Reasoning with TLA+

Praseem Banzal
Suvidha Kancharla
Gowtham Kaki

PURDUE
UNIVERSITY

A Recap of TLA & TLA+

- TLA - Temporal Logic of Actions (Lamport, 1993).
- A logic that lets us
  - Abstractly describe the behaviour of concurrent algorithms (eg: FIFO reliable broadcast).
  - Precisely state its properties, such as FIFO order and validity, whose English descriptions contain words like "before", "after", "never" and "eventually".
- TLA+ is TLA with syntactic sugar.
- TLC is a model checker to check TLA+ specs on finite instances

Project Goals

- Write TLA+ implementations, and the specifications of their interfaces for three distributed algorithms:
  - Reliable Broadcast
  - FIFO Reliable Broadcast
  - Causal-Order Reliable Broadcast
- Construct right models for model-checking to be effective.
- See if model-checking can find answers to questions such as : Does adding uniform agreement to reliable broadcast provide uniform agreement at FIFO reliable broadcast?
- Provide an experience report quantifying the effort and identifying subtle problems in writing TLA+ specs of distributed algorithms.

Demo

PURDUE
UNIVERSITY

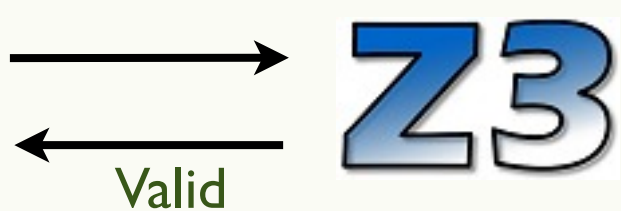
Project Goals (2)

- If time permits:
  - Identify those properties which can be proved automatically (using an SMT solver) by reasoning in first-order logic.
  - Give SMT-LIB encoding of such properties, and statistics on amount of time that solver took to discharge proof obligations.

Can we automate?

```
---------------------- MODULE HourClock ----------------------
EXTENDS Naturals
VARIABLE hr
HCini  ==  hr \in (1 .. 12)
HCnxt  ==  hr' = IF hr # 12 THEN hr + 1 ELSE 1
HC  ==  HCini /\ [][HCnxt]_hr
--------------------------------------------------------------
THEOREM  HC => []HCini
==============================================================
```

```
hr[0] ∈ {0 .. 12}
∀t.  (hr[t] ≠ 12 ⇒ hr[t+1] = hr[t]+1)
   ∧ (hr[t] = 12 ⇒ hr[t+1]=1)
------------------------------------
∀t. hr[t] ∈ {0 .. 12} ??
------------------------------------
```



How far can we stretch this approach?

Reliable Channel in SMT-LIB

```
(define-sort Time () Nat)
```

Time is a natural number

Every variable is a function of time

```
; ReceiveSet and DeliverSet
(declare-fun rcvSet (Time Message) Bool)
(declare-fun dvrSet (Time Message) Bool)
```

```
; sending at time t...
(define-fun sendMsg ((t Time)(m Message)) Bool
  (and (incRcv t m) (invDvr t)))
```

```
; involves adding message to rcvSet
(forall ((m1 Message))
  (= (rcvSet (S t) m1) (or (rcvSet t m1) (= m1 m)))))
```

Reliable Channel in SMT-LIB (2)

- At any given time t:
  - sendMsg at t, or
  - rcvMsg at t, or
  - stutter (i.e, do nothing)

```
; Next State Predicate, on any given time t.
(define-fun Next ((t Time)) Bool
    (xor (exists ((m Message))
             (sendMsg t m))
         (rcvMsg t)
         (and (invRcv t) (invDvr t)))
)
(assert (and Init (forall ((t Time)) (Next t))))
```

Reliable Channel in SMT-LIB (3)

```
;; Consider a time t
(declare-const t Time)
;; and a message m,
(declare-const m Message)
;; which is sent now
(assert (sendMsg t m))
;; receive is going to be executed some time later
(declare-const WF-rcv-t Bool)
(assert (= WF-rcv-t
  (exists ((t1 Time)) (and (less-than t t1) (rcvMsg t1)))))
(assert WF-rcv-t)

;; Does there exist a time where the sent message is delivered?
(declare-const ev-t Bool)
(assert (= ev-t
    (exists ((t1 Time)) (and (less-than t t1) (dvrQ t1 m)))))
(assert (not ev-t))
(check-sat)
```

Weak Fairness Condition

Validity

Limits of explicit time approach

- We also wrote Reliable Broadcast in first-order logic, with same behaviour and specifications as TLA+
  - Orders of magnitude easier than writing TLA+ specs
- We found that explicit time encoding in first-order logic is as expressive as TLA+ (unsurprising result, as TLA $\subseteq$ FOL)
- Model Checking is also possible - Z3 SMT solver can generate models for finite instances of the problem
- For reliable channel, with help of some clever encoding, we even have an automatic proof!
- However, prover times out on reliable broadcast.

Conclusion

- We have shown that TLA is indeed expressive enough to describe concurrent algorithms, such as FIFO broadcast
- However, the spec language TLA+ camouflages semantics, needlessly complicated, & not at all composable.
- IO Automata based IOA language is a better alternative.
- We demonstrated simple first-order logic encoding of reliable channel & broadcast using explicit time encoding.
  - Straightforward semantics. Easy to understand.
  - No separate model checker. Off-the-shelf SMT solver is enough.
- Formal methods are also useful for "rapid prototyping"

Future Work

- A promising avenue for future research is to combine temporal reasoning methods (eg: Buechi Tree Automata) with our methods of first-order encoding (eg: Time-indexed values, Pure relational representation of lists etc) to construct a decision procedure for a well-defined class of concurrent algorithms and temporal properties.
- Extend specification languages, like TLA+ and IOA, with monadic type system to cleanly reason with state (eg: message queues.)

Thank you!

PURDUE
UNIVERSITY

Written Assignment-1 Question

- 3. Take the Causal Atomic Broadcast seen in class
  - a. Does it implement Uniform Causal Order? Why (not)?
  - b. Suppose the underlying FIFO Atomic Broadcast algorithm provides Uniform Agreement instead of Agreement (i.e., I, II, III, V, IX, XIII instead of I, II, III, IV, IX, XIII). Does the resulting algorithm also provide Uniform Agreement? Can we simplify it?

Written Assignment-1 Question

- 3. Take the Causal Atomic Broadcast seen in class
  - a. Does it implement Uniform Causal Order? Why (not)?
  - b. Suppose the underlying FIFO Atomic Broadcast algorithm provides Uniform Agreement instead of Agreement (i.e., I, II, III, V, IX, XIII instead of I, II, III, IV, IX, XIII). Does the resulting algorithm also provide Uniform Agreement? Can we simplify it?
- Ans : No. There exists a counter-example.
- Ok, but how do we arrive at this conclusion? Trial and error?