Protocol verification

Hard problem
Separable from software
I4: inductive invariant.
Reachable state.

Hard problem
Async. network:
Delay
Drop
Dup
Corrupt.

Node failures:
Crash
Restart
Inconsistencies.

Open systems: membership,
Misbehave
"Byzantine".
Ivy: protocol verification tool
Protocol reasoning
Close analogue: TLA.

IronFleet:
type client
    type server

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

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

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

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

export connect
export disconnect

invariant [1] link(C1, S) & link(C2, S) -> C1 = C2
Safety of reachable states

Hard?
- Reachable states defined transitively.
- Spec set: well defined but not inductive.
- Proof plan: inductive invariant

Spec allows: safety invariant.

All possible states.
Ivy proof: safety of reachable states

1.) \( \forall s \in \text{init}, \quad \text{Inv}(s) \).

2.) \( \forall s, s', \quad \text{Inv}(s) \land \text{step}(s, s') \Rightarrow \text{Inv}(s') \)

\( \text{Inv} = \text{over-approx of reachable states.} \)
Model checking: safety of reach. states

Bounded stack size

\[ S \quad C_0 \quad C_1 \]

\[ \text{sem}(s) \quad \text{link}(C_0, s) \quad \text{link}(C_2, s) \]

\[ \begin{array}{ccc}
T & F & F \\
F & T & F \\
F & F & T \\
\end{array} \]

init:

safety spec

AVR: synthesize compact predicate for reachable states.

T4's trick:

AVR predicate \( \rightarrow \) my invariant.
I4's workflow

Ivy protocol → I4 → Finite version (AVR) → AVR → Pred of reachable states → I4 → Pred of unbounded states (inductive, hopefully) → Ivy → OK.

Bigger bounds