reTLA:
Towards an automatic transpiler from TLA+ to VMT

TlaConf 2022 @ St. Louis, MO

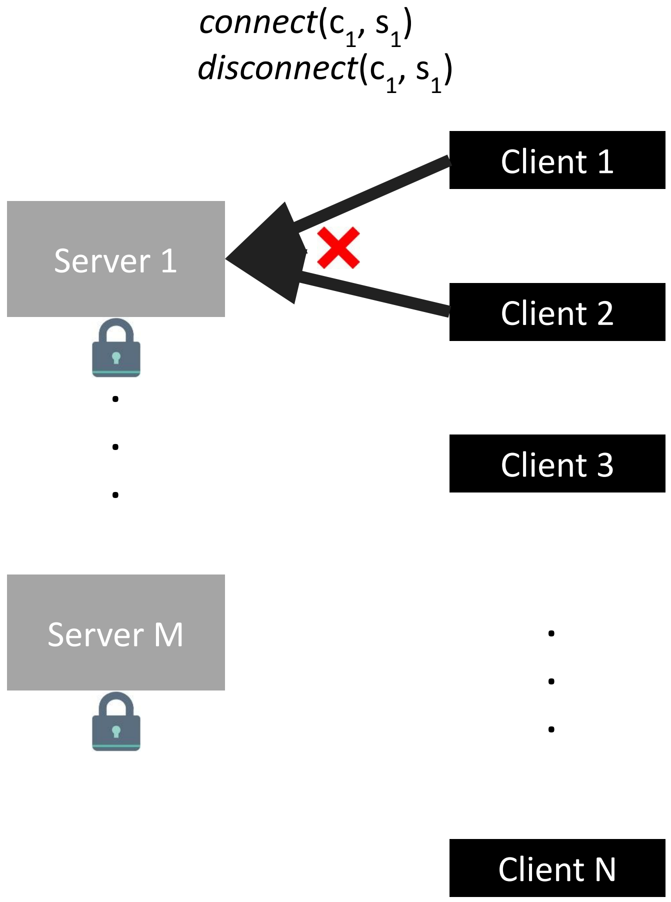
Jure Kukovec (IS), Aman Goel (UM), Igor Konnov (IS),
Stephan Merz (IN), and Karem Sakallah (UM)

Example: Client Server Protocol

- Any number of clients & servers
- Each client can connect/disconnect to a server

Safety property:

Each server can be connected to at most 1 client

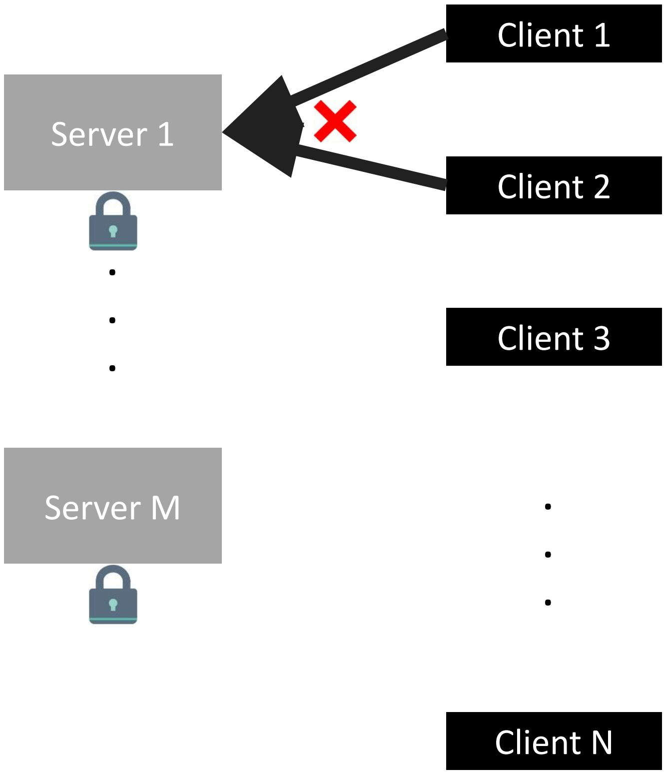


Relational Encoding in Ivy

Ivy-
http://microsoft.github.io/ivy

connect(c_1, s_1)
disconnect(c_1, s_1)

```
type client
type server

relation semaphore(X:server)
relation link(X:client, Y:server)

after init {
    forall Y.    semaphore(Y) := true;
    forall X, Y. link(X, Y)   := false;
}

action connect(c: client, s: server) = {
    require semaphore(s);
    link(c, s)    := true;
    semaphore(s) := false;
}

action disconnect(c: client, s: server) = {
    require link(c, s);
    link(c, s)    := false;
    semaphore(s) := true;
}

invariant forall C1, C2: client, S: server.
          link(C1, S) & link(C2, S) -> C1 = C2
```



Relational Encoding in Ivy

```
type client
type server

relation semaphore(X:server)
relation link(X:client, Y:server)

after init {
    forall Y.    semaphore(Y) := true;
    forall X, Y. link(X, Y)   := false;
}

action connect(c: client, s: server) = {
    require semaphore(s);
    link(c, s)    := true;
    semaphore(s) := false;
}

action disconnect(c: client, s: server) = {
    require link(c, s);
    link(c, s)    := false;
    semaphore(s) := true;
}

invariant forall C1, C2: client, S: server.
          link(C1, S) & link(C2, S) -> C1 = C2
```

Quantified formulas using relations/functions over uninterpreted domains

- Infinite-state system
- Learn a quantified inductive invariant

Initial state formula

Transition relation

Safety property

IC3PO

informal SYSTEMS

IC3PO's Key Ingredients

Finite-Domain Model Checking

Leslie Lamport <[redacted]>: Apr 15 09:45AM -0700

While large sets can cause performance problems, it's rare for an algorithm to be correct for a set of 3 elements and not for a set of 1000 elements.

Spatial & Temporal Regularity

Symmetry & Range Boosting using Protocol's Domain Regularities

Regularity ↔ Quantification

Compact Quantified Clause Learning

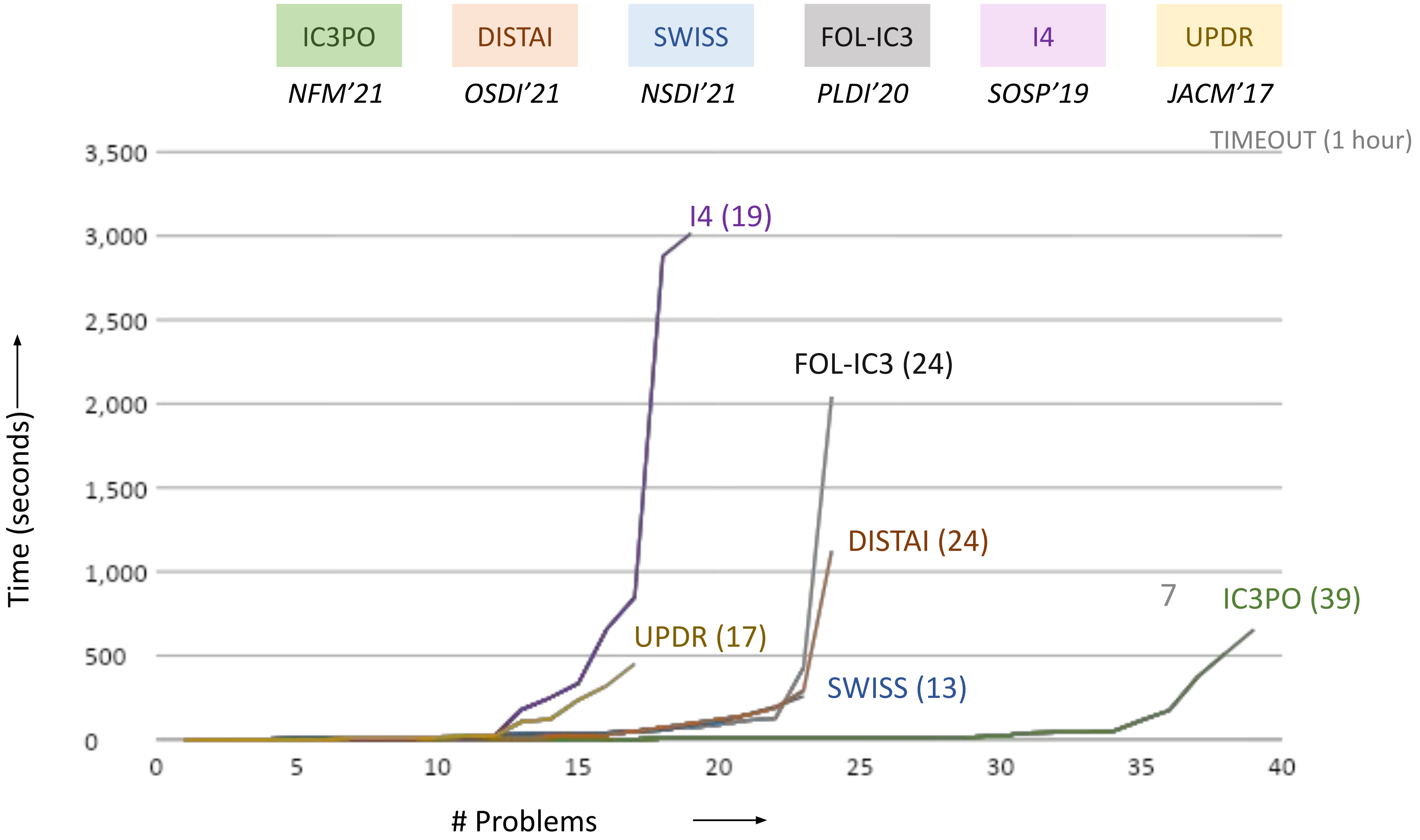
Finite Convergence

Automatically reach *Cutoff*

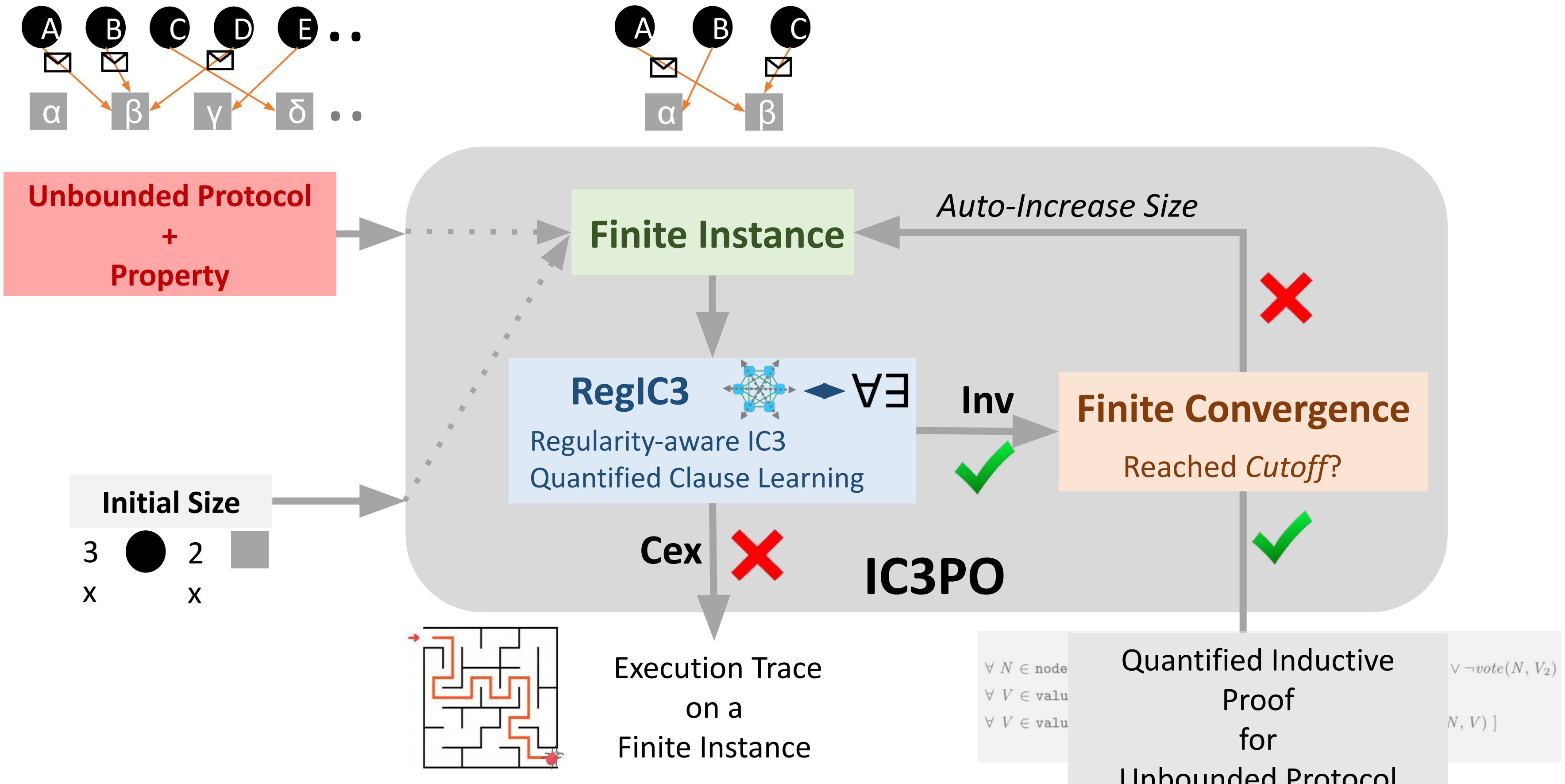
Hierarchical Structure

Hierarchical Strengthening for High Scalability

Automatic Quantified Inductive Invariant Inference



IC3PO: IC3 for Proving Protocol Properties

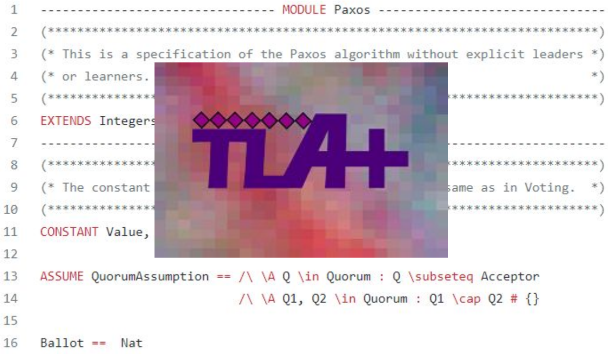


https://github.com/aman-goel/ic3po

GOALS

informal SYSTEMS

Goal: Automatic Inductive Invariant Inference for TLA+

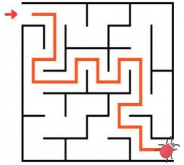


```
-------------------------------- MODULE Paxos -------------------------------
(***************************************************************************)
(* This is a specification of the Paxos algorithm without explicit leaders *)
(* or learners.                                                           *)
(**************                                      ********************)
EXTENDS Integers
-------------                                      -----------------------
(**************                                      ********************)
(* The constant                                    same as in Voting.   *)
(**************                                      ********************)
CONSTANT Value,

ASSUME QuorumAssumption == /\ \A Q \in Quorum : Q \subseteq Acceptor
                           /\ \A Q1, Q2 \in Quorum : Q1 \cap Q2 # {}

Ballot ==  Nat
```


IC3PO

Quantified Inductive Proof for Unbounded Protocol

Property obeyed under ALL executions for ***any*** size



Execution Trace on a Finite Instance

Goal: Automatic Inductive Invariant Inference for TLA+

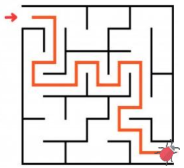
APALACHE

VMT-LIB

IC3PO

Quantified Inductive Proof for Unbounded Protocol

Property obeyed under ALL executions for ***any*** size



Execution Trace on a Finite Instance

[1] https://apalache.informal.systems
[2] https://vmt-lib.fbk.eu

reTLA: relational TLA+

informal SYSTEMS

Basic reTLA syntax

- Literals:
  - TRUE, FALSE
  - ..., -1, 0, 1, ...
  - "a", "b", ...
  - "1_OF_T", "X_OF_Y", ...
- Restricted sets:
  - Int, Nat, BOOLEAN
  - CONSTANT-declared with type Set(T)
- (In)equality: $=$, $\neq$
- Boolean operators: $\land$, $\lor$, $\Rightarrow$, $\Leftrightarrow$, $\neg$
- Quantified expressions: $\exists x \in S: P$, $\forall x \in S: P$
  - S must be a restricted set
- Functions:
  - Definitions: $[x_1 \in S_1, \ldots, x_n \in S_n \mapsto e]$, restricted set domains
  - Updates: [f EXCEPT ![x] = y]
  - Applications: f[x]

informal SYSTEMS

Limiting integers

- Full integer theory not supported downstream
- We want a strict total order: <
- TLA+ integers used as syntax sugar for uninterpreted sort with axiomatic total order
  - Specification uses literals 1, 8, 71 $\rightsquigarrow$ encoding defines constants a, b, c and asserts a < b < c
  - 4 < a and a < 6 do not imply a = 5 (reTLA integers are just sugar!)

informal SYSTEMS

Examples

Two-phase commit

- 1 Transaction manager (TM)
  +
  N resource managers (RM)
- Phase 1:
  All RMs must Prepare
- Phase 2:
  All RMs must Commit
- Nondeterministic Aborts

```
CONSTANT
    @type: Set(RM);
  RM

VARIABLES
   @type: RM → Str;
  rmState,
   @type: Str;
  tmState,
   @type: Set(RM);
  tmPrepared,
   @typeAlias: message =
    Commit(NIL)
   | Abort(NIL)
   | Prepared(RM);
   @type: Set($message);
  msgs
```

```
CONSTANT
    @type: Set(SORT_RM);
  Values_RM

VARIABLES
   @type: SORT_RM → SORT_STATE;
  rmState,
   @type: SORT_STATE;
  tmState,
   @type: SORT_RM → Bool;
  tmPrepared,
    @type: SORT_RM → Bool;
  msgsPrepared,
   @type: Bool;
  msgsCommit,
   @type: Bool;
  msgsAbort
```

What changes

$$
\begin{array}{l}
\text{@type: } (RM) \Rightarrow Bool; \\
RMPrepare1(rm) \triangleq \\
\quad \land\, rmState[rm] = \text{"working"} \\
\quad \land\, rmState' = [rmState \text{ EXCEPT } ![rm] = \text{"prepared"}] \\
\quad \boxed{\land\, msgs' = msgs \cup \{MkPrepared(rm)\}} \\
\quad \land\, \text{UNCHANGED } \langle tmState, tmPrepared \rangle
\end{array}
$$

$$
\begin{array}{l}
\text{@type: } (SORT\_RM) \Rightarrow Bool; \\
RMPrepare2(rm) \triangleq \\
\quad \land\, rmState[rm] = \text{"working\_OF\_SORT\_STATE"} \\
\quad \land\, rmState' = [rmState \text{ EXCEPT } ![rm] = \text{"prepared\_OF\_SORT\_STATE"}] \\
\quad \boxed{\land\, msgsPrepared' = [msgsPrepared \text{ EXCEPT } ![rm] = \text{TRUE}]} \\
\quad \land\, \text{UNCHANGED } \langle tmState, tmPrepared, msgsAbort, msgsCommit \rangle
\end{array}
$$

informal SYSTEMS

From TLA+ to reTLA?

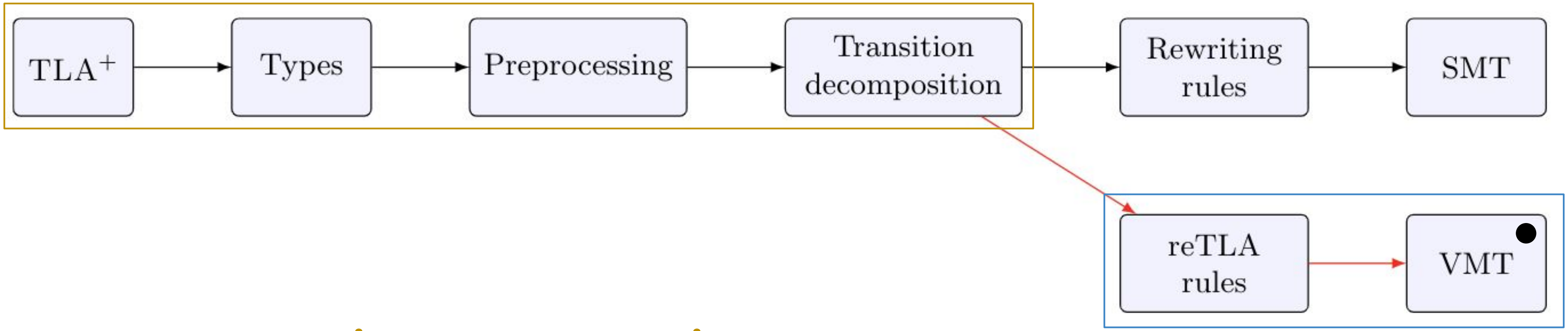
Set-function duality

Set-theoretic view	Function view
$S, T \subseteq U$	$f, g: U \to \text{Bool}$
$x \in S$	$f[x] = \text{TRUE}$
$S \cap T$	$[\, x \in U \mapsto f[x] \wedge g[x] \,]$
$\{\, x \in S: P(x) \,\}$	$[\, x \in U \mapsto f[x] \wedge P(x) \,]$
$\{\, Q(x): x \in S \,\}$, $Q: U \to V$	$[\, y \in V \mapsto \underline{\exists x \in U}: f[x] \wedge Q(x) = y \,]$
$\{\, Q(x): x \in S \,\}$, invertible $Q: U \to V$	$[\, y \in V \mapsto f[Q^{-1}(y)] \,]$

informal SYSTEMS

Reduce, reuse, recycle

Revised Apalache pipeline



- Keep parsing & preprocessing
- Re-implement (simplified) rules
- Output constraints instead of running the solver directly

informal SYSTEMS

Example: f[x] rule in TLA+

$$\dfrac{\langle \mathsf{c}[\mathsf{c}_{arg}]_{\mathrm{F}} \mid \mathcal{A} \mid \nu \mid \Phi \rangle \qquad \dfrac{\mathsf{c} \rightarrow_{\mathcal{A}} \mathsf{c}_{\mathsf{dom}}, \mathsf{c}_{\mathsf{cdm}} \quad \langle \mathrm{FROM}\ \mathsf{c}_{\mathsf{cdm}} \mid \mathcal{A} \mid \nu \mid \Phi \rangle}{\langle \mathsf{c} \mid \mathcal{A}_2 \mid \Phi_2 \mid \nu_2 \rangle} \qquad \mathsf{c}_{\mathsf{dom}} \rightarrow_{\mathcal{A}} \mathsf{c}_1, \dots, \mathsf{c}_n}{\langle \mathsf{c} \mid \mathcal{A}_2 \mid \nu_2 \mid \Phi_2, \mathit{FunRes} \rangle} \ (\textsc{FunApp})$$

$$\bigvee_{1 \le i \le n} in(\mathsf{c}_i, \mathsf{c}_{\mathsf{dom}}) \wedge \mathsf{c}_i = \mathsf{c}_{arg} \wedge \mathsf{c}_{res} = fun_{\mathsf{c}}(\mathsf{c}_i) \qquad (\mathit{FunRes})$$

informal SYSTEMS

Example: f[x] rule in reTLA

$$\frac{\langle \mathsf{c}[\mathsf{c}_{arg}]_{\mathrm{F}} \mid \mathcal{A} \mid \nu \mid \Phi \rangle \qquad \mathsf{c} \rightarrow_{\mathcal{A}} \mathsf{c}_{\mathsf{dom}}, \mathsf{c}_{\mathsf{cdm}} \qquad \mathsf{c}_{\mathsf{dom}} \rightarrow_{\mathcal{A}} \mathsf{c}_1, \ldots, \mathsf{c}_n \quad \langle \text{FROM } \mathsf{c}_{\mathsf{cdm}} \mid \mathcal{A} \mid \nu \mid \Phi \rangle}{} \quad (\textsc{FunApp})$$

$$\bigvee_{1 \le i \le n} in(\mathsf{c}_i, \mathsf{c}_{\mathsf{dom}}) \wedge \mathsf{c}_i = \mathsf{c}_{arg} \wedge \mathsf{c}_{res} = fun_{\mathsf{c}}(\mathsf{c}_i) \qquad (FunRes)$$

$$\boxed{\frac{f \rightarrow g \qquad x \rightarrow y}{f[x] \rightarrow (\mathrm{g\ y})} \quad (\textsc{reTLAFunApp})}$$

informal SYSTEMS

<VIDEO>

Experiments

Initial Experiments

Client Server

1 sec

/\ *Property*
/\ (forall S1, C1 . (clientlocks(C1, S1) -> ~semaphore(S1)))

TCommit

1 sec

/\ *Property*

TwoPhase

4 sec

/\ *Property*
/\ (msgsCommit -> (committed_SORT_STATE = tmState))
/\ (msgsAbort -> (tmState = aborted_SORT_STATE))
/\ (forall S1 . ((rmState(S1) = committed_SORT_STATE) -> msgsCommit))
/\ (forall S1 . (msgsCommit -> ((prepared_SORT_STATE = init_SORT_STATE) | …
/\ (forall S1 . (tmPrepared(S1) -> msgsPrepared(S1)))
/\ (forall S1 . ((msgsPrepared(S1) & (init_SORT_STATE = tmState)) -> …

Sharded Key-Value

8 sec

/\ *Property*
/\ (forall N2, N1, K1, V1 . (owner(N1, K1) -> ~transfer_msg(N2, K1, V1)))
/\ (forall N2, N1, K1, V1 . ((transfer_msg(N1, K1, V1) & transfer_msg(N2, K1, V1)) -> (N2 = N1)))
/\ (forall N2, N1, K1, V1 . (transfer_msg(N2, K1, V1) -> (table(N1, K1) = Nil)))
/\ (forall N2, N1, K1 . (owner(N1, K1) -> ((table(N2, K1) = Nil) | (N2 = N1))))
/\ (forall K1, N1 . (((table(N1, K1) = Nil) & owner(N1, K1)) -> (start = N1)))
/\ (forall V2, N2, N1, K1, V1 . ((transfer_msg(N1, K1, V1) & transfer_msg(N2, K1, V2)) -> (V1 = V2)))

Decentralized Lock

3 sec

/\ *Property*
/\ (forall N2, N3, N1 . ((message(N3, N2) & message(N3, N1)) -> (N2 = N1)))
/\ (forall N1, N2, N3 . (message(N3, N2) -> ~has_lock(N1)))
/\ (forall N1, N4, N2, N3 . ((message(N1, N4) & message(N3, N2)) -> (N3 = N1)))

Initial Experiments: Initial vs Cutoff Sizes

Protocol	Initial Size	Cutoff Size
Client Server	|C|=1, |S|=1	|C|=2, |S|=1
TCommit	|SORT_RM|=1, |SORT_STATE|=4	|SORT_RM|=1, |SORT_STATE|=4
TwoPhase	|SORT_RM|=1, |SORT_STATE|=4	|SORT_RM|=2, |SORT_STATE|=5
Sharded Key-Value	|K|=1, |N|=1, |V|=1	|K|=1, |N|=2, |V|=3
Decentralized Lock	|N|=1	|N|=4

Future work

- Automatic translation of TLA+ to reTLA
- Identifying the maximal translatable fragment
- Tendermint in reTLA

https://github.com/aman-goel/ivybench/tree/master/tla

Thanks!

Questions? … jure@informal.systems

informal SYSTEMS