Asynchronous ASMs

Network Leader Election

Egon Börger

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

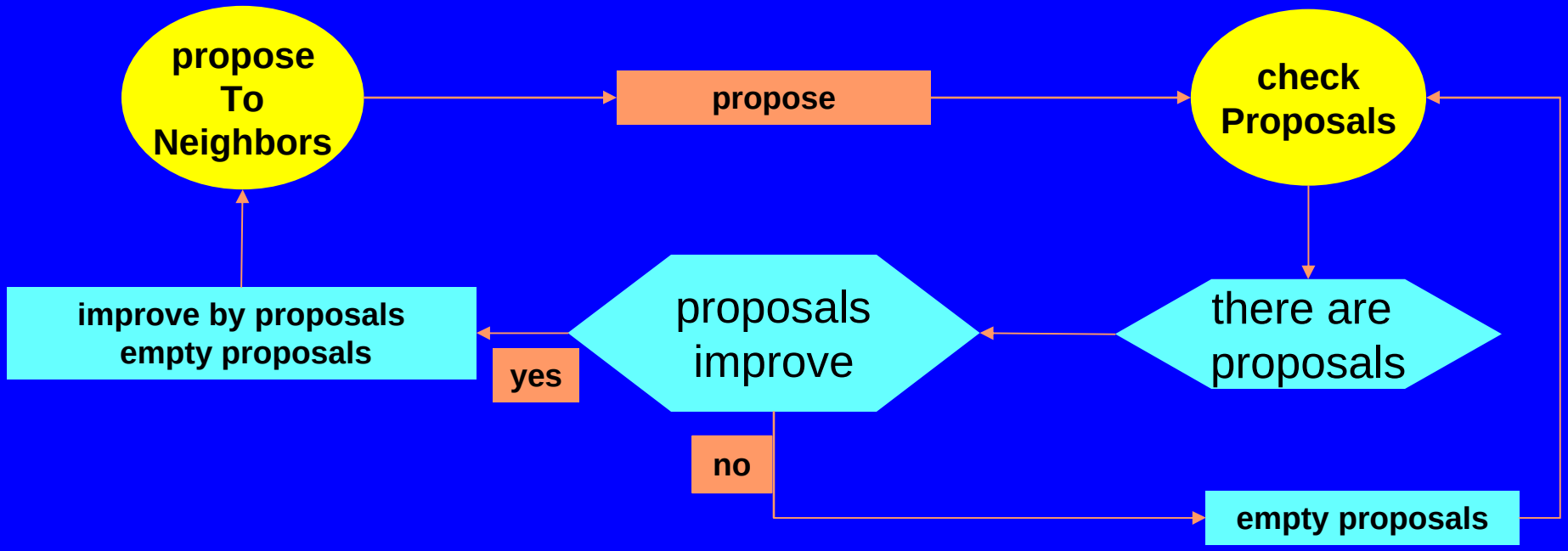
Leader Election: problem statement

- Goal: Design a distributed algorithm for the election of a leader in finite connected networks of homogeneous agents, using only communication (message passing) between neighbor nodes.
- Assumptions:
  - network nodes (agents) are connected & linearly ordered
  - leader = max (Agent) wrt the linear order <
- Algorithmic Idea: every agent (network node)
  - proposes to his neighbors his current leader candidate
  - checks the leader proposals received from his neighbors
    - upon detecting a proposal which improves his leader candidate he improves his candidate for his next proposal
- Eventually cand=max(Agent) holds for all agents

Leader Election: Agent Signature

- Agent : finite linearly ordered connected set
  - < linear order of Agent (external function)
    - leader = max (Agent) wrt the linear order <
- Each agent equipped with:
  - neighb ⊆ Agent (external function)
  - cand: Agent (controlled function)
  - proposals ⊆ Agent (controlled function)
  - ctl_state : {proposeToNeighbors, checkProposals}
- Initially ctl_state=proposeToNeighbors, cand=self, proposals = empty

Leader Election Control State ASM



propose ° forall n ∈ neighb insert cand to proposals(n)

proposals improve ° max (proposals) > cand

improve by proposals ° cand := max (proposals)

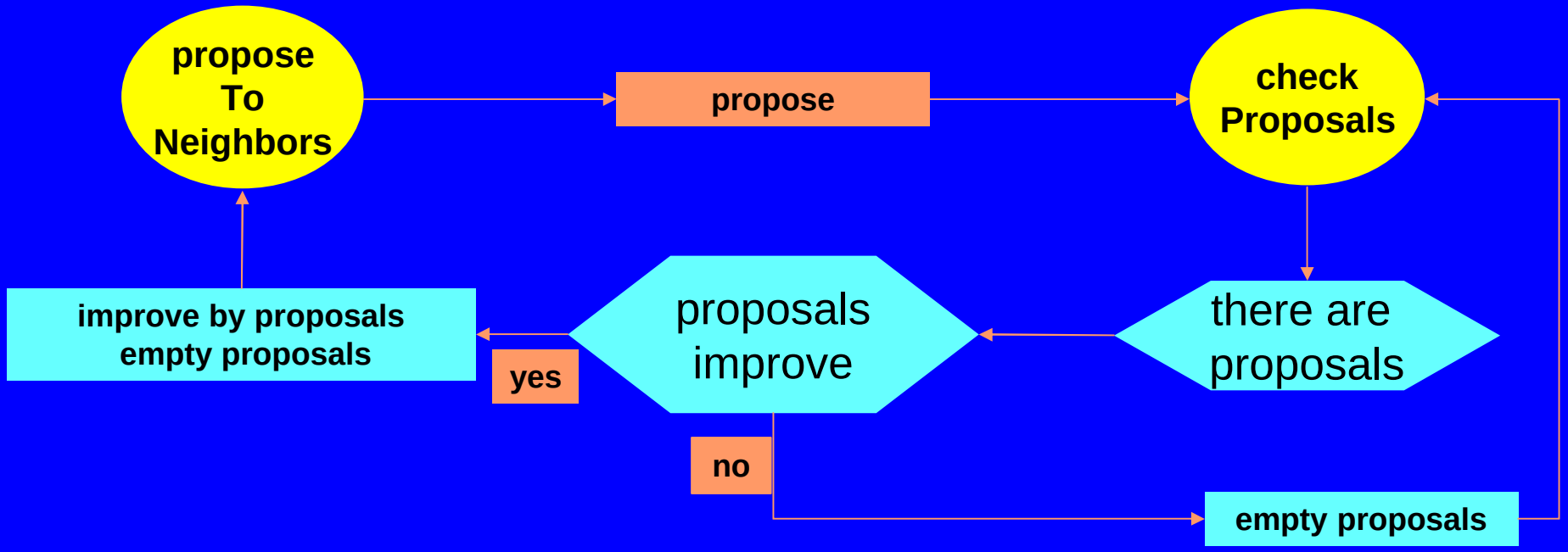
Leader Election: Correctness property

- Proposition: In every distributed run of agents equipped with the leader election ASM, eventually for every agent holds:
  - cand=max(Agent)
  - ctl_state = checkProposals
  - proposals = empty
- Proof (assuming that every enabled agent will eventually make a move): induction on runs and on $\Sigma\{\text{leader} - \text{cand}(n) \mid n \in \text{Agent}\}$
  - measuring “distances” of candidates from leader

Refining Leader Election: compute minimal path to leader

- Goal: refine the leader election algorithm to compute for each agent also a shortest path to the leader, providing
  - a neighbor (except for leader) which is closest to the leader
  - the minimal distance to the leader
- Idea: enrich cand and proposals by neighbor with minimal distance to leader:
  - nearNeighb: Agent
  - distance: Distance (e.g. = Nat $\cup\{\infty\}$)
  - proposals: PowerSet(Agent x Agent x Distance)
- Initially nearNeighbor = self distance =0

ASM Computing Minimal Path To Leader



propose ° forall n ∈ neighb insert (cand, nearNeighb, distance) to proposals(n)

proposals improve ° **let** m = Max (proposals) **in** m > cand
or (m=cand **and** minDistance(proposalsFor m)+1 < distance)

Max taken over agents

update PathInfo to m °
choose (n,d) with (m,n,d) ∈ proposals
d=minDistance(proposalsFor m) **in**
nearNeighb := n
distance := d+1

improve by proposals °
cand := Max (proposals)
update PathInfo to Max (proposals)

Minimal Path Computation: Correctness

- Proposition: In every distributed run of agents equipped with the ASM computing a minimal path to the leader, eventually for every agent holds:
  - cand=max(Agent)=leader
  - distance=minimal distance of a path from agent to leader
  - nearNeighbor = a neighbor of agent on a minimal path to the leader (except for leader where nearNeighbor=leader)
  - ctl_state = checkProposals
  - proposals = empty
- Proof (assuming that every enabled agent will eventually make a move): induction on runs and on $\Sigma\{leader - cand(n) | n \in Agent\}$ with side induction on the minimal distances in proposalsFor Max(proposals)

Exercises

- Refine the CHECK submachine of the leader election ASM by a machine which checks proposals elementwise. Prove that the refinement is correct.
  - Hint: See Reisig op.cit. Fig.32.1
- Adapt the ASM for the election of a maximal leader and for computing a minimal path wrt a partial order $\leq$ instead of a total order.
- Reuse the leader election ASM to define an algorithm which, given the leader, computes for each agent the distance (length of a shortest path) to the leader and a neighbor where to start a shortest path to the leader.
  - Hint: See Reisig op.cit. Fig.32.2

References

- W.Reisig: Elements of Distributed Algorithms Springer-Verlag 1998
  - See Section 32 (in particular Fig. 32.1 and 32.2) and Section 76 for a correctness proof.
- 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