant Induction over t:
The case t = n - 1 is easy.
Assume the invariant is valid for $0 < t < n$, prove it is valid for $t - 1$.

Distributed Abstract State Machines: Model

Signature:

static

$COLOR = \{red, white\}$ $TOKEN = \{redToken, whiteToken\}$

$MACHINE = \{0, 1, 2, \ldots, n-1\}$

$next : MACHINE \rightarrow MACHINE$

e.g. with $next(0) = n-1, next(n-1) = n-2, \ldots, next(1) = 0$

controlled

$color : MACHINE \rightarrow COLOR$ $token : MACHINE \rightarrow TOKEN$

$RedTokenEvent, WhiteTokenEvent : MACHINE \rightarrow BOOL$

monitored $Active : MACHINE \rightarrow BOOL$

$SendMessageEvent : MACHINE \rightarrow BOOL$

Distributed Termination Detection: DASM-Procedure

Macros: (Rule definitions)

- $ReactOnEvents(m : MACHINE) =$
  $if\ \ RedTokenEvent(m)\ \ then$
  $token(m) := redToken$
  $RedTokenEvent(m) :=\ \ undef$
  $if\ \ WhiteTokenEvent(m)\ \ then$
  $token(m) := whiteToken$
  $WhiteTokenEvent(m) :=\ \ undef$
  $if\ \ SendMessageEvent(m)\ \ then\ \ color(m) := red$ Rule 1

- $Forward(m : MACHINE, t : TOKEN) =$
  $if\ \ t = whiteToken\ \ then$
  $WhiteTokenEvent(next(m)) :=\ \ true$
  $else$
  $RedTokenEvent(next(m)) := true$

Distributed Termination Detection: DASM-Procedure

Programs

- *RegularMachineProgram* =

  ReactOnEvents(*me*)
  if ¬ *Active*(*me*) ∧ *token*(*me*) ≠ *undef* *then* Rule 0
    InitializeMachine(*me*) Rule 5
    if *color*(*me*) = *red* *then*
      Forward(*me*, *redToken*) Rule 2
    else
      Forward(*me*, *token*(*me*)) Rule 2

- With *InitializeMachine*(*m* : *MACHINE*) =

  token(*m*) := *undef*
  color(*m*) := *white*

Distributed Termination Detection: Procedure

Programs

- *SupervisorMachineProgram* =

$$
\begin{array}{l}
\quad \mathit{ReactOnEvents}(\mathit{me}) \\
\quad \mathit{if} \neg\ \mathit{Active}(\mathit{me}) \wedge\ \mathit{token}(\mathit{me}) \neq\ \mathit{undef}\ \ \mathit{then} \\
\qquad \mathit{if}\ \ \mathit{color}(\mathit{me}) =\ \mathit{white} \wedge\ \mathit{token}(\mathit{me}) =\ \mathit{whiteToken}\ \ \mathit{then} \\
\qquad\quad \mathit{ReportGlobalTermination} \\
\qquad \mathit{else} \quad \text{Rule 3} \\
\qquad\quad \mathit{InitializeMachine}(\mathit{me}) \quad \text{Rule 4} \\
\qquad\quad \mathit{Forward}(\mathit{me}, \mathit{whiteToken}) \quad \text{Rule 4}
\end{array}
$$

Distributed Termination Detection

Initial states

$$\exists m_0 \in MACHINE$$
$$(program(m_0) = SupervisorMachineProgram \wedge$$
$$token(m_0) = redToken \wedge$$
$$(\forall m \in MACHINE)(m \neq m_0 \Rightarrow$$

$$(program(m) = RegularMachineProgram \wedge token(m) = undef)))$$

Environment constraints For all the executions and all linearizations holds:

$$\mathbf{G}\ (\forall m \in MACHINE)$$
$$(SendMessageEvent(m) = true \Rightarrow (\mathbf{P}(Active(m)) \wedge Active(m)))$$
$$\wedge\ ((Active(m) = true \wedge \mathbf{P}(\neg Active(m)) \Rightarrow$$
$$(\exists m' \in MACHINE)\ (m' \neq m \wedge\ SendMessageEvent(m'))))$$

Nextconstraints