Parametrized Verification of Fault-Tolerant Distributed Algorithms

(work in progress)

Ilina Stoilkovska

stoilkov@forsyte.at

logics Logical Methods in Computer Science

12.02.2016

Motivation

- wide-spread use of distributed algorithms
- literature features manual proofs of correctness of distributed algorithms
- goal: extended integration of verification techniques with distributed algorithms
- increase the trust in distributed algorithms by
  - formalization
  - automated verification of safety and liveness

Parametrized Verification of Fault-Tolerant Distributed Algorithms

Distributed Algorithms

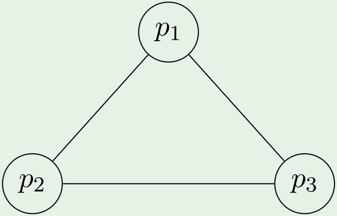
- designed to run on hardware consisting of interconnected processors
- many applications
- classical problems: leader election, consensus, mutual exclusion...
- different system settings
  1. timing model
  2. interprocess communication

Parametrized Verification of Fault-Tolerant Distributed Algorithms

Fault-Tolerant Distributed Algorithms (FTDAs)

- distributed algorithms should be reliable
- different fault models: crash, omission, Byzantine
- parameters
  - N - number of processes
  - T - upper bound on number of faults
  - F - actual number of faults
- resilience condition, eg. $N \geq 3T + 1$
- properties that must be satisfied

Example (Synchronous Consensus with Crash Faults)

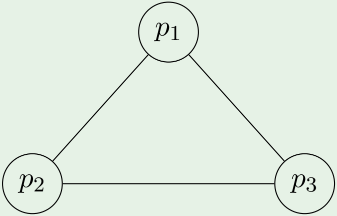


process p_i has

- v initial value
- W possible values
- d decision value

- resilience condition $N \geq T + 2$
- $N = 3, T = 1$

Example (Synchronous Consensus with Crash Faults)

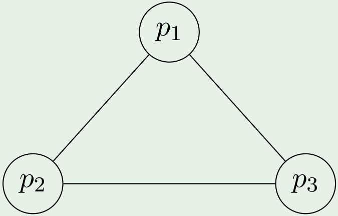


process p_i has

- v initial value
- W possible values
- d decision value

- resilience condition $N \geq T + 2$
- $N = 3, T = 1$
- validity: if all processes start with the same value, this is the only possible decision value

Example (Synchronous Consensus with Crash Faults)

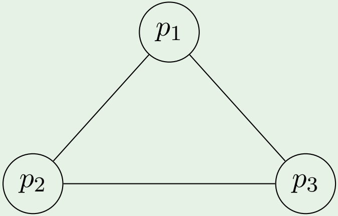


process p_i has

- v initial value
- W possible values
- d decision value

- resilience condition $N \geq T + 2$
- $N = 3, T = 1$
- validity: if all processes start with the same value, this is the only possible decision value
- agreement: no two correct processes decide on different values

Example (Synchronous Consensus with Crash Faults)

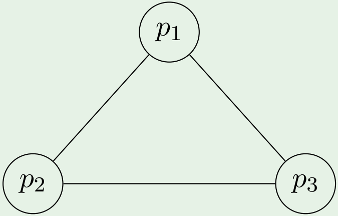


process p_i has

- v initial value
- W possible values
- d decision value

- resilience condition $N \geq T + 2$
- $N = 3, T = 1$
- validity: if all processes start with the same value, this is the only possible decision value
- agreement: no two correct processes decide on different values
- termination: all correct processes eventually decide

Example (Synchronous Consensus with Crash Faults)

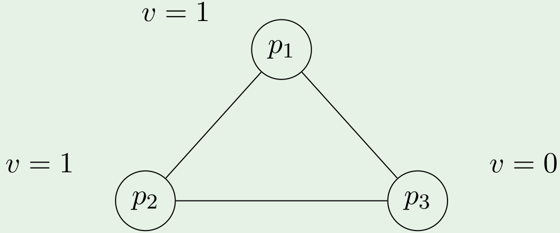


process p_i has

- v initial value
- W possible values
- d decision value

- resilience condition $N \geq T + 2$
- $N = 3, T = 1, F = 0$
- validity: if all processes start with the same value, this is the only possible decision value
- agreement: no two correct processes decide on different values
- termination: all correct processes eventually decide

Example (Synchronous Consensus with Crash Faults)

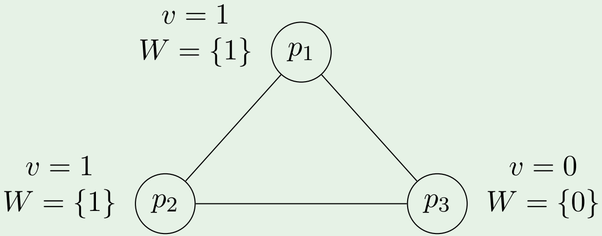


process p_i has

- v initial value
- W possible values
- d decision value

- resilience condition $N \geq T + 2$
- $N = 3, T = 1, F = 0$
- validity: if all processes start with the same value, this is the only possible decision value
- agreement: no two correct processes decide on different values
- termination: all correct processes eventually decide

Example (Synchronous Consensus with Crash Faults)

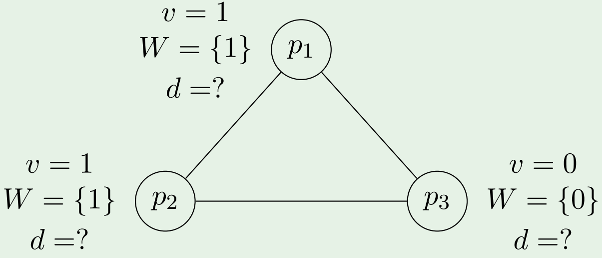


process p_i has

- v initial value
- W possible values
- d decision value

- resilience condition $N \geq T + 2$
- $N = 3, T = 1, F = 0$
- validity: if all processes start with the same value, this is the only possible decision value
- agreement: no two correct processes decide on different values
- termination: all correct processes eventually decide

Example (Synchronous Consensus with Crash Faults)

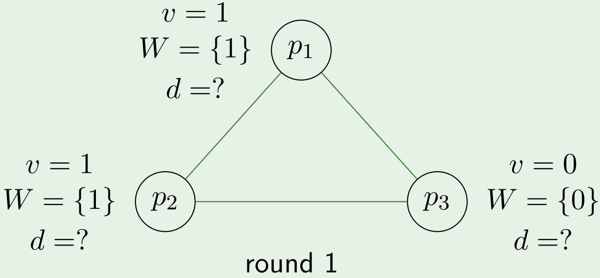


process p_i has

- v initial value
- W possible values
- d decision value

- resilience condition $N \geq T + 2$
- $N = 3, T = 1, F = 0$
- validity: if all processes start with the same value, this is the only possible decision value
- agreement: no two correct processes decide on different values
- termination: all correct processes eventually decide

Example (Synchronous Consensus with Crash Faults)

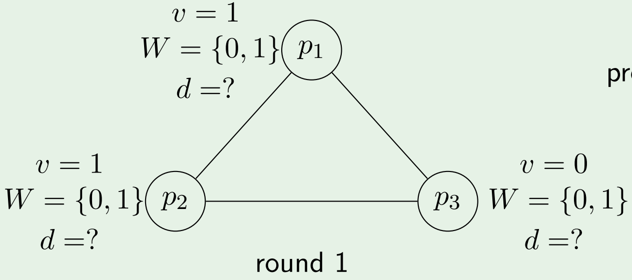


process p_i has

- v initial value
- W possible values
- d decision value

- resilience condition $N \geq T + 2$
- $N = 3, T = 1, F = 0$
- validity: if all processes start with the same value, this is the only possible decision value
- agreement: no two correct processes decide on different values
- termination: all correct processes eventually decide

Example (Synchronous Consensus with Crash Faults)

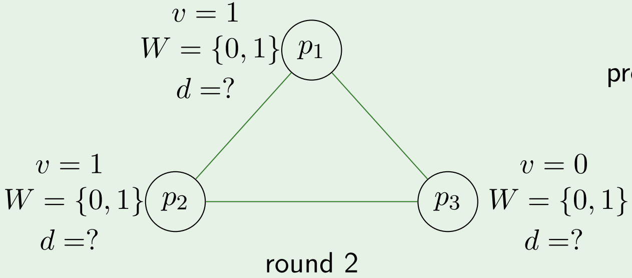


process p_i has

- v initial value
- W possible values
- d decision value

- resilience condition $N \geq T + 2$
- $N = 3, T = 1, F = 0$
- validity: if all processes start with the same value, this is the only possible decision value
- agreement: no two correct processes decide on different values
- termination: all correct processes eventually decide

Example (Synchronous Consensus with Crash Faults)

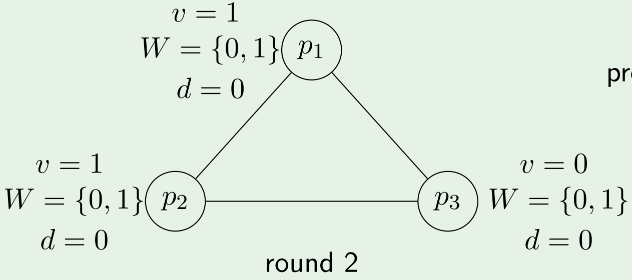


process p_i has

- v initial value
- W possible values
- d decision value

- resilience condition $N \geq T + 2$
- $N = 3, T = 1, F = 0$
- validity: if all processes start with the same value, this is the only possible decision value
- agreement: no two correct processes decide on different values
- termination: all correct processes eventually decide

Example (Synchronous Consensus with Crash Faults)

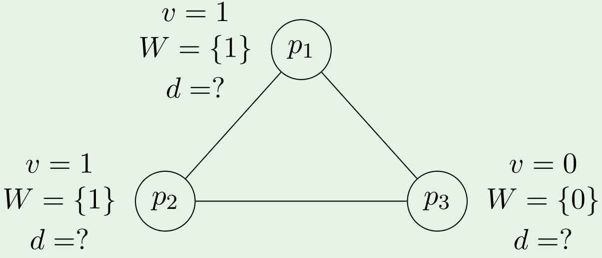


process p_i has

- v initial value
- W possible values
- d decision value

- resilience condition $N \geq T + 2$
- $N = 3, T = 1, F = 0$
- validity: if all processes start with the same value, this is the only possible decision value
- agreement: no two correct processes decide on different values
- termination: all correct processes eventually decide

Example (Synchronous Consensus with Crash Faults)

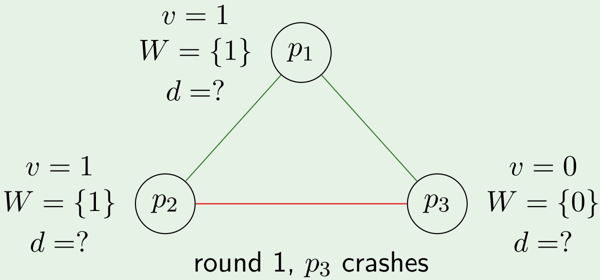


process p_i has

- v initial value
- W possible values
- d decision value

- resilience condition $N \geq T + 2$
- $N = 3, T = 1, F = 1$
- validity: if all processes start with the same value, this is the only possible decision value
- agreement: no two correct processes decide on different values
- termination: all correct processes eventually decide

Example (Synchronous Consensus with Crash Faults)

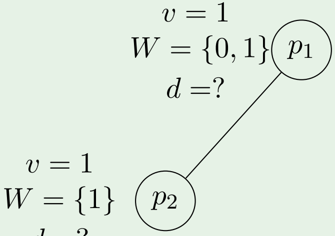


process p_i has

- v initial value
- W possible values
- d decision value

- resilience condition $N \geq T + 2$
- $N = 3, T = 1, F = 1$
- validity: if all processes start with the same value, this is the only possible decision value
- agreement: no two correct processes decide on different values
- termination: all correct processes eventually decide

Example (Synchronous Consensus with Crash Faults)



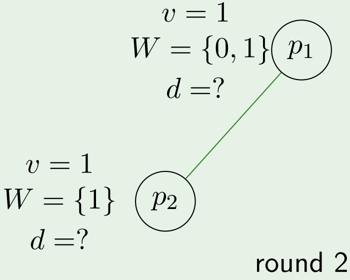
round 1, p_3 crashes

process p_i has

- v initial value
- W possible values
- d decision value

- resilience condition $N \geq T + 2$
- $N = 3, T = 1, F = 1$
- validity: if all processes start with the same value, this is the only possible decision value
- agreement: no two correct processes decide on different values
- termination: all correct processes eventually decide

Example (Synchronous Consensus with Crash Faults)

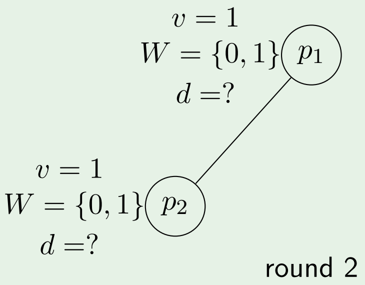


process p_i has

- v initial value
- W possible values
- d decision value

- resilience condition $N \geq T + 2$
- $N = 3, T = 1, F = 1$
- validity: if all processes start with the same value, this is the only possible decision value
- agreement: no two correct processes decide on different values
- termination: all correct processes eventually decide

Example (Synchronous Consensus with Crash Faults)

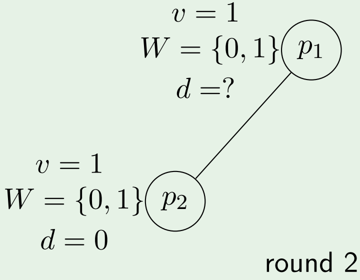


process p_i has

- v initial value
- W possible values
- d decision value

- resilience condition $N \geq T + 2$
- $N = 3, T = 1, F = 1$
- validity: if all processes start with the same value, this is the only possible decision value
- agreement: no two correct processes decide on different values
- termination: all correct processes eventually decide

Example (Synchronous Consensus with Crash Faults)

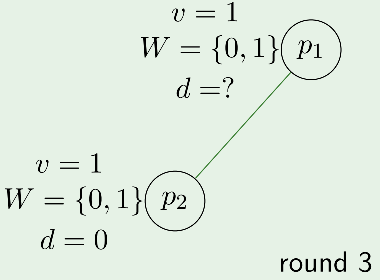


process p_i has

- v initial value
- W possible values
- d decision value

- resilience condition $N \geq T + 2$
- $N = 3, T = 1, F = 1$
- validity: if all processes start with the same value, this is the only possible decision value
- agreement: no two correct processes decide on different values
- termination: all correct processes eventually decide

Example (Synchronous Consensus with Crash Faults)

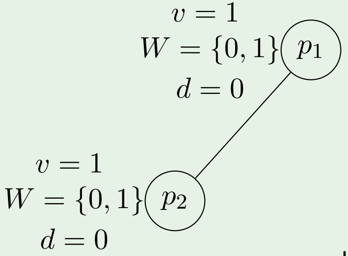


process p_i has

- v initial value
- W possible values
- d decision value

- resilience condition $N \geq T + 2$
- $N = 3, T = 1, F = 1$
- validity: if all processes start with the same value, this is the only possible decision value
- agreement: no two correct processes decide on different values
- termination: all correct processes eventually decide

Example (Synchronous Consensus with Crash Faults)



round 3

process p_i has

- v initial value
- W possible values
- d decision value

- resilience condition $N \geq T + 2$
- $N = 3, T = 1, F = 1$
- validity: if all processes start with the same value, this is the only possible decision value
- agreement: no two correct processes decide on different values
- termination: all correct processes eventually decide

Parametrized Verification of Fault-Tolerant Distributed Algorithms

Formal Verification

- guarantee that a system design is free of faults
- model checking: determine if a system model satisfies a specification
- given:

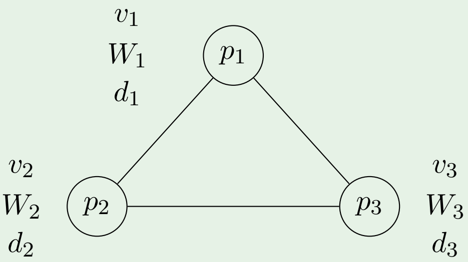
system model:	$\langle S, S_0, T \rangle$	S - set of states, $S_0 \subseteq S$ - set of initial states, $T \subseteq S \times S$ - transition relation
specification (property):	φ	logical formula

  do

  1. exhaustively examine the reachable states of the program
  2. check if the property is satisfied

- safety: nothing bad ever happens
- liveness: something good eventually happens

Example



each p_i is characterized by the local state

$$l_i = (v_i, W_i, d_i)$$

system state $s \in S$

$$s = \langle l_1, l_2, l_3 \rangle$$

initially we had

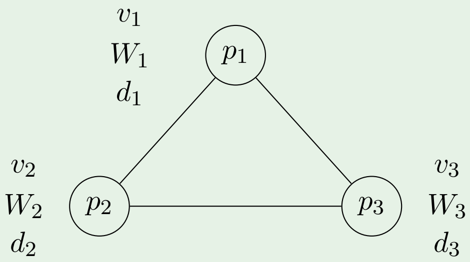
$$v_1 = 1, W_1 = \{1\}, d_1 =?$$
$$v_2 = 1, W_2 = \{1\}, d_2 =?$$
$$v_3 = 0, W_3 = \{0\}, d_3 =?$$

our initial state looked like:

$$s_0 = \langle (1, \{1\}, ?), (1, \{1\}, ?), (0, \{0\}, ?) \rangle$$

starting from s_0 we can generate all possible behaviours

Example



each p_i is characterized by the local state

$$l_i = (v_i, W_i, d_i)$$

system state $s \in S$

$$s = \langle l_1, l_2, l_3 \rangle$$

properties:

- validity: if all processes start with the same value, this is the only possible decision value safety
- agreement: no two correct processes decide on different values safety
- termination: all correct processes eventually decide liveness

Parametrized Verification of Fault-Tolerant Distributed Algorithms

Parametrized Verification

- guarantee there are no faults in a system of arbitrary size
- undecidable even in the absence of concurrency!
- additional challenges posed by FTDAs
  - unbounded parameters
  - non-determinism
  - state space explosion

Abstraction

- simulate an infinite system using a finite one

$$\langle S, S_0, T \rangle \xrightarrow{\alpha} \langle \hat{S}, \hat{S}_0, \hat{T} \rangle$$

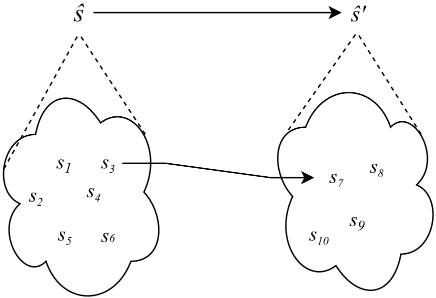
- overapproximation
- precision is traded for efficiency
- reason about properties of the concrete system by reasoning about the abstract system

$$\text{if } \langle \hat{S}, \hat{S}_0, \hat{T} \rangle \models \hat{\varphi} \text{ then } \langle S, S_0, T \rangle \models \varphi$$

Our Approach

How do we Tackle the Problem?

- specification language: TLA+
- model checking: TLC
- new kind of existential abstraction
- abstract states keep track whether a process in a certain state exists



Current State

- a new abstraction technique defined
- one synchronous consensus algorithm with crash faults formalized
  - checked for system sizes up to $N = 7$
  - even for a small system sizes, state space explosion cannot be avoided!
- abstraction of the consensus algorithm
  - safety properties verified
- search for algorithms that can be abstracted

Future Directions

- improve abstraction, capture other classes of algorithms
  - different timing models: asynchronous, partially synchronous
  - different fault models: omission, Byzantine faults
  - different problems: mutual exclusion, cache coherence...
- investigate liveness

Thank you!