Geldautomat

Übung 4.4 *Abstrakte Modellierung eines Geldautomaten:*
Drei Agenten sind im Modell: GA-Manager, Autentitifikations-Manager, Konto-Manager. Um eine Summe vom Konto abzuheben sind folgende logische Operationen durchzuführen:

1. *Eingabe der Karte (Nummer) und der PIN.*
2. *Überprüfe Gültigkeit der Karte und PIN (AU-manager).*
3. *Eingabe der Summe.*
4. *Überprüfe ob Summe vom Konto abgehoben werden kann (KO-Manager).*
5. *Falls OK aktualisiere Konto Stand und gebe Summe aus.*
6. *Falls nicht OK gebe entsprechende Nachricht aus.*

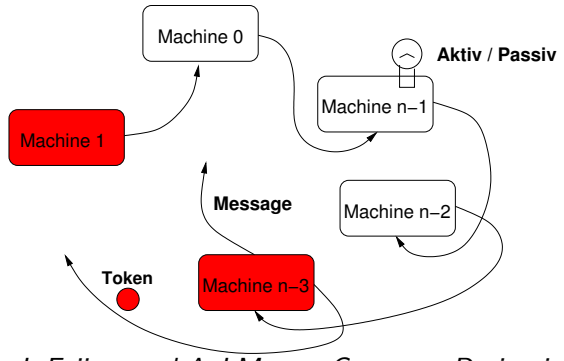
Realisiere ein asynchrones Kommunikationsmodell wobei Timeouts Transaktionen abbrechen können.

Distributed Termination Detection

Beispiel 4.3 *Modelliere folgendes Terminierungsfestellungsprotokoll:*

Genau dann wird eine passive Maschine aktiv, wenn sie eine Nachricht einer anderen Maschine erhält.

Nur aktive Maschinen können Nachrichten senden.



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).

Annahmen für Distributed Termination Detection

Rule 0 Falls $Machine_{i+1}$ aktiv ist, so behält sie das Token; ist sie inaktiv so leitet sie das Token an $Machine_i$ weiter.

Rule 1 Eine Maschine die eine Message sendet färbt sich rot.

Rule 2 Propagiert $Machine_{i+1}$ die Probe, so leitet sie ein rotes Token weiter falls ihre Farbe rot ist, ansonsten leitet sie das Token unverändert weiter an $Machine_i$.

Rule 3 Nach Beendigung einer erfolglosen Probe leitet $Machine_0$ eine neue Probe ein.

Rule 4 $Machine_0$ initiiert eine neue Probe indem sie sich weiß färbt und ein weißes Token an $Machine_{n-1}$ leitet.

Rule 5 Nach Weiterleiten des Tokens an $Machine_i$ wird $Machine_{i+1}$ weiß gefärbt. (Beachte die ursprüngliche Farbe von $Machine_{i+1}$ kann die Farbe des Tokens beeinflusst haben).

Distributed Termination Detection: Verfahren

Signatur:

static
$COLOR = \{red, white\}$ $TOKEN = \{redToken, whiteToken\}$
$MACHINE = \{0, 1, 2, \ldots, n-1\}$
$next : MACHINE \to MACHINE$
Z.B. mit $next(0) = n-1, next(n-1) = n-2, \ldots, next(1) = 0$

controlled
$color : MACHINE \to COLOR$ $token : MACHINE \to TOKEN$
$RedTokenEvent, WhiteTokenEvent : MACHINE \to BOOL$

monitored $Active : MACHINE \to BOOL$
$SendMessageEvent : MACHINE \to BOOL$

Distributed Termination Detection: Verfahren

Makros: (Rule Definitions)

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

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

Distributed Termination Detection: Verfahren

Programme

- $RegularMachineProgram =$

  $ReactOnEvents(me)$
  $if \neg\ Active(me) \wedge\ \ token(me) \neq\ \ undef\ \ then$ Rule 0
  $\quad InitializeMachine(me)$ Rule 5
  $\quad if\ \ color(me) =\ \ red\ \ then$
  $\quad\quad Forward(me, redToken)$ Rule 2
  $\quad else$
  $\quad\quad Forward(me, token(me))$ Rule 2

- Mit $InitializeMachine(m : MACHINE) =$

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

Distributed Termination Detection: Verfahren

Programme

- $SupervisorMachineProgram =$

  $ReactOnEvents(me)$
  $if \neg\ Active(me) \wedge\ \ token(me) \neq\ \ undef\ \ then$
  $\quad if\ \ color(me) =\ \ white \wedge\ \ token(me) =\ \ whiteToken\ \ then$
  $\quad\quad ReportGlobalTermination$
  $\quad else$ Rule 3
  $\quad\quad InitializeMachine(me)$ Rule 4
  $\quad\quad Forward(me, whiteToken)$ Rule 4

Distributed Termination Detection

Initiale Zustände

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

Umgebungsconstraints Für alle Läufe und alle Linearisierungen gilt:

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

Nextconstraints

Distributed Termination Detection

Korrektheit nach Dijkstra

Annahmen: Die Maschinen bilden ein geschlossenes System, d.h. Nachrichten können nur untereinander Versendet werden. Das System im Initialzustand kann beliebig gefärbt sein und mehrere Maschinen können aktiv sein. Das Token befindet sich bei der 0'ten Maschine. Die angebebenen Regel beschreiben die Übergabe des Tokens und die Färbung der Maschinen bei bestimmten Aktivitäten. Festzustellen ist ein Zustand bei dem alle Maschinen passiv (nicht aktiv) sind. Dies ist ein stabiler Zustand des Systems, da nur aktive Maschinen Nachrichten versenden können und passive Maschinen nur aktiviert werden können durch Erhalt einer Nachricht.

Die Invariante: Sei t die Stelle an der sich das Token befindet, dann gilt
$(\forall i : t < i < n \;\; Machine_i \text{ ist passiv}) \vee (\exists j : 0 \leq j \leq t \;\; Machine_j \text{ ist rot}) \vee$
$(Token \text{ ist rot})$

Distributed Termination Detection

$(\forall i : t < i < n \;\; Machine_i \text{ ist passiv}) \vee (\exists j : 0 \leq j \leq t \;\; Machine_j \text{ ist rot}) \vee$
$(Token \text{ ist rot})$

Korrektheitsargument

Wenn das Token zu $Machine_o$ gelangt ist $t = 0$ und die Invariante gilt.
Falls
$(Machine_o \text{ ist passiv}) \wedge (Machine_o \text{ ist weiß}) \wedge (Token \text{ ist weiß})$
so muß
$(\forall i : 0 < i < n \;\; Machine_i \text{ ist passiv})$ gelten, d.h. Terminierung.

Nachweis der Invariante Induktion nach t:
Der Fall t = n - 1 ist einfach.
Invariante gelte für $0 < t < n$, zeige sie gilt für $t - 1$.