Channel with Termination Detection Service

Shankar

April 26, 2014

Overview

- Distributed termination detection problem
  - given computation X where each system is active or inactive
    - active: send/rcv msgs, become inactive
    - inactive: become active upon rcving a msg
  - obtain computation Y that does not disturb X and informs a "sink" sytem when/if X has terminated
    - termination: all systems inactive, no msgs in transit
- Formalize as fifo channel with termination detection service
  - subset of addresses active initially
  - at each addr: send msgs; rcv msgs; signal inactive
  - at "sink" addr a0: get termination status

Termination detection channel: overview

- Vars:
  - active flag for every addr j
  - txh and rxh for every j,k
- input j.tx(k, msg)
  - allowed only if j active; send msg, update txh
- input j.rx()
  - rcv msg from any k; set j active
- input j.inactive()
  - allowed only if j active; set j inactive
- input a0.isTerminated()
  - return only if j inactive and txh equals rxh for every j,k

Termination detection channel – 1

- Service TdChannel(ADDR, a0, initActive)
  - initActive: initially active addresses
- Main
  - $txh_{j,k} \leftarrow []$ // tx history at j to k
  - $rxh_{j,k} \leftarrow []$ // rx history at k from j
  - $active_j \leftarrow$ j in initActive // active status at j
  - $v_j \leftarrow$ sid() // access system at j
  - return $\{v_j\}$ // sid map over ADDR
- Helper function terminated():
  - forall(j: not $active_j$) and forall(j,k: $txh_{j,k} = rxh_{j,k}$)

Termination detection channel – 2

- input v_j.tx(k,msg)
  - ic {$active_j$ and $k \neq j$ and no ongoing v_j.tx(.)}
    append msg to $txh_{j,k}$;
  - oc {true}
    return

- input Seq v_j.rx()
  - ic {no ongoing v_j.rx()}
  - output msg, k
    oc {$rxh_{k,j} \circ [msg]$ prefix-of $txh_{k,j}$}
    append msg to $rxh_{k,j}$
    $active_j \leftarrow$ true
    return msg

Termination detection channel – 3

- input v_j.inactive()
  - ic {$active_j$ and no ongoing v_j.inactive()}
    $active_j \leftarrow$ false
  - oc {true}
    return

- input v_{a0}.isTerminated()
  - ic {no ongoing v_{a0}.isTerminated()}
  - oc {terminated()}
    return

Termination detection channel – 4

- atomicity assumption: input parts and output parts

- progress assumption
  - ongoing v_j.tx *leads-to* not ongoing v_j.tx
  - $txh_{k,j}$.size $\geq$ i *leads-to*
    $rxh_{k,j}$.size $\geq$ i or not ongoing v_j.rx
  - ongoing v_j.inactive *leads-to* not ongoing v_j.inactive()
  - (terminated() and ongoing v_{a0}.isTerminated)
    leads-to not ongoing v_{a0}.isTerminated