Disco: Bridging Synchrony and Asynchrony

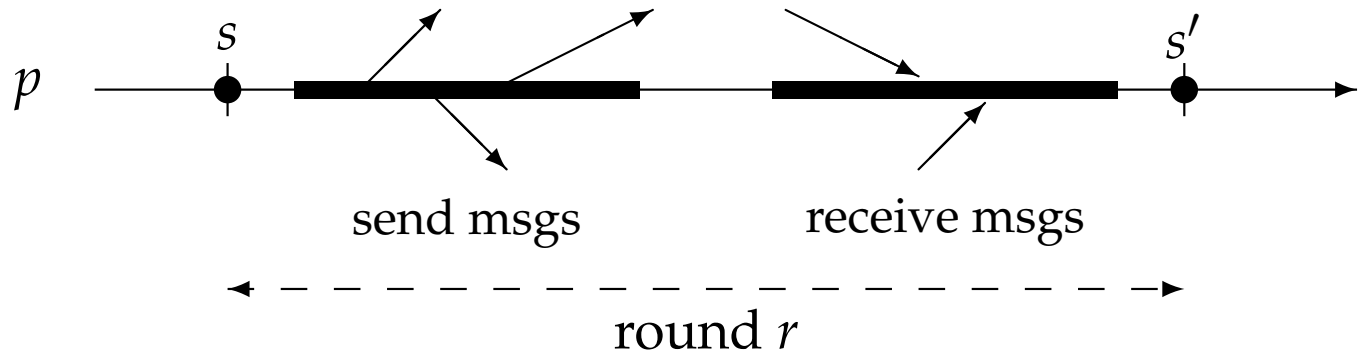
Stephan Merz

(project involving Antique and VeriDis)

Journée scientifique Nomadic Labs - Inria
Paris, 21 september 2020

Communication-closed Rounds

- Round-based distributed algorithms



  - process-local organization of computation
  - state s' computed from s and messages received
  - messages sent in round r are received only in round r

- Purely logical structure of algorithms
  - processes execute fully asynchronously
  - most classical consensus algorithms fall into this class

[Elrad & Francez 1982], [Gafni 1998], [Charron-Bost & Schiper 2006]

Reduction to Synchronous Execution Model

- Processes execute each round[1] atomically

- Rearrange rounds of different processes
  - round rnd_p^m right-commutes with rnd_q^n if $m > n$
  - messages sent during rnd_q^n do not influence rnd_p^m

[1] shifted by "half a round"

Reduction to Synchronous Execution Model

- Processes execute each round[1] atomically

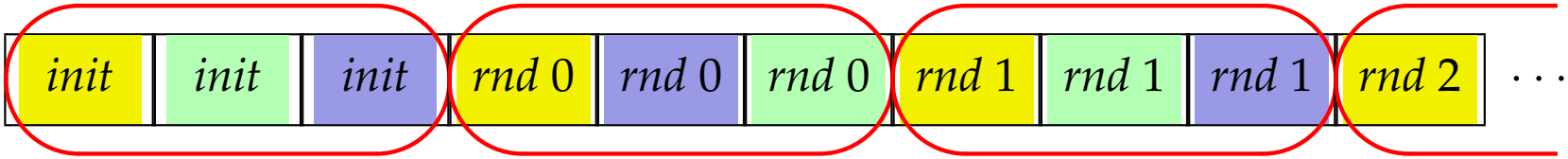
- Rearrange rounds of different processes
  - round rnd_p^m right-commutes with rnd_q^n if $m > n$
  - messages sent during rnd_q^n do not influence rnd_p^m

[1] shifted by "half a round"

Reduction to Synchronous Execution Model

- Processes execute each round[1] atomically

- Rearrange rounds of different processes
  - round rnd_p^m right-commutes with rnd_q^n if $m > n$
  - messages sent during rnd_q^n do not influence rnd_p^m



- Executions of same round by different processes are independent
  - pretend that rounds are the unit of atomicity

[1] shifted by "half a round"

Correspondence of Executions

Theorem

For every asynchronous execution of a round-based distributed algorithm there exists a synchronous execution such that every process sees the same sequence of local states.

Correspondence of Executions

Theorem

For every asynchronous execution of a round-based distributed algorithm there exists a synchronous execution such that every process sees the same sequence of local states.

Corollary

Any stuttering-invariant property that depends only on local views can be verified over synchronous executions.

- "Local" LTL-X formulas
  - any formula containing state variables of a single process
  - first-order combinations of "local" formulas
  - examples: distributed consensus, k-set agreement, ...

Existing Work (Antique / Veridis)

- Mechanized verification
  - representation of Heard-Of model in Isabelle/HOL
  - SMT-based verification of consensus algorithms
  - classical algorithms tolerating crash and Byzantine failures
- PSync: domain-specific language
  - express algorithms and verify them in the synchronous model
  - generate executables for asynchronous execution
  - strong guarantees and reasonable performance

Objectives of Disco

1. Design synchronous computational model for blockchains
   - support full Byzantine failures, including signed messages
   - take into account probabilistic guarantees
   - validation: represent Tezos and similar protocols

Objectives of Disco

1. Design synchronous computational model for blockchains
   - support full Byzantine failures, including signed messages
   - take into account probabilistic guarantees
   - validation: represent Tezos and similar protocols
2. Mechanized verification techniques
   - generate verification conditions for proving safety properties
   - design SMT theories and/or support interactive proof
   - reduce liveness proofs to checking safety properties

Objectives of Disco

1. Design synchronous computational model for blockchains
   - support full Byzantine failures, including signed messages
   - take into account probabilistic guarantees
   - validation: represent Tezos and similar protocols
2. Mechanized verification techniques
   - generate verification conditions for proving safety properties
   - design SMT theories and/or support interactive proof
   - reduce liveness proofs to checking safety properties
3. Design an execution platform
   - efficient execution of programs in an asynchronous context
   - account for timeouts, digital signatures, optimizations
   - formally prove correctness in a proof assistant