Pardi – avancement

Acadie

11 juillet 2017

Plan

1. Case Studies
2. Framework
3. Future Work

Case Studies

Distributed Algorithms

- Two Phase Commit
- Chandy-Misra Mutual Exclusion
- Naimi-Tréhel Mutual Exclusion
- Token-based Mutual Exclusion
- Misra Termination
- Consensus

Workflow

- Examination Management System
- Shift Worker Scheduling

Shared Memory

- Splitter
- Renaming
- Linked List

Number of Variants

Problem	TLA^+ (& PlusCal)	Cubicle	Workflow
Two Phase Commit	2	3	
Consensus	10	–	
Chandy-Misra Mutual Exclusion	1	1	
Naimi-Tréhel Mutual Exclusion	4	1	
Misra Termination	3	–	
Token-based Mutual Exclusion	5	5	
Examination Management Sys.	1	3	2
Shift Worker Scheduling	–	–	1
Splitter	1	3	
Renaming	1	–	
Linked List	1	–	

Numerical Parameters

Number of Processes

- No knowledge necessary, except $\forall site$, $\exists site$, $\exists j \neq self$ (splitter, Naimi-Tréhel)
- Set operations ($\in$, $\subseteq$) (Chandy-Misra, Two Phase Commit)
- Structural properties: $next(self)$ (Misra Termination, token-based mutual exclusion)
- Explicit count ($+1$ and test $= N$) (Misra termination, synchronous consensus, renaming)

Numerical Parameters

Number of Failures

For instance, synchronous consensus:

site failure	constraint
none	ok
crash	f failures $< n$ processes
omission	$f < n/2$
byzantine	$f \leq \lfloor (n-1)/3 \rfloor$

Exchanged Values

(e.g. consensus) $\rightarrow$ can be instantiated with a few distinct values (few $= 2$ usually)

Additional Parameters

k-consensus, with a relation to the number of failures and the number of processes (e.g. asynchronous communication and crash failure $\Rightarrow f < k < n$)

Functional Parameters

Network Topology

Mesh, Ring, spanning tree...

Process Failures

Crash failure, omission failure, byzantine failure.
Dedicated algorithms → not a parameter

Communication Parameters

- Multiplicity (point-to-point, multicast, broadcast)
- Synchronous / asynchronous (also computation model, e.g. by round)
- Failures (message loss, duplication, corruption)
- Delivery ordering (FIFO 1-1, causal...)

Results

Problem	TLA$^+$ (N max)	Cubicle	Workflow
Two Phase Commit	N=7	∼OK	
Consensus	N=2 to 4	no model	
Chandy-Misra Mutual Exclusion	N=4	OK	
Naimi-Tréhel Mutual Exclusion	N=5	∼OK	
Misra Termination	N=5	no model	
Token-based Mutual Exclusion	N=6	OK	
Examination Management Sys.	fixed	KO	collab. & chor.
Shift Worker Scheduling	no model	no model	collab.
Splitter	N=4	KO	
Renaming	N=3	no model	
Linked List	NP+NC=7	no model	

N max: for light TLC verification

Framework for Compatibility Checking

Modeling

- **Asynchronously** communicating peers
- Multi-senders multi-receivers channels
- Group of channels associated to different communication models

Method

- Peers and communication specified using transition systems
- Compatibility checking of LTL properties
- Modular (new models)
- Fully automatic
- **Model checking**: finite state systems
- TLA$^+$

New Framework for Compatibility Checking

Two point-to-point channels, with fifo11 ordering, and at most two messages in transit on channel `Request`.

```
MODELS == {[name |-> "p2p",
              params |-> [chan |-> {"Request", "Token"} ]],
            [name |-> "fifo11",
              params |-> [chan |-> {"Request", "Token"} ]],
            [name |-> "message_cap",
              params |-> [chan |-> {"Request"}, bound |-> 2]]}
```

A communication action (send/receive) on a channel is enabled if it is enabled in all concerned models.

(Adam Shimi's Master thesis)

Sending & Receiving

```
Request(i) ==
  ∧ ~ requesting[i]
  ∧ requesting' = [ requesting EXCEPT ![i] = TRUE ]
  ∧ COM!send(i, {father[i]}, "Request", i)
       (* sender, destination, channel, content *)
  ∧ ...


ReceiveToken(i) ==
    ∧ ∃ j ∈ Sites : COM!receive(j, i, "Token", 0)
                    (* sender, receiver, channel, content *)
    ∧ token' = [ token EXCEPT ![i] = TRUE ]
    ∧ ...
```

Case Studies

Case studies implemented in our framework:

- Chandy-Misra Mutual Exclusion
- Naimi-Tréhel Mutual Exclusion
- Consensus (asynchronous)
- Examination Management System

$\rightarrow$ easy tests and confirmations of the required ordering of delivery.

Limitations:

- One communication action per transition (no multiple sends, no receive-and-reply)
- Generic logical action vs ad-hoc implementation:
  $\exists i, j \in Site : \exists d \in Data : COM!receive(i, j, "chan", d) \wedge \ldots$
  $\text{LET } m = Head(chan[i][j]) \text{ IN} \ldots$
  $\rightarrow$ same number of states/transitions but slower

Demonstration?

Demonstration?
To be prepared. . .

And Now?

- Framework: Failure $\rightarrow$ Initial work not satisfactory $\rightarrow$ Heard-Of?
- Minimal parameters?
- Cubicle with weak variables $=$ channels?
- Other workflow examples?