Asynchronous ASMs

Phase Synchronization on Undirected Trees

Egon Börger

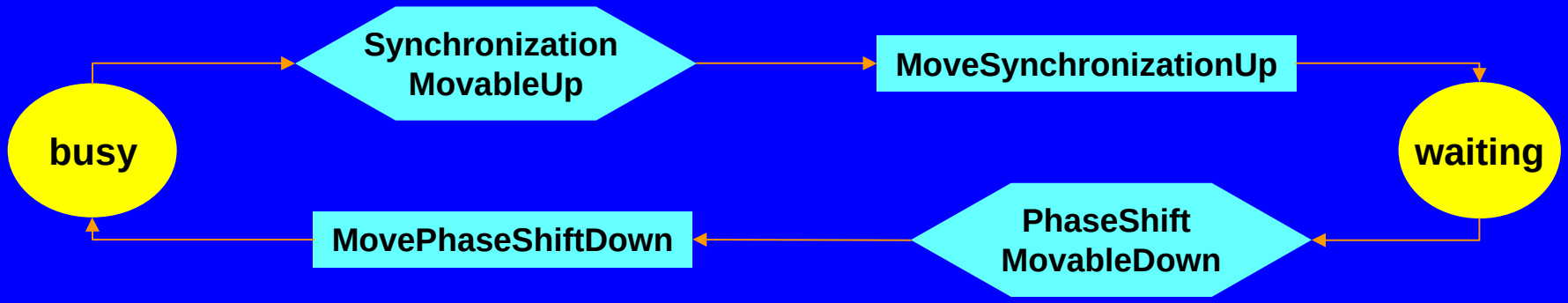
Dipartimento di Informatica, Universita di Pisa
http://www.di.unipi.it/~boerger

Phase Synchronization : problem statement

- Goal: Design a distributed algorithm that guarantees synchronized execution of phases of computations, occurring at nodes of an undirected tree (connected acyclic network), using only communication between tree neighbors
- Algorithmic Idea: every agent (tree node) before becoming busy in a new phase has to
  - synchronize with its neighbor nodes, moving the synchronization up along the tree structure
  - wait until all agents are waiting for the phase increase to start
  - increase its phase - when the phase shift becomes movable down - and move the phase shift further down along the tree structure
    - until all nodes have become busy in the new phase

Phase Synchronization ASM: Agent Signature

- Agent : an undirected tree of agents equipped with:
  - neighb $\subseteq$ Agent (external function)
  - phase: the current phase
  - synchPartner, waitPartner: Agent
    yields the neighbor node an agent is synchronized with (in moving the synchronization up or down the tree) resp. waiting for (to reverse the previous upward synchronization downwards)
  - ctl_state : {busy, waiting }
- Initially
  - ctl_state = busy, phase = 0, synchPartner = waitPartner = undef



SynchronizationMovableUp ° ∃ y∈ neighb SynchronizationMovableUpTo(y)

for some neighb y all other neighbors synchronize with self in phase(self)

MoveSynchronizationUp ° **choose** y∈ neighb with SynchronizationMovableUpTo(y)

synchronize self with y in phase(self) MoveSynchronizationUpTo(y)

SynchronizationMovableUpTo(y) ° ∀z∈ neighb-{y}

z.synchPartner(phase(self)) = self

PhaseShiftMovableDown

waitPartner synchronizes with self in phase(self)

MoveSynchronizationUpTo(y) ° ∀z∈ neighb-{y}

z.synchPartner(phase(self)) := undef

self.synchPartner(phase(self)) := y, self.waitPartner(phase(self)) := y

MovePhaseShiftDown ° phase : = phase +1

∀z∈ neighb-{waitPartner} self.synchPartner(phase(self)) : = z

waitPartner.synchPartner (phase(self)):= undef

Phase Synchronization ASM : Correctness & Liveness

- Proposition: In every infinite distributed run of agents equipped with the ASM for phase synchronization:
  - in any state any two busy agents are in the same phase (correctness property)
  - if every enabled agent will eventually make a move, then each agent will eventually reach each phase (liveness property)

Phase Synch: **PhaseShiftMovableDown Lemma**

- Lemma: For every phase p, whenever in a run a state is reached where for the first time

  u.waitPartner(p)=v synchronizes with u in phase p,

  every element of

  subtree(u,v) ∪subtree(v,u)∪{u,v}

  is waiting in phase p

  where subtree(x,y) = {n| n reachable by a path from x without touching y}

Proof of **PhaseShiftMovableDown Lemma**

- Proof of the lemma follows from the following two claims.
- Claim 1. When Synchronization is moved up in phase(u)=p from busy u to to v=u.synchPartner(p), the elements of subtree(u,v) are waiting, u becomes waiting, and they all remain so until the next application of a Shift rule "MovePhaseShiftDown".
  - Claim 1 follows by induction on the applications of the Synchronization rule.
    - n=1: true at the leaves
    - n+1: follows by induction hypothesis and Synchronization rule from

      subtree(u,v)= $\dot{E}_{i<n}$ subtree(u_i,u) $\cup$ {u_0,...,u_{n-1}} for neighb(u}= {u_0,...,u_n} with v= u_n (by connectedness)

Proof of **PhaseShiftMovableDown Lemma**

- Claim 2. For every infinite run and every phase p, a state is reached in which all agents are waiting in phase p and for some agent u its waitPartner in phase p synchronizes with u in phase p.
  - Follows by induction on p.

Reference

- W.Reisig: Elements of Distributed Algorithms
  Springer-Verlag 1998
  - See Section 36 (in particular Fig. 36.1) and a correctness and liveness proof in Section 81
- E. Börger, R. Stärk: Abstract State Machines. A Method for High-Level System Design and Analysis Springer-Verlag 2003, see http://www.di.unipi.it/AsmBook
  - See Chapter 6.1