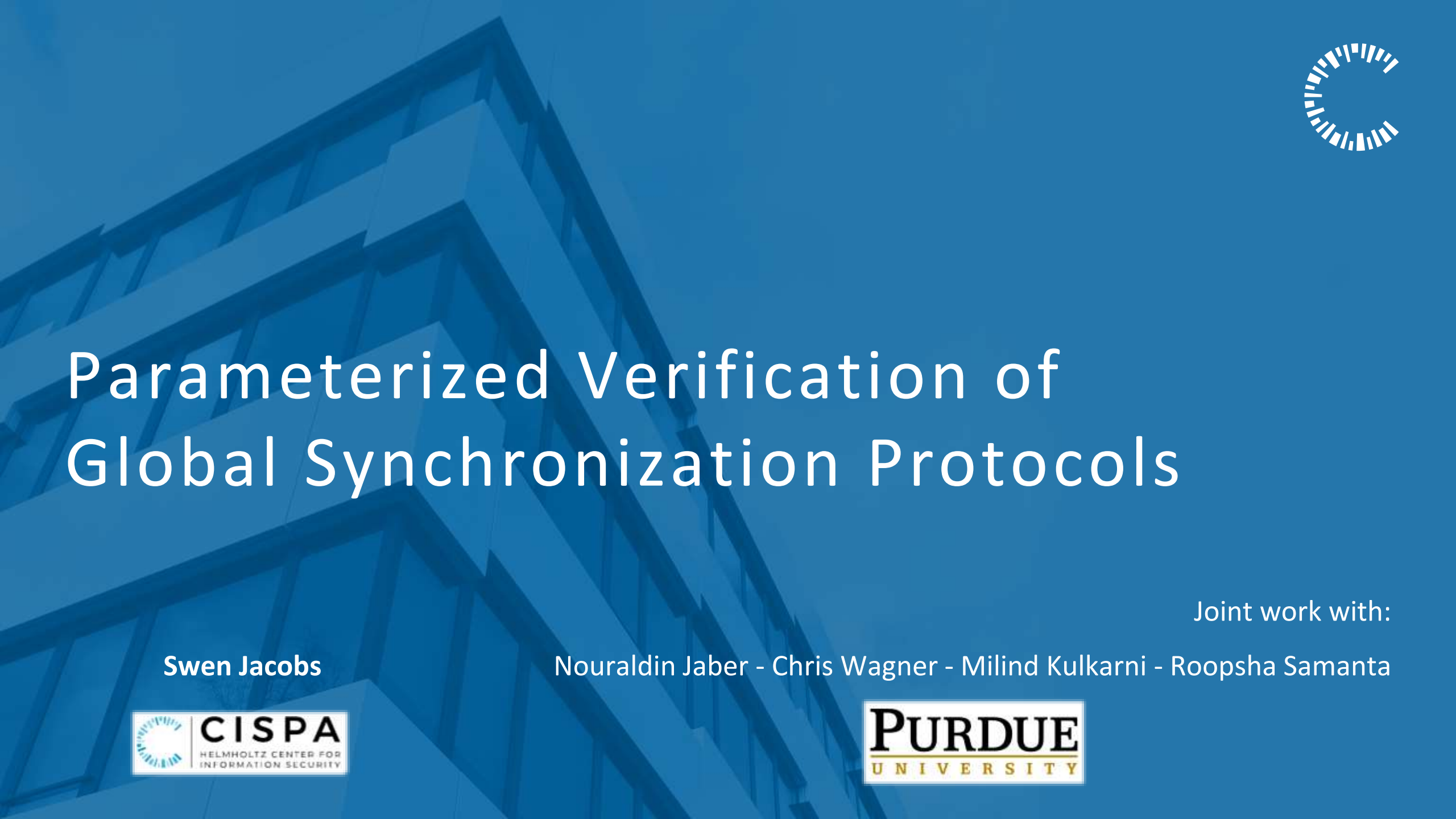


Parameterized Verification of Global Synchronization Protocols

Swen Jacobs

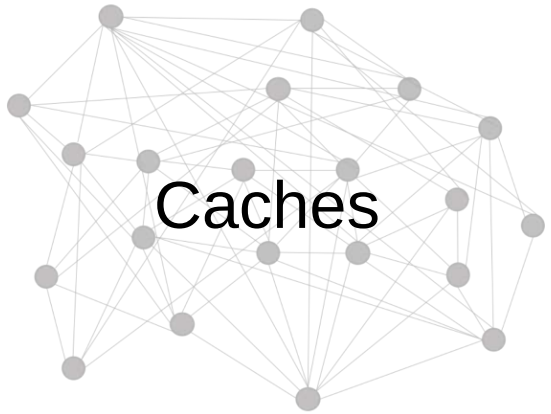
CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY

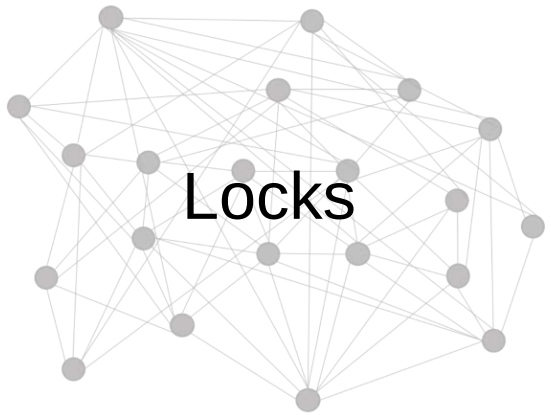
Joint work with:

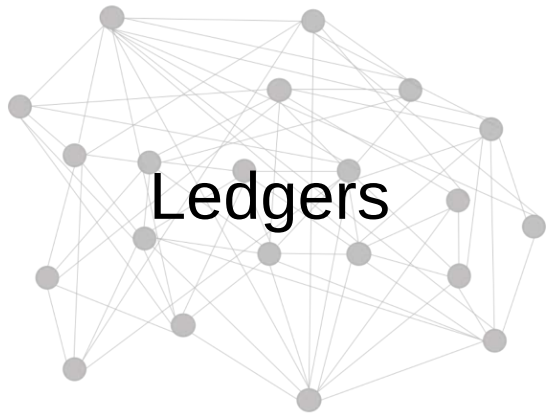
Nouraldin Jaber - Chris Wagner - Milind Kulkarni - Roopsha Samanta

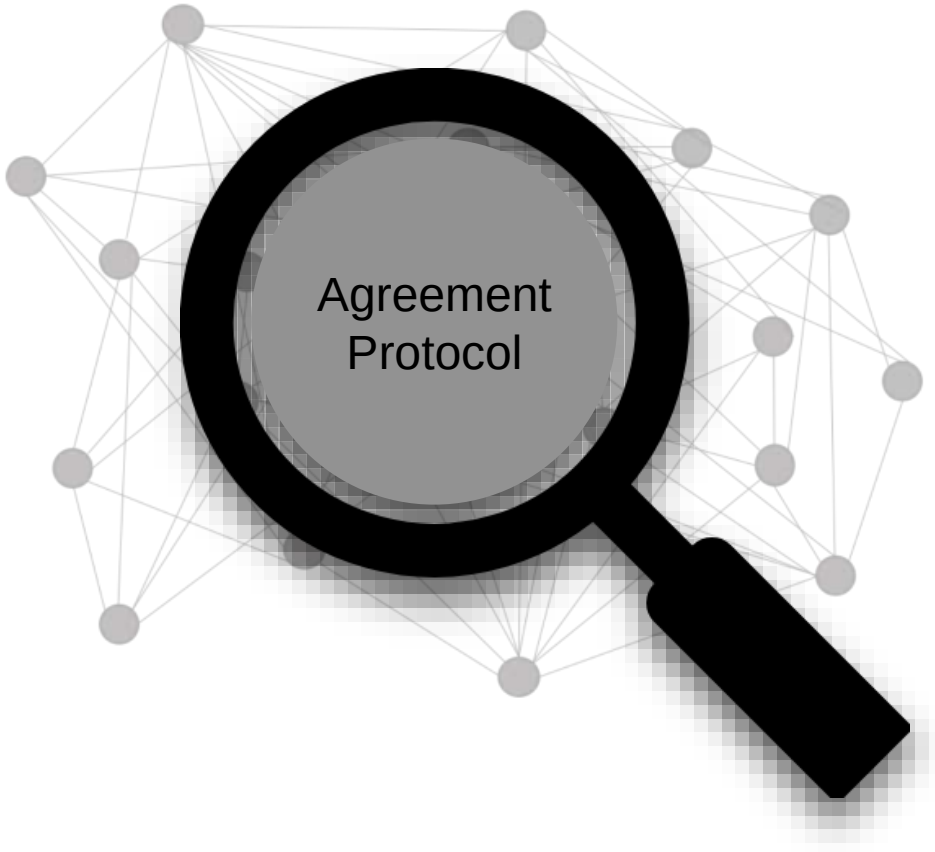
PURDUE UNIVERSITY

Distributed Services based on Agreement Protocols









CISPA
HELMHOLTZ CENTER FOR INFORMATION SECURITY

Chubby

RedisRaft

APACHE
ZooKeeper™

cassandra

CISPA
HELMHOLTZ CENTER FOR INFORMATION SECURITY

Chubby

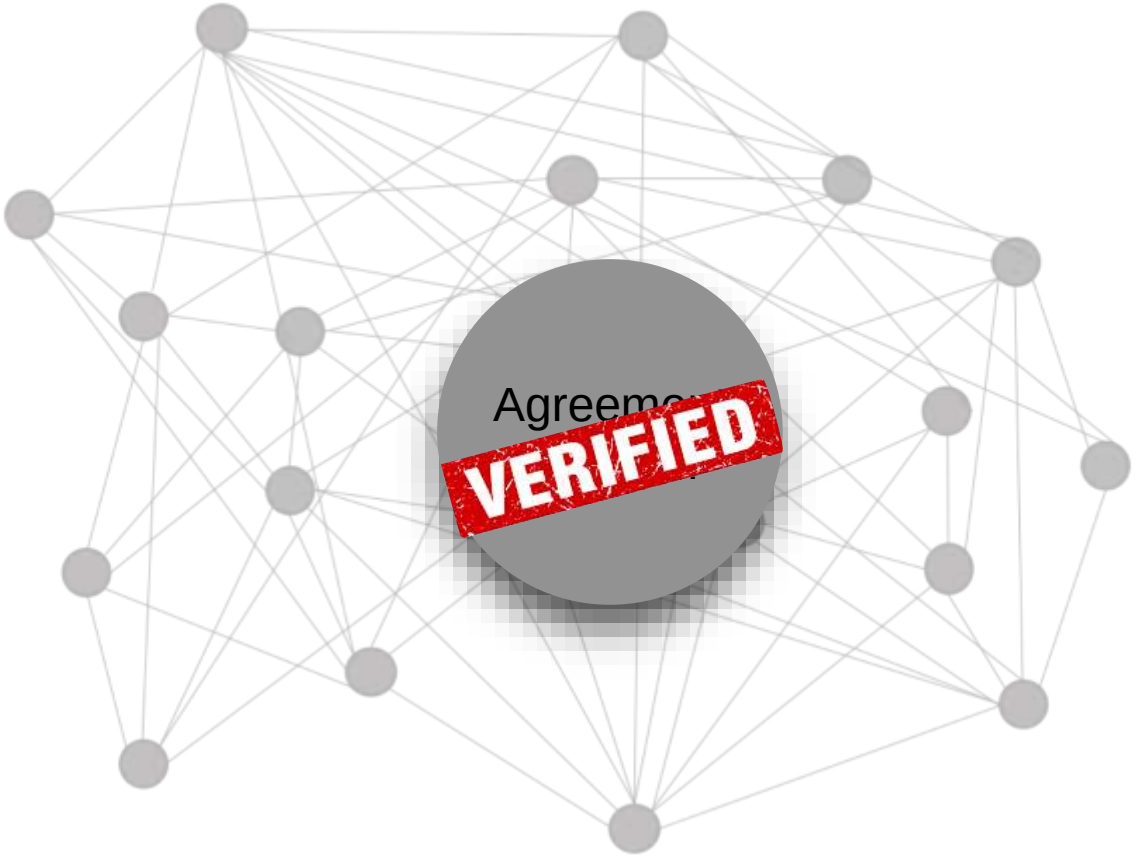
RedisRaft

Provably Correct Applications with Verified Building Blocks for Agreement

CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY

Modular verification:

Assume important properties of agreement primitive,
abstract from its implementation



Global Synchronization Protocols: our Contributions

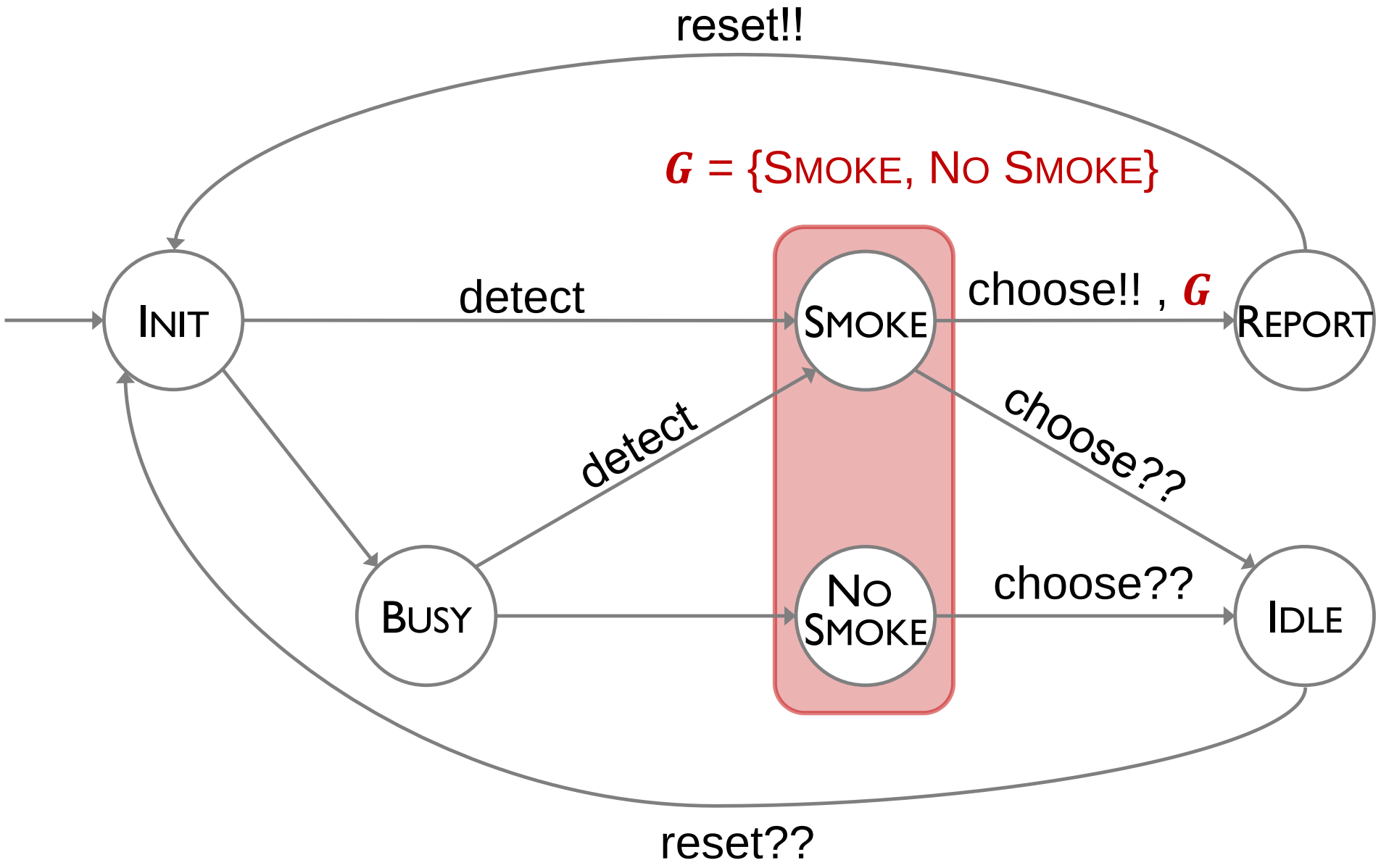
Parameterized verification	A fragment with decidable parameterized model checking problem (PMCP)
Cutoffs for the PMCP	In many cases, efficient parameterized reasoning is possible
Parameterized Synthesis	Cutoffs enable automatic design of correct systems, with additional benefits

Global Synchronization Protocols

a fragment with decidable PMCP

Global Synchronization Protocols

- Identical FSMs
- Interleaving semantics
- Broadcasts, Rendezvous
- Guarded commands



Capturing the Essence of Consensus

CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY

Consensus protocol

Participants

$$cons(\{1,2,4\},2) = \{\{1,2\},\{1,4\},\{2,4\}\}$$

Cardinality

Winners

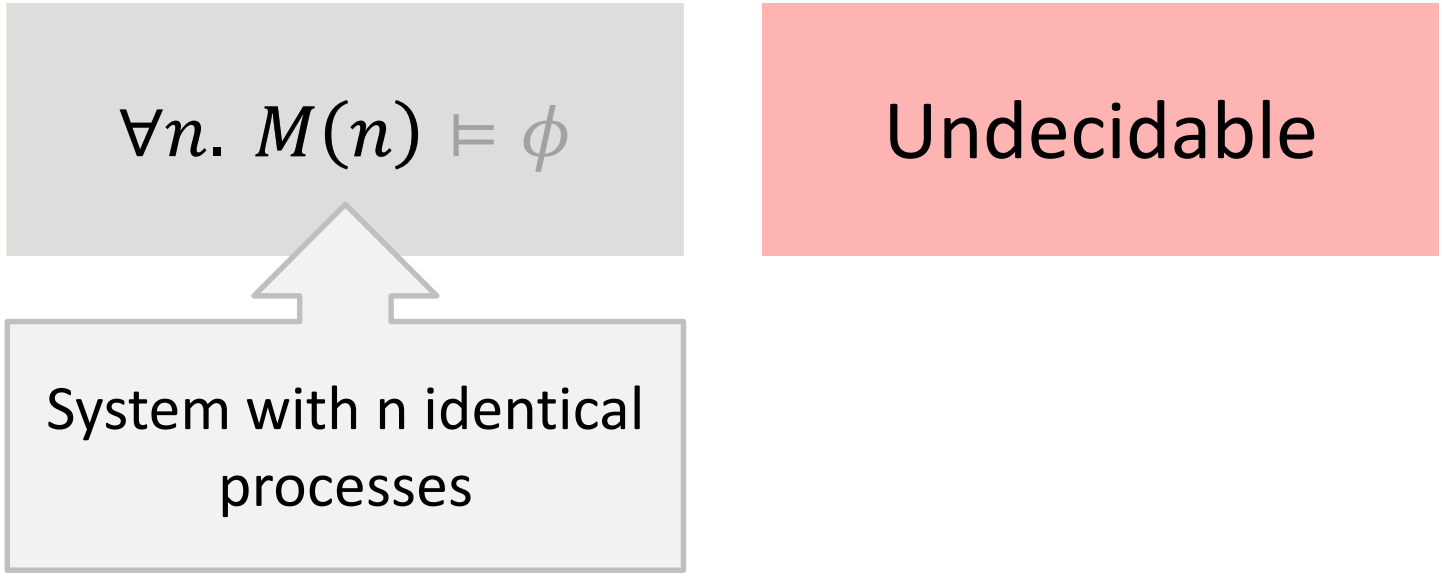
- Consistent Participants
- Consistent Winners

choose protocol

Counter abstraction, Id-based communication abstraction

Global Synchronization Protocol

The Parameterized Model Checking Problem (PMCP)



The PMCP for Broadcast Protocols and Guarded Protocols

Decidable fragments	Broadcast protocols	Guarded protocols
Communication primitives	Broadcasts	Global guards
Network topology	Clique	Clique
Specification	Safety	Safety + Liveness
	[Esparza et al. 1999]	[Emerson&Kahlon 2000]

Decidability of Parameterized Verification

CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY

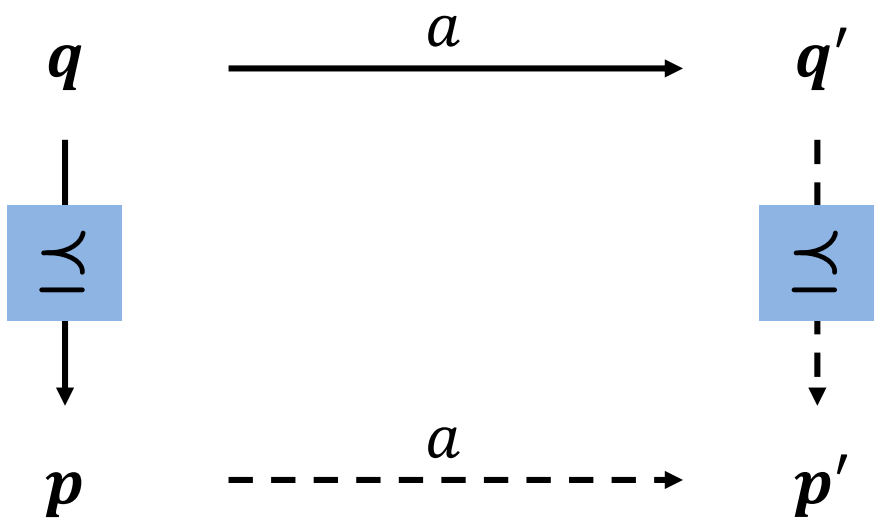
Key result [CAV 2020]:

PMCP is **decidable** for
well-behaved GSPs
w.r.t. safety properties.

Well-structured Transition Systems

$q \preceq p$ A well-quasi order (wqo) on global states

Compatibility:



wqo

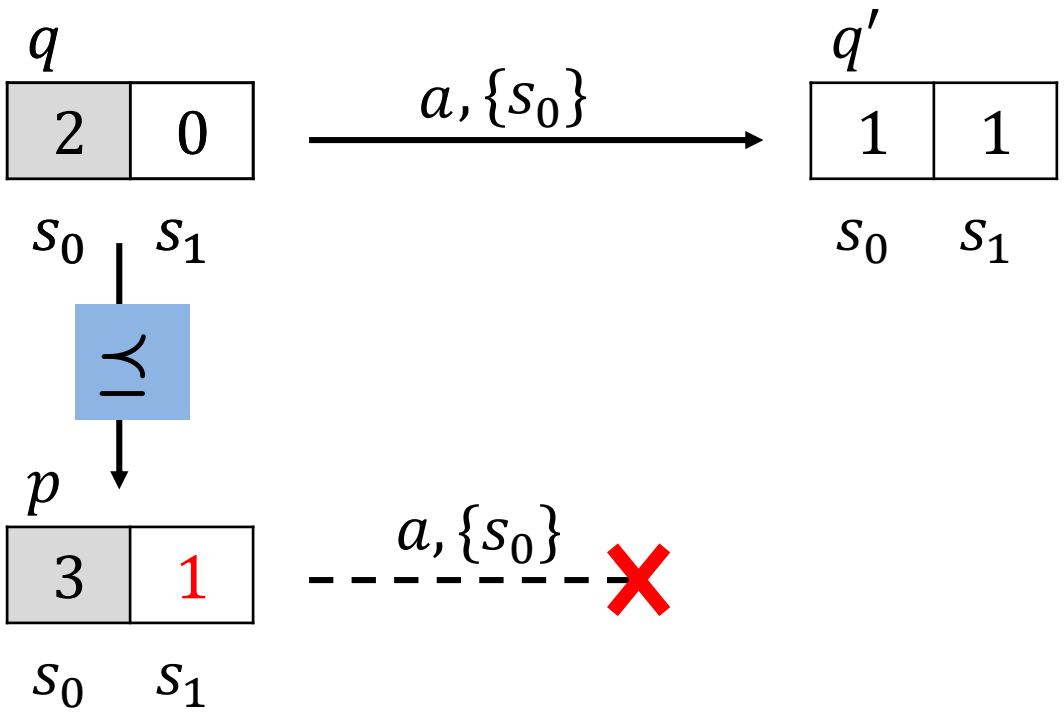
+ compatibility

+ computability of pred

= coverability decidable

A WQO for GSPs?

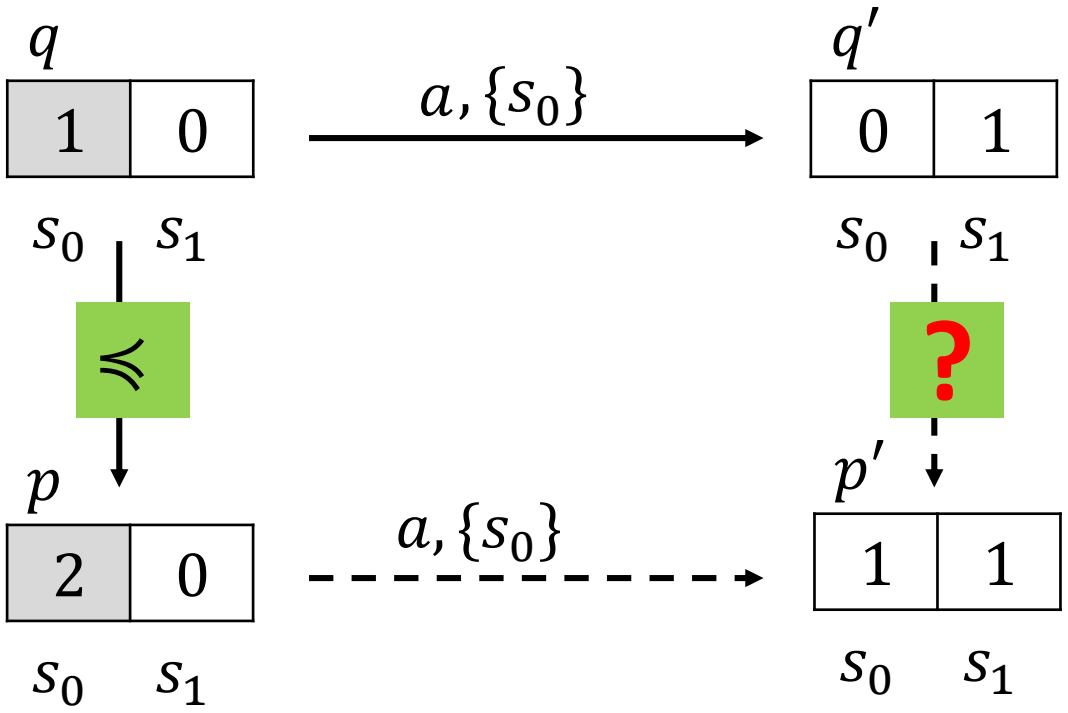
$q \preceq p$ At least as many processes in each local state



compatibility does not hold with respect to $\preceq$

A WQO for GSPs!

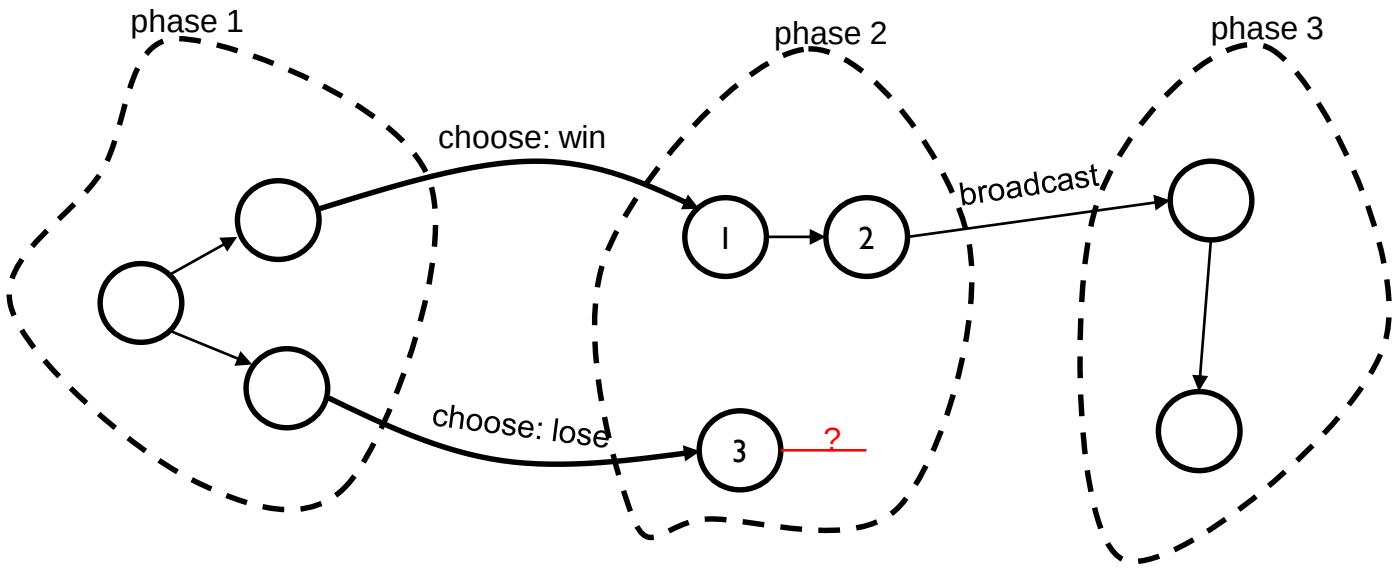
$q \preceq p$	At least as many processes	
$q \preccurlyeq p$	At least as many processes	Satisfaction of guards is unchanged



compatibility with respect to $\preccurlyeq$ **holds under well-behavedness conditions**

A Sufficient Condition for Well-behavedness [OOPSLA21]

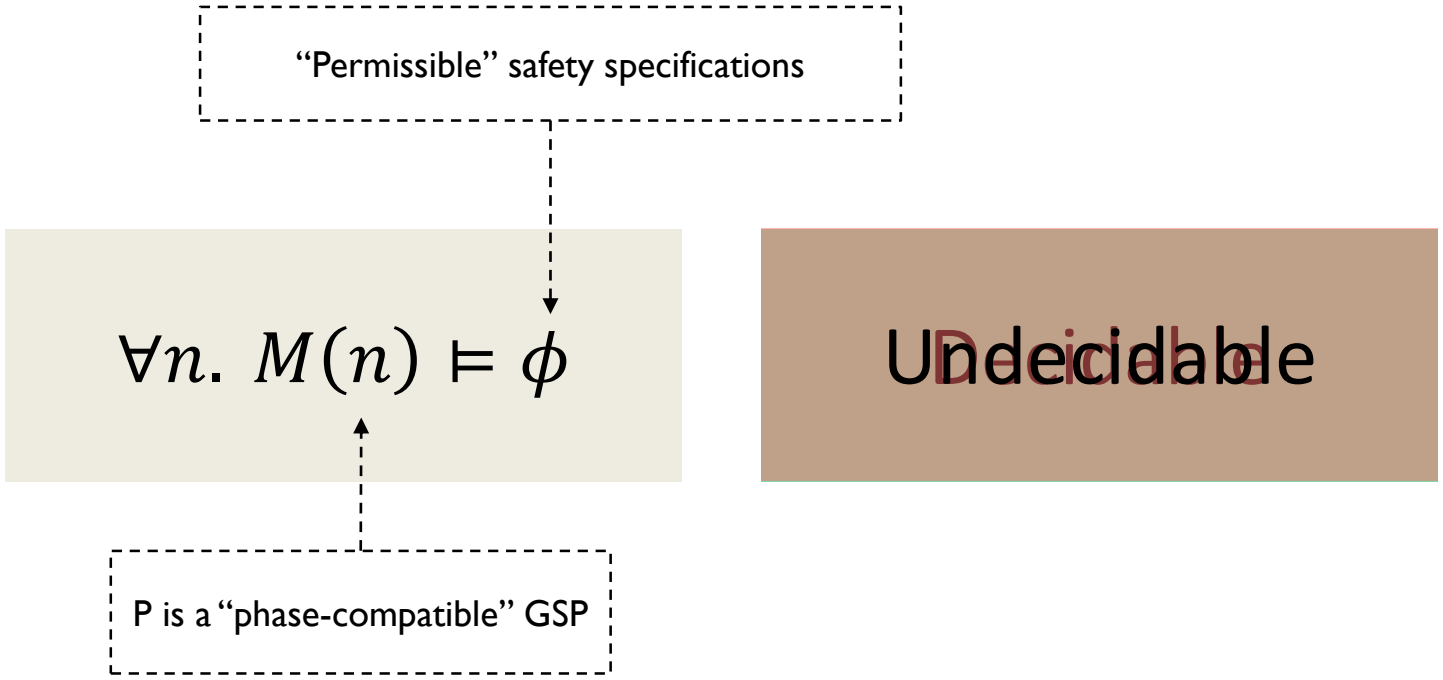
Phase-compatibility implies well-behavedness



phase-compatibility is easy to show for many applications

determined by **local analysis** of protocol, no composition of instances

Decidability of Parameterized Verification



Cutoffs for Parameterized Verification

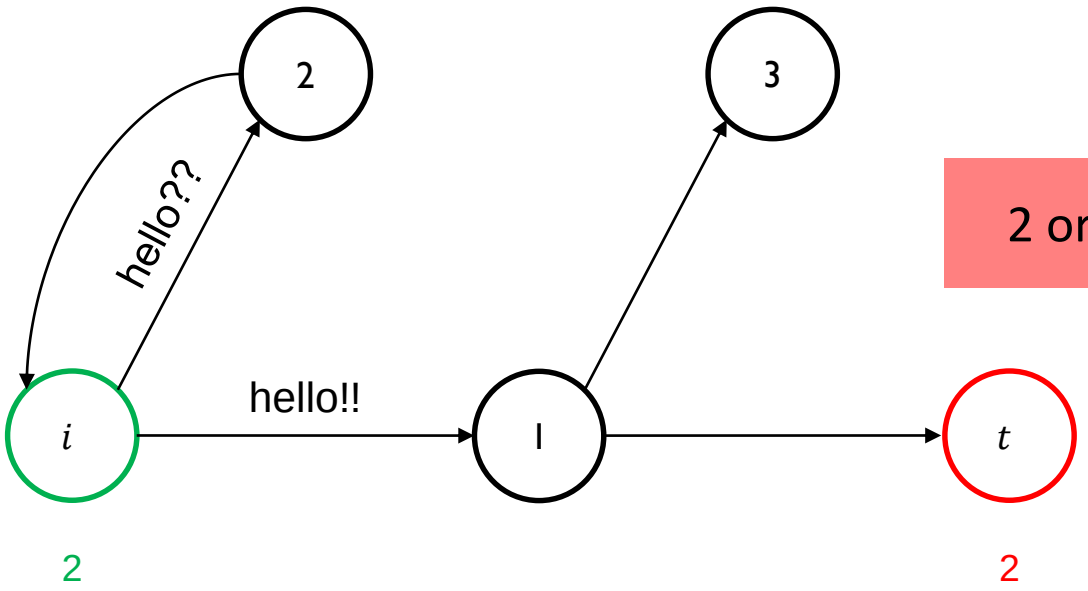
GSPs and other fragments

Cutoffs for Efficient Parameterized Verification

$$\forall n \geq c.\,(M(c) \vDash \phi \Leftrightarrow M(n) \vDash \phi)$$

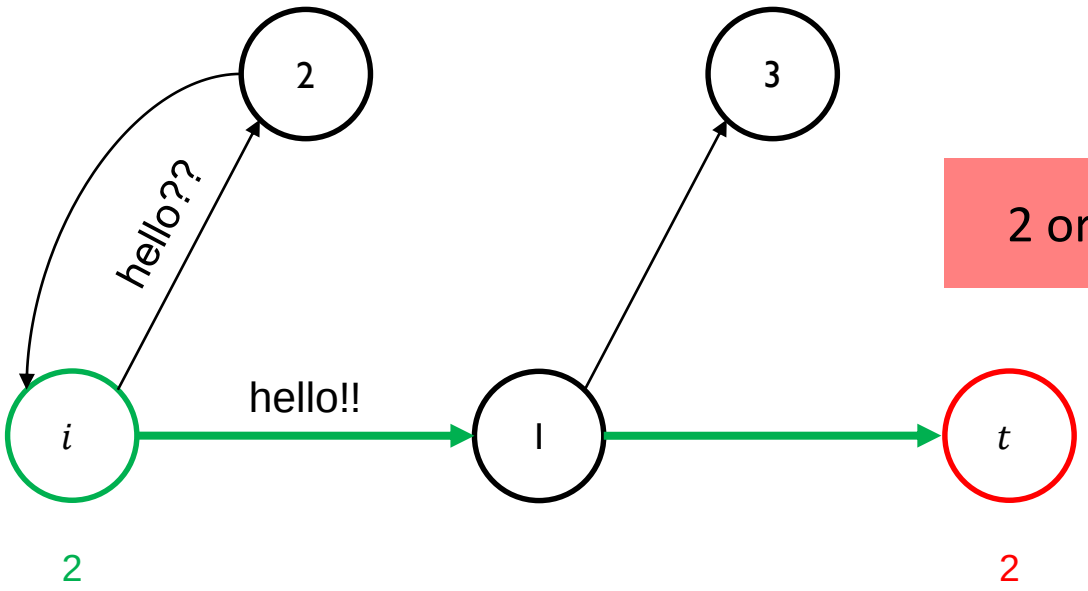
Minimal

Cutoffs for Efficient Parameterized Verification



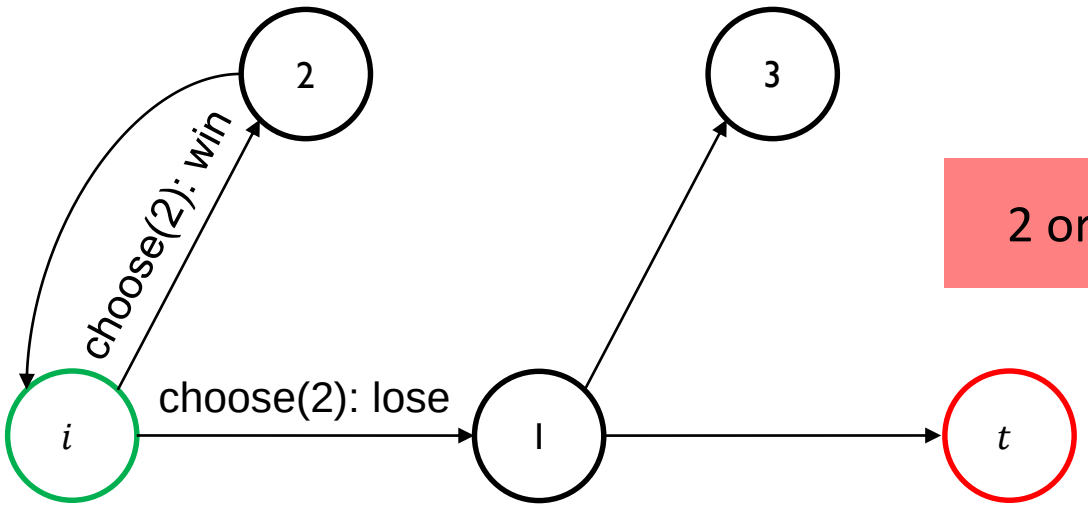
2 or more processes in t

Cutoffs for Efficient Parameterized Verification



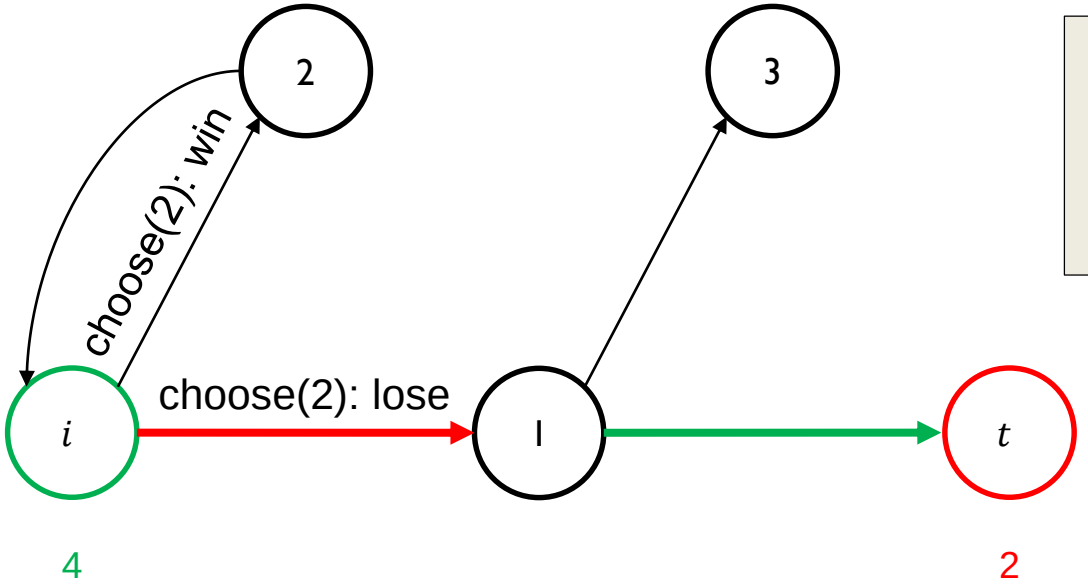
2 or more processes in t

Cutoffs for Efficient Parameterized Verification



2 or more processes in t

Cutoff-amenability conditions



determined by **local analysis** of protocol, no composition of instances

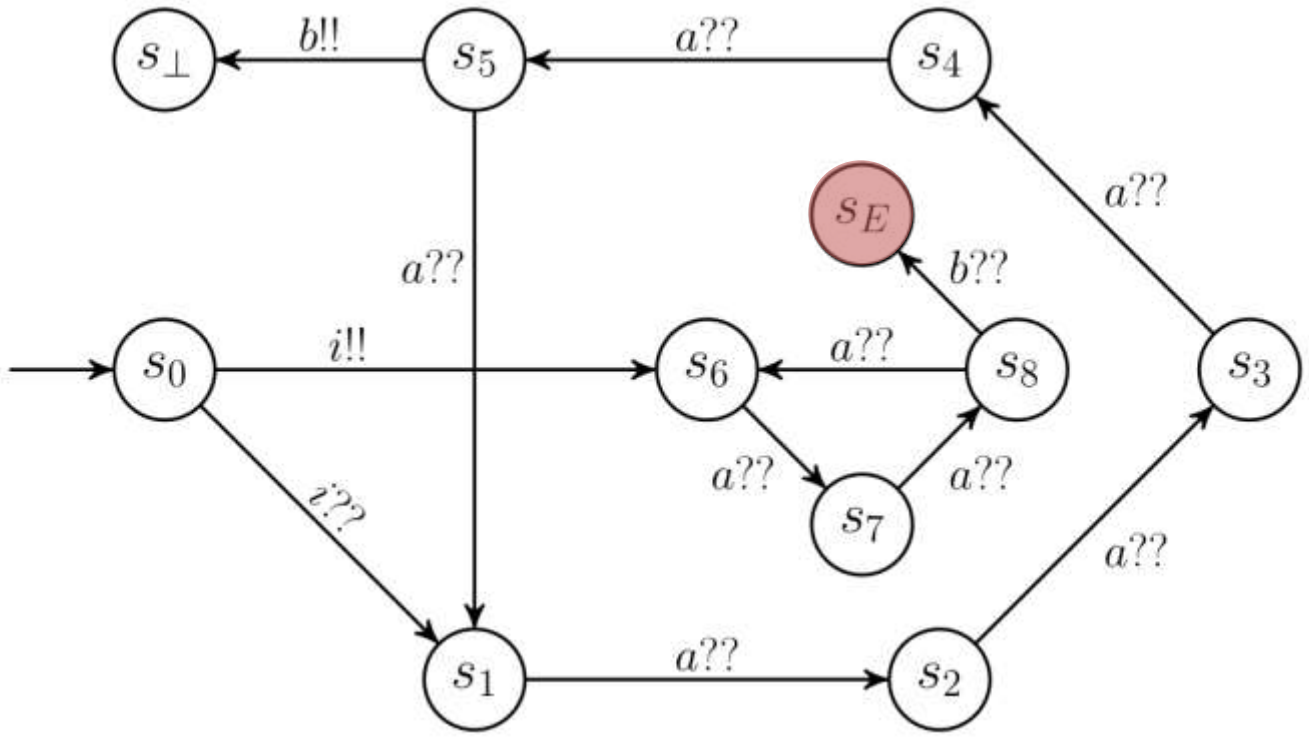
Systems where the minimum number of processes needed to trigger an m-process error is, in fact, m.

Do we have Small Cutoffs in General?

no, even for broadcast protocols we can get very large cutoffs

quadratic cutoff in examples of this form

towers of exponentials with more complex construction



states $s_1, \ldots, s_5$ have transitions with $a!!$ to sink state $s_\perp$

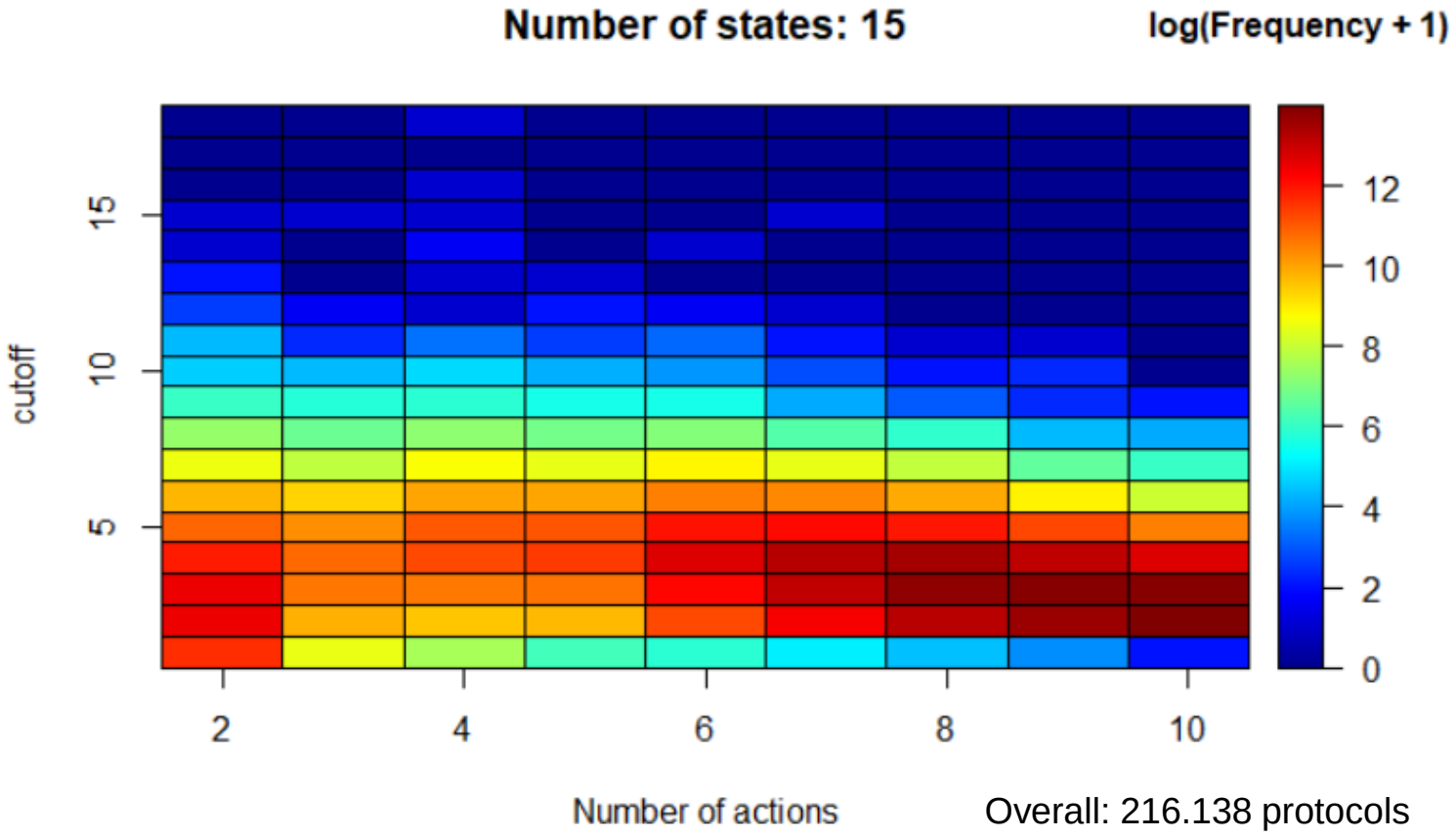
But: Experimental Evidence that Large Cutoffs are Rare [work in progress]

randomly generated broadcast protocols

ϕ = reachability of "last" state

determine individual cutoff w/ model checker

out of >20M random protocols, less than 0.01% have a cutoff greater than the number of local states



Cutoffs in our Example Applications

Benchmark	Phases	Cutoff
Distributed Store	2	3
Consortium*	9	3
Two-Object Tracker*	9	3
Distributed Robot Flocking	7	2
Distributed Lock Service	2	2
Distributed Sensor Network	3	3
Sensor Network with Reset	3	3
SATS Landing Protocol*	3	5
SATS Landing Protocol II*	5	5
Mobile Robotics Motion Planning	5	2
Mobile Robotics with Reset*	4	2
Distributed Register	1	2

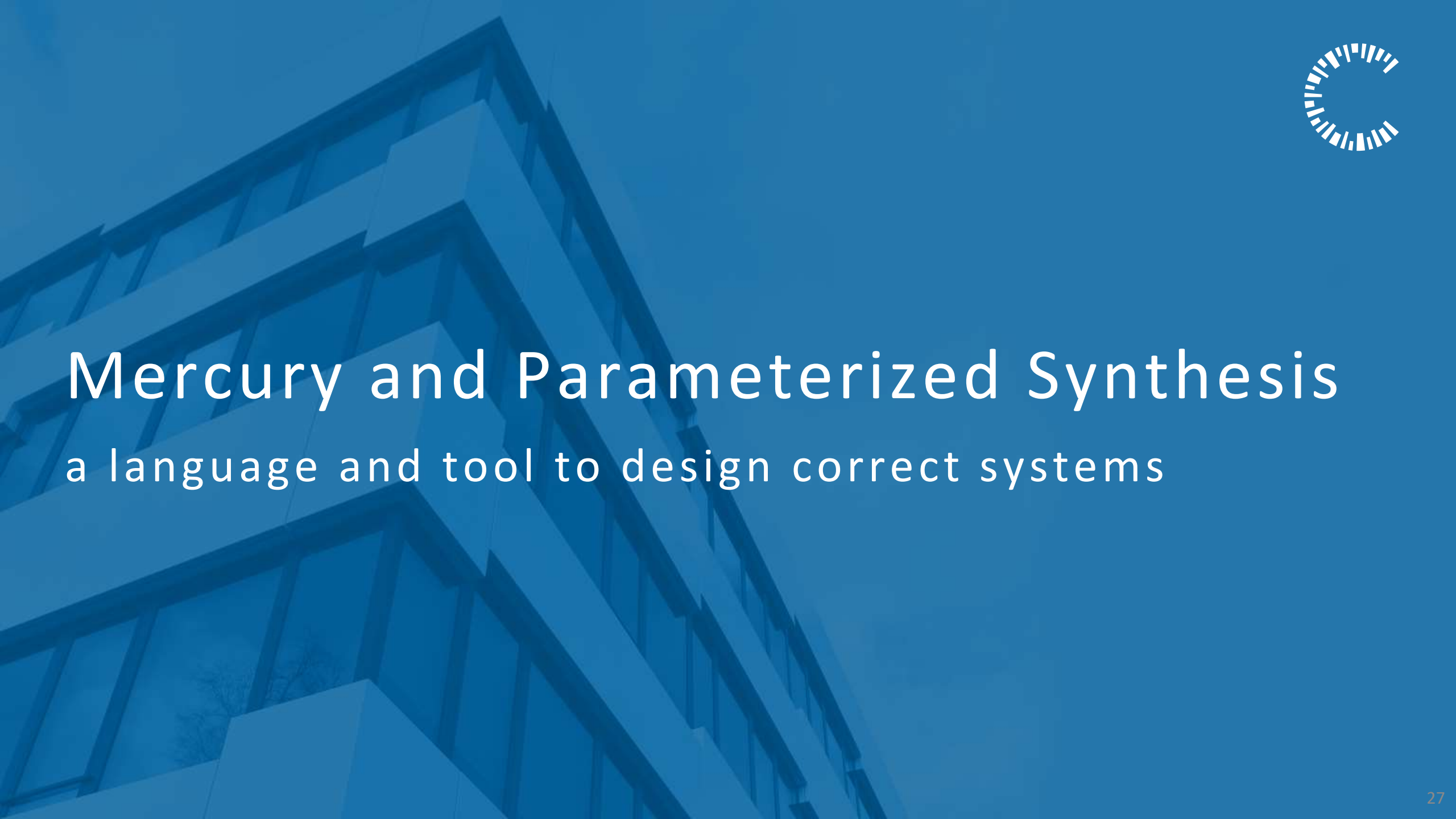
Small Cutoffs should be achievable for most Applications

very large cutoffs in theoretical worst case, and in artificial examples

very small cutoffs proved by hand for our applications

random examples have cutoffs $c \leq |P|$ in 99.99%

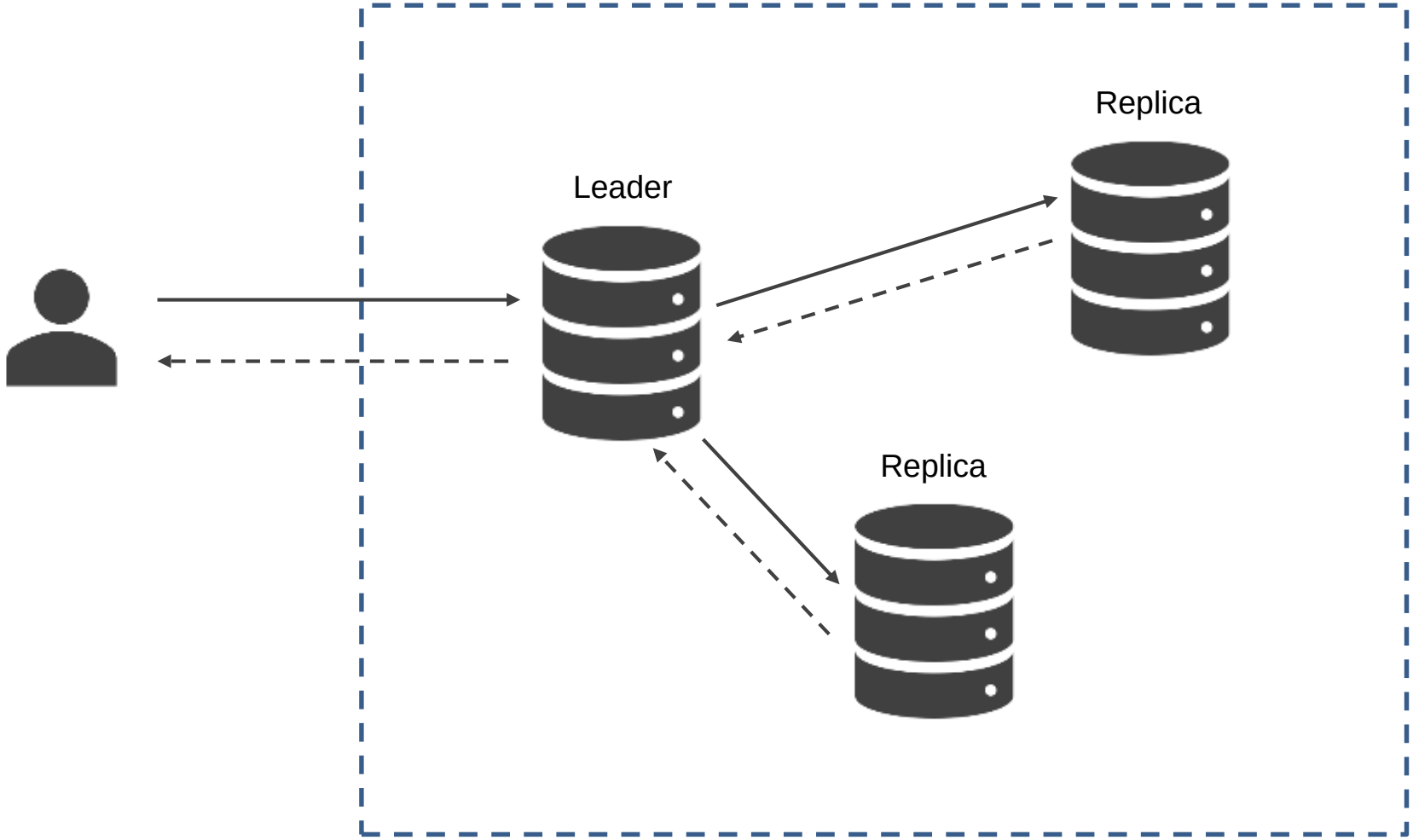
identification of classes of GSPs with small cutoffs a promising research direction



Mercury and Parameterized Synthesis

a language and tool to design correct systems

Distributed Store Example



Mercury

```
process DistributedStore
```

Define variables and actions

```
initial location Candidate
```

Elect a leader

```
location Leader
```

Serve client reads directly

```
location RepCmd
```

Agree on command to replicate

Execute the agreed-upon command

Confirm to client

```
location Replica
```

Agree on command to replicate

Execute the agreed-upon command

Elect a new leader when the leader is down

Mercury

```
process DistributedStore
  variables
    int[1,5] cmd
    int[1,2] stored

  actions
   env
    rz doCmd : int[1,5]
    rz ackCmd : int[1,5]
    rz ret : int[1,2]
    br LeaderDown : unit

  initial location Candidate
    on Partition<elect>(All,1)
      win: goto Leader
      lose: goto Replica

  location Leader
    on recv(doCmd) do
      cmd ≔ doCmd.payld

      if(cmd <= 2 && stored = cmd)
        goto Leader
      else if(cmd = 3)
        sendrz(ret[stored],doCmd.sID)
      else
        goto RepCmd
```

```
location RepCmd
  on Consensus<vc>(All,1,cmd) do
    cmd ≔ vcCmd.decVar[1]

    if(cmd <= 2) /*set*/
      stored ≔ cmd
    else if(cmd = 4) /*inc*/
      stored ≔ stored + 1
    else /*dec*/
      stored ≔ stored - 1
    sendrz(ackCmd[cmd],doCmd.sID)
    goto Leader

location Replica
  on Consensus<vc>(All,1,_) do
    cmd ≔ vcCmd.decVar[1]

    if(cmd <= 2) /*set*/
      stored ≔ cmd
    else if(cmd = 4) /*inc*/
      stored ≔ stored + 1
    else /*dec*/
      stored ≔ stored - 1

  on recv(LeaderDown) do
    goto Candidate
```

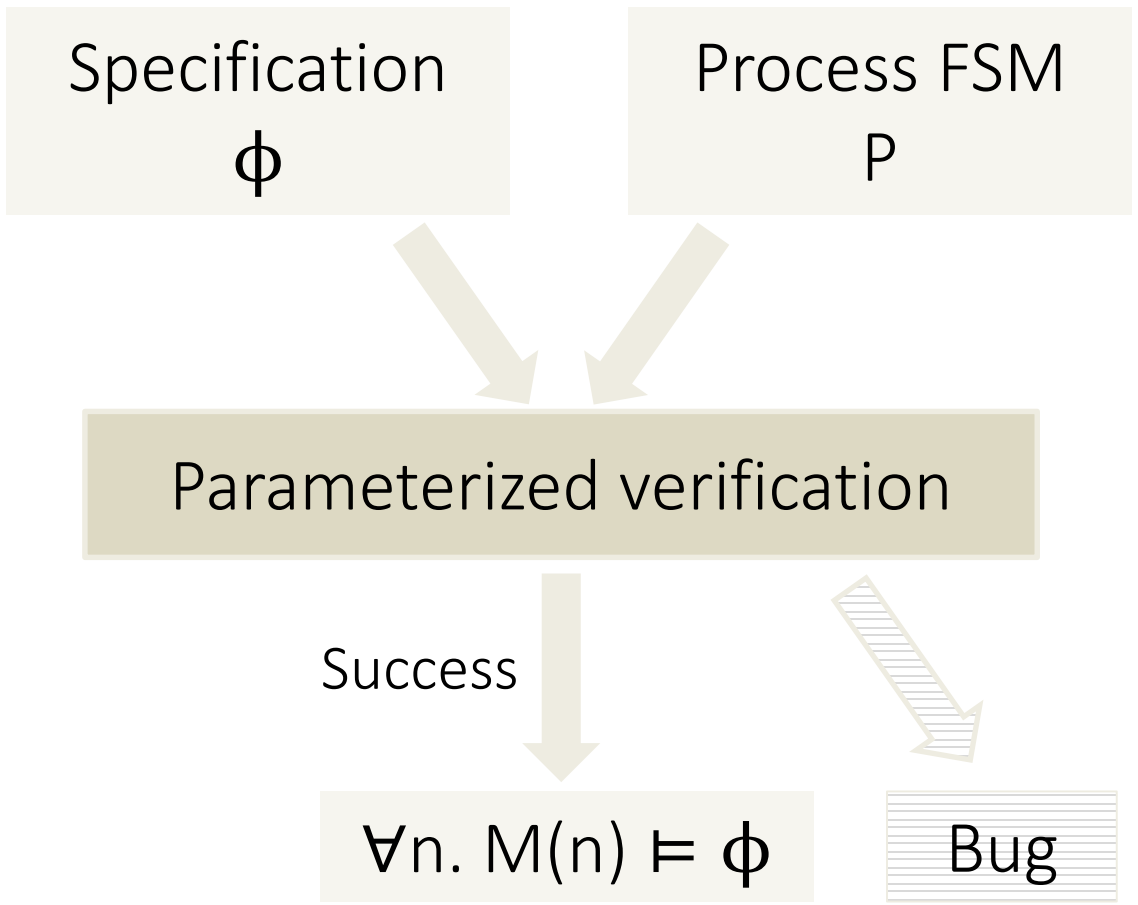
Mercury

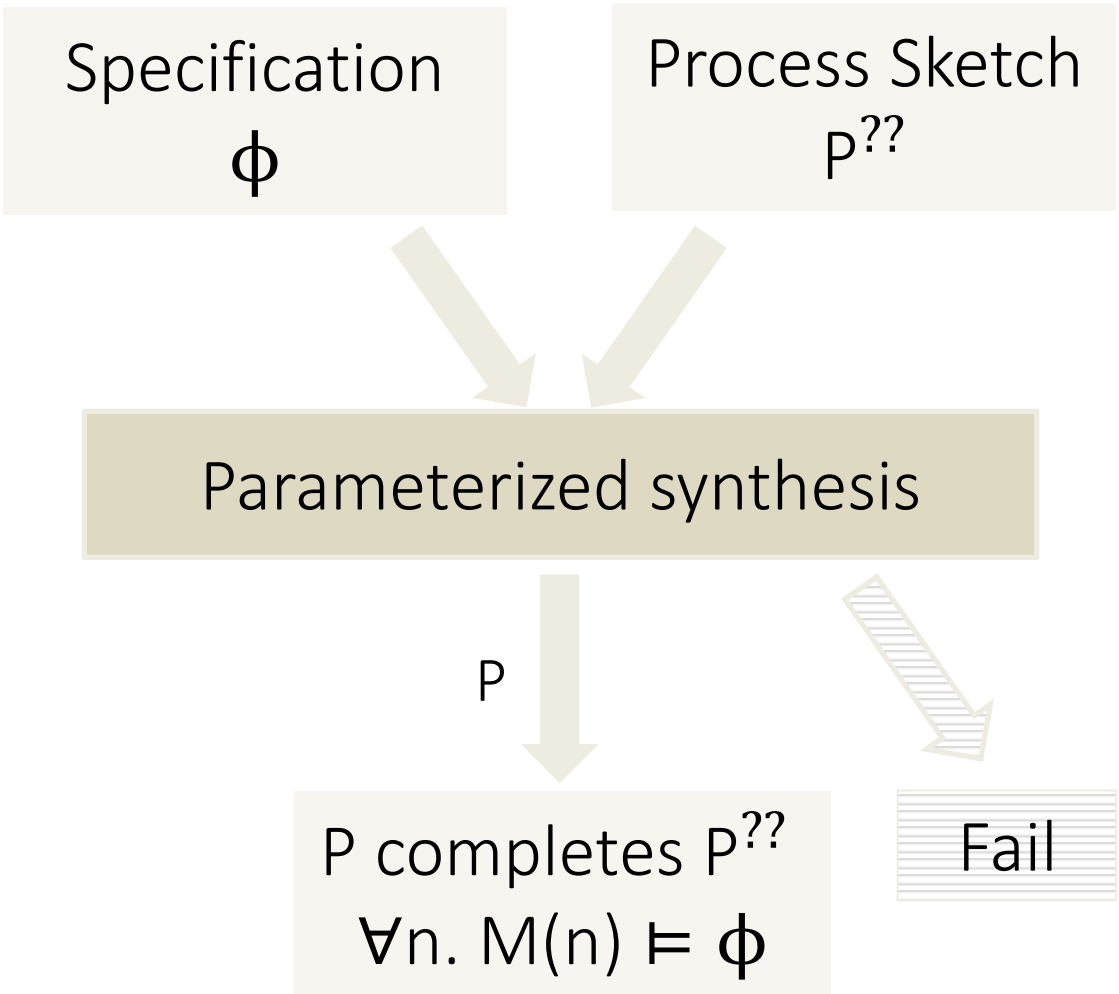
- Variables
- Actions
  - Broadcasts
  - Rendezvous
- Locations
- Event handlers
  - Receive
  - Send
  - Internal
  - Partition
  - Consensus

```
process DistributedStore
  variables
    int[1,5] cmd
    int[1,2] stored

  actions
   env
    rz doCmd : int[1,5]
    rz ackCmd : int[1,5]
    rz ret : int[1,2]
    br LeaderDown : unit

  initial location Candidate
    on Partition<elect>(All,1)
      win: goto Leader
      lose: goto Replica

  location Leader
    on recv(doCmd) do
      cmd ≔ doCmd.payld

      if(cmd <= 2 && stored = cmd)
        goto Leader
      else if(cmd = 3)
        sendrz(ret[stored],doCmd.sID)
      else
        goto RepCmd
```

```
location RepCmd
  on Consensus<vc>(All,1,cmd) do
    cmd ≔ vcCmd.decVar[1]

    if(cmd <= 2) /*set*/
      stored ≔ cmd
    else if(cmd = 4) /*inc*/
      stored ≔ stored + 1
    else /*dec*/
      stored ≔ stored - 1
    sendrz(ackCmd[cmd],doCmd.sID)
    goto Leader

location Replica
  on Consensus<vc>(All,1,_) do
    cmd ≔ vcCmd.decVar[1]

    if(cmd <= 2) /*set*/
      stored ≔ cmd
    else if(cmd = 4) /*inc*/
      stored ≔ stored + 1
    else /*dec*/
      stored ≔ stored - 1

  on recv(LeaderDown) do
    goto Candidate
```

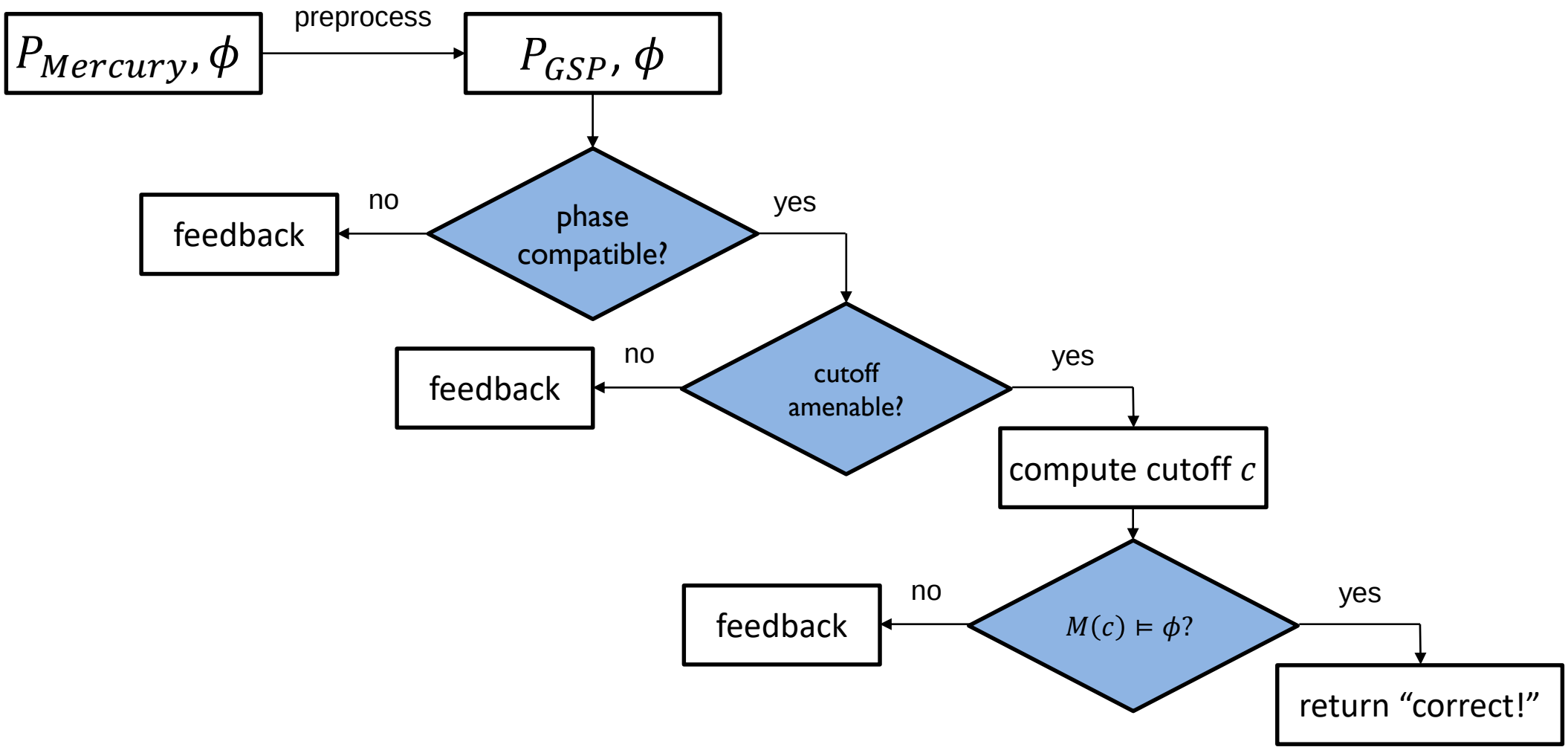
Parameterized Verification and Synthesis





$$M(n) = P_1 \| \cdots \| P_n$$

The Quicksilver Verification Tool

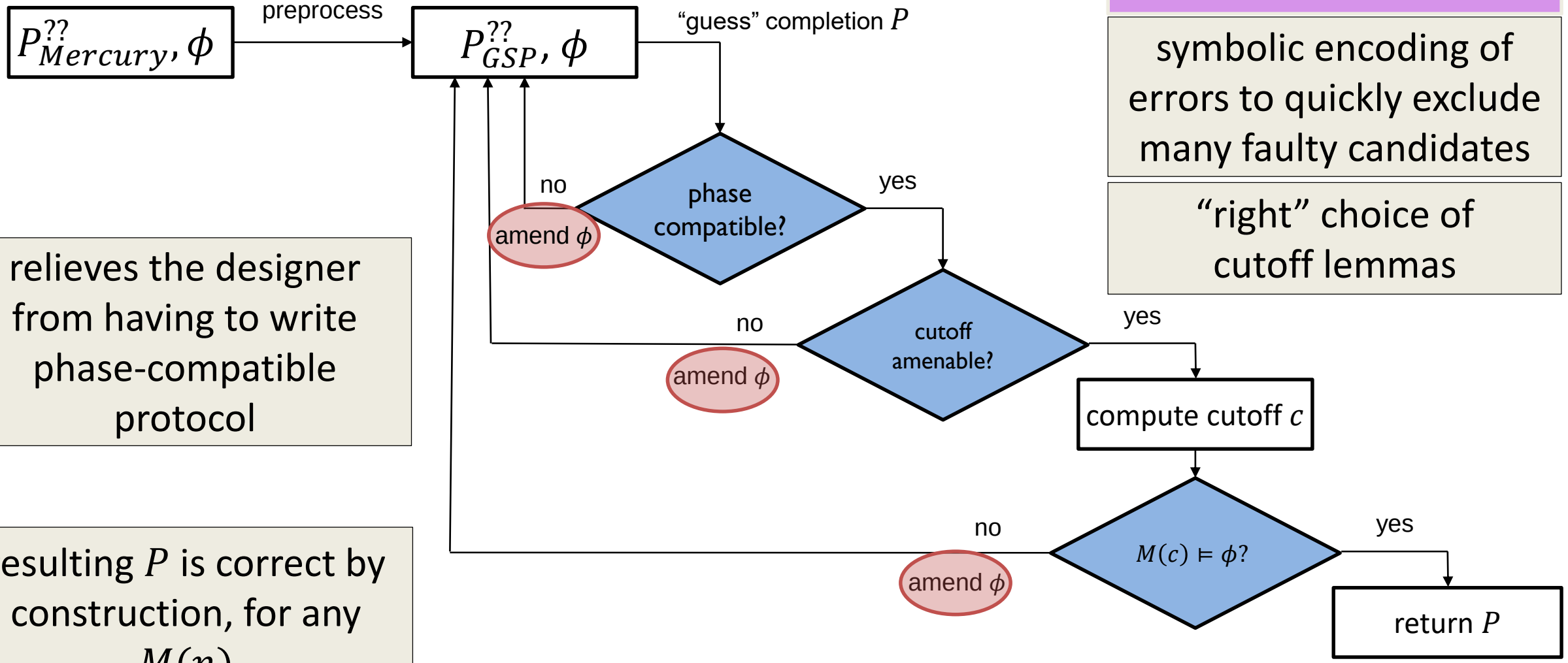


Quicksilver: Benchmarks (again)

CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY

Benchmark	LoC	Phases	Cutoff	Time(s)
Distributed Store	64	2	3	45.079 ± 0.621
Consortium*	58	9	3	6.953 ± 0.022
Two-Object Tracker*	69	9	3	0.641 ± 0.006
Distributed Robot Flocking	78	7	2	0.105 ± 0.002
Distributed Lock Service	38	2	2	0.059 ± 0.002
Distributed Sensor Network	55	3	3	1.041 ± 0.003
Sensor Network with Reset	63	3	3	1.662 ± 0.012
SATS Landing Protocol*	90	3	5	638.393 ± 0.872
SATS Landing Protocol II*	99	5	5	736.417 ± 3.659
Mobile Robotics Motion Planning	71	5	2	0.114 ± 0.004
Mobile Robotics with Reset*	83	4	2	0.166 ± 0.003
Distributed Register	32	1	2	0.329 ± 0.006

Quicksilver: Extension to Parameterized Synthesis [work in progress]



relieves the designer from having to write phase-compatible protocol

resulting P is correct by construction, for any $M(n)$

main challenges:

symbolic encoding of errors to quickly exclude many faulty candidates

"right" choice of cutoff lemmas

Summary: Parameterized Verification of Global Synchronization Protocols

CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY

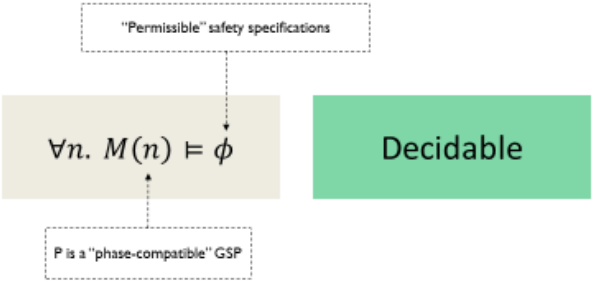
Provably Correct Applications with Verified Building Blocks for Agreement

Modular verification:

Assume important properties of agreement primitive, abstract from its implementation

4

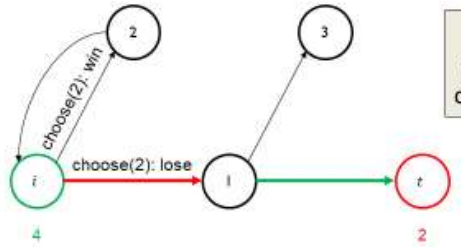
Decidability of Parameterized Verification



11

Cutoffs for Efficient Parameterized Verification

Cutoff-amenability conditions

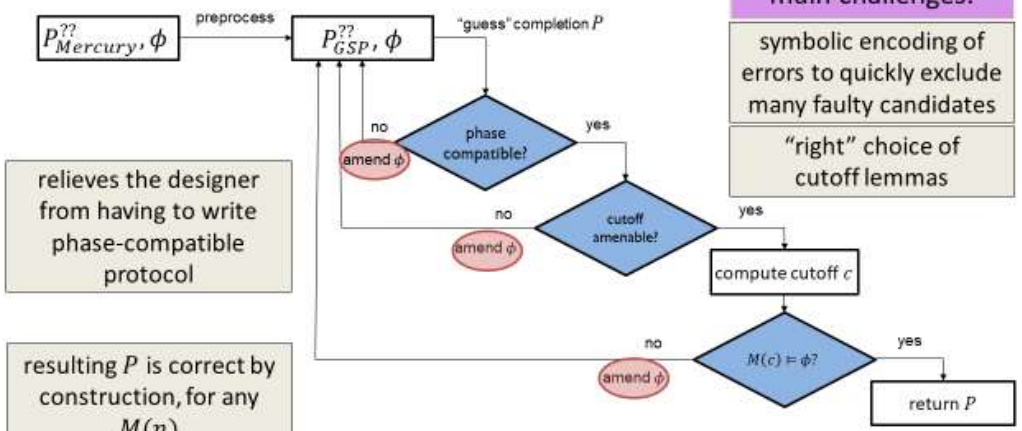


determined by **local analysis** of protocol, no composition of instances

Systems where the minimum number of processes needed to trigger an m-process error is, in fact, m.

22

Quicksilver: Extension to Parameterized Synthesis [work in progress]



main challenges:

symbolic encoding of errors to quickly exclude many faulty candidates

"right" choice of cutoff lemmas

relieves the designer from having to write phase-compatible protocol

resulting P is correct by construction, for any $M(n)$

35