Synthesis of Distributed Algorithms with Parameterized Threshold Guards

Igor Konnov

Marijana Lazić Josef Widder Roderick Bloem

OPODIS'17, December 2017

logics LOGICAL METHODS IN COMPUTER SCIENCE

RiSE Rigorous Systems Engineering

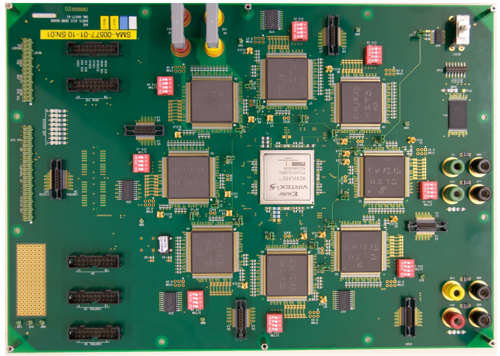
WWTF VIENNA SCIENCE AND TECHNOLOGY FUND

Verifying fault-tolerant systems

safety critical systems: cars, planes, etc.

- rare but dangerous faults

- 3 to 7 processes



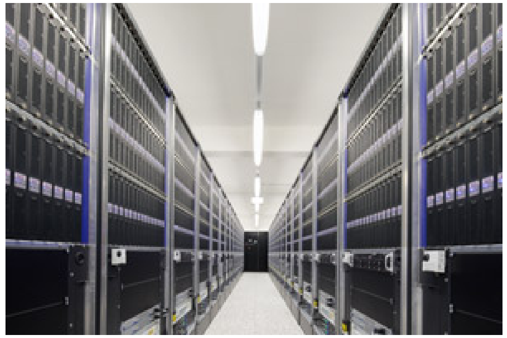
---

finite-state model checking

datacenters: thousands of computers

- faults happen every day

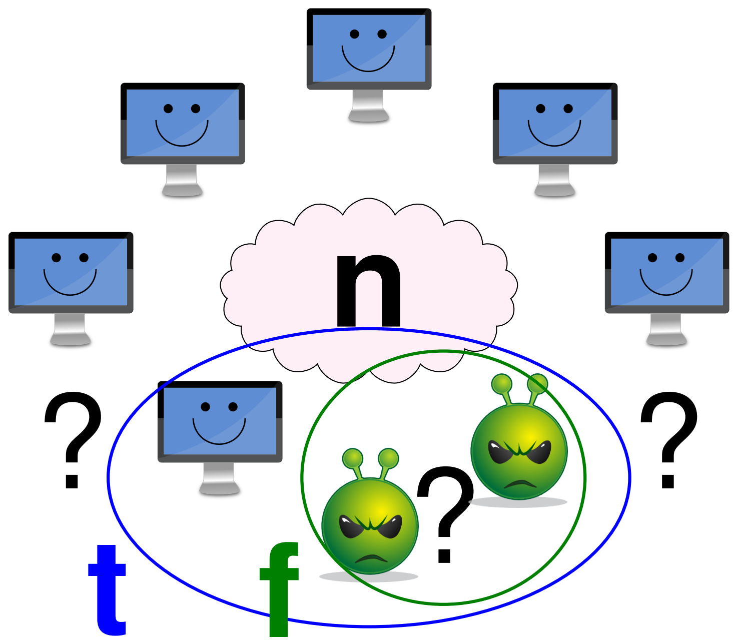
- 100–10,000 processes



---

parameterized model checking

Fault-tolerant distributed algorithms



n processes communicate by sending messages

f processes are faulty (unknown)

t is an upper bound on ***f*** (known)

resilience condition on n, t, and f, e.g., $n > 3t \wedge t \geq f \geq 0$

Reliable broadcast service (informally)

one process broadcasts a message **bcast**

correctness: if all correct processes received **bcast**, then some correct process eventually accepts **bcast** — $111\ldots1$

relay: if a correct process accepts **bcast**, then all correct processes eventually accept **bcast** — $011\ldots1$

unforgeability: if no correct process received **bcast**, then no correct process ever accepts **bcast** — $000\ldots0$

fairness: every sent message is eventually received

Reliable broadcast by Srikanth & Toueg 87

```
local myval_i ∈ {0,1}          - did process i receive bcast?

while true do
 if myval_i = 1 and not sent ECHO before
 then send ECHO to all

 if received ECHO from at least t + 1    distinct processes
    and not sent ECHO before
 then send ECHO to all

 if received ECHO from at least n - t    distinct processes
 then accept
od
```

resilience: of $n > 3t$ processes, $f \leq t$ processes are Byzantine

Reliable broadcast by Srikanth & Toueg 87

```
local myval_i ∈ {0,1}          - did proc

while true do
  if myval_i = 1 and not sent ECHO before
  then send ECHO to all

  if received ECHO from at least t + 1    distinct processes
      and not sent ECHO before
  then send ECHO to all

  if received ECHO from at least n - t
  then accept
od
```

a threshold guard

a threshold guard

resilience: of $n > 3t$ processes, $f \leq t$ processes are Byzantine

More threshold guards...

Reliable broadcast	$x \geq t + 1$ $x \geq n - t$	[Srikanth, Toueg'86]
Hybrid broadcast	$x \geq t_b + 1$ $x \geq n - t_b - t_c$	[Widder, Schmid'07]
Byzantine agreement	$x \geq \lceil \frac{n}{2} \rceil + 1$	[Bracha, Toueg'85]
Non-blocking atomic commitment	$x \geq n$	[Raynal'97], [Guerraoui'01]
Condition-based consensus	$x \geq n - t$ $x \geq \lceil \frac{n}{2} \rceil + 1$	[Mostéfaoui, Mourgaya, Parvedy, Raynal'03]
Consensus in one communication step	$x \geq n - t$ $x \geq n - 2t$	[Brasileiro, Greve, Mostéfaoui, Raynal'03]
Byzantine one-step consensus	$x \geq \lceil \frac{n+3t}{2} \rceil + 1$	[Song, van Renesse'08]

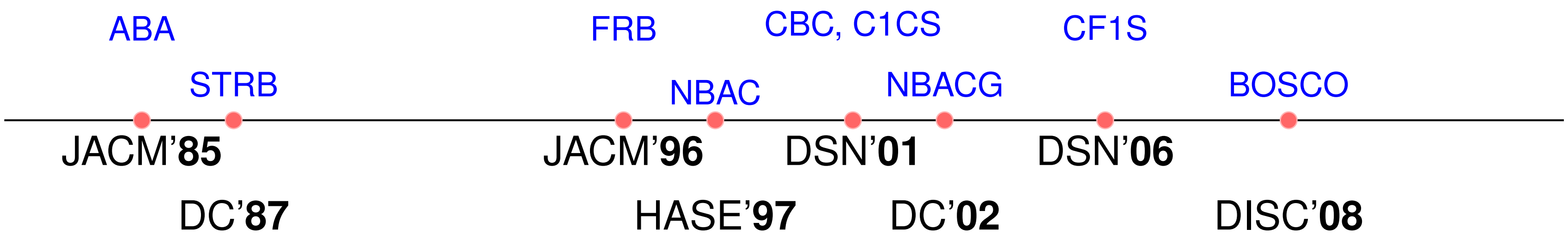
In general, there is a resilience condition, e.g., $n > 3t$, $n > 7t$

Byzantine model checker

[forsyte.at/software/bymc]

(source code, benchmarks, virtual machines, etc.)

10 parameterized fault-tolerant distributed algorithms:



[CAV'15] & [POPL'17]

From verification to synthesis

Different threshold guards for one sketch

```
local myval_i ∈ {0, 1}          - did process i receive bcast?

while true do
  if myval_i = 1 and not sent ECHO before
  then send ECHO to all

  if received ECHO from at least t + 1    distinct processes
     and not sent ECHO before
  then send ECHO to all

  if received ECHO from at least n - t    distinct processes
  then accept
od
```

✓

resilience: of $n > 3t$ processes, $f \leq t$ processes are Byzantine

Different threshold guards for one sketch

```
local myval_i ∈ {0, 1}            - did process i receive bcast?

while true do
 if  myval_i = 1 and not sent ECHO before
 then send ECHO to all

 if  received ECHO from at least t + 1      distinct processes
     and not sent ECHO before
 then send ECHO to all

 if  received ECHO from at least n - t      distinct processes
 then accept
od
```

- $t+1$ → $t+1$ ✓
- $n-t$ → $2t+1$ ✓

resilience: of $n > 3t$ processes, $f \le t$ processes are Byzantine

Different threshold guards for one sketch

local $myval_i \in \{0, 1\}$ `- did process` i `receive` **bcast**?

while `true` **do**
 if $myval_i = 1$ **and not** `sent ECHO before`
 then `send ECHO to all`

 if `received ECHO from at least` `t + 1` **distinct** `processes`
   and not `sent ECHO before`
 then `send ECHO to all`

 if `received ECHO from at least` `n - t` **distinct** `processes`
 then `accept`
od

$t + 1$ | $n - 2t$

$2t + 1$ | $n - t$

✓ ✓ ✓

resilience: of $n > 3t$ processes, $f \leq t$ processes are Byzantine

Different threshold guards for one sketch

local $myval_i \in \{0,1\}$ - did process i receive **bcast**?

while true **do**
  if $myval_i = 1$ **and not** sent ECHO before
  then send ECHO to all

  if received ECHO from at least t + 1 distinct processes
    and not sent ECHO before
  then send ECHO to all

  if received ECHO from at least n - t distinct processes
  then accept
od

$t+1$	$t+1$	$n-2t$	$n-2t$
$n-t$	$2t+1$	$n-t$	$2t+1$
✓	✓	✓	✗

resilience: of $n > 3t$ processes, $f \le t$ processes are Byzantine

Find thresholds automatically?

```
local myval_i ∈ {0,1}          - did process i receive bcast?

while true do
 if myval_i = 1 and not sent ECHO before
 then send ECHO to all

 if received ECHO from at least [ ? ] distinct processes
    and not sent ECHO before
 then send ECHO to all

 if received ECHO from at least [ ? ] distinct processes
 then accept
od
```

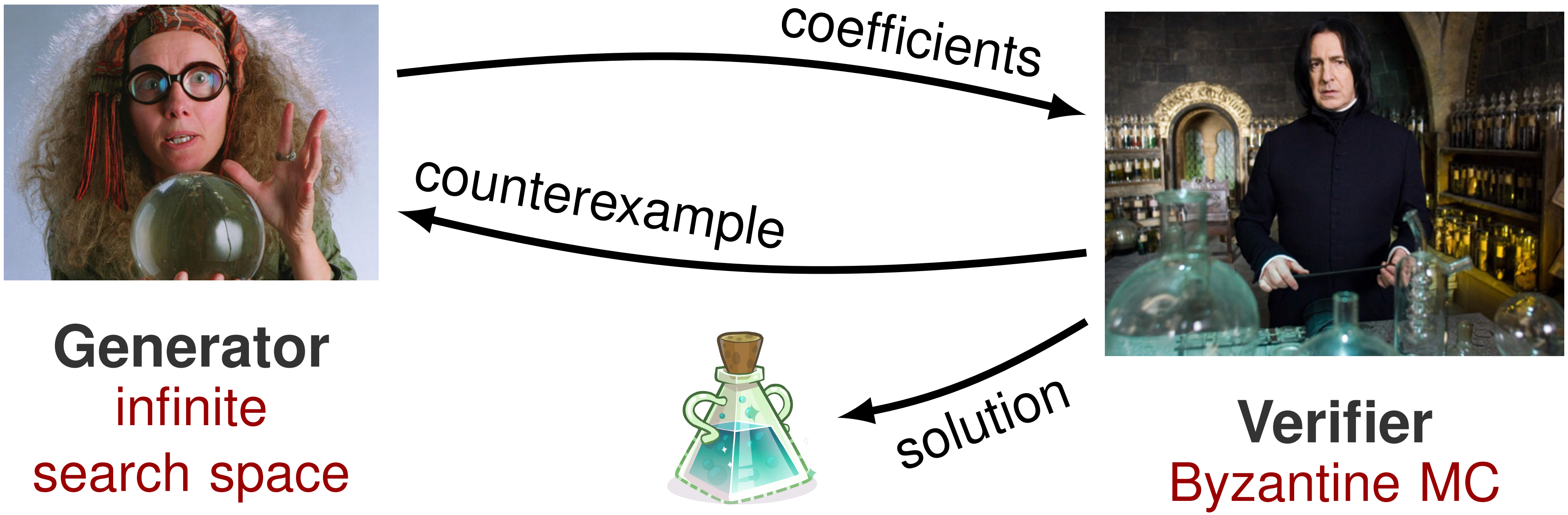
$?_1 \cdot n + ?_2 \cdot t + ?_3$

$?_4 \cdot n + ?_5 \cdot t + ?_6$

resilience: of $n > 3t$ processes, $f \leq t$ processes are Byzantine

Synthesis loop

Find $\textbf{?}_1, \ldots, \textbf{?}_k \in \mathbb{Q}$



Synthesis problem

Find a distributed algorithm that satisfies spec φ

for all parameter values n, t, and f that satisfy resilience condition

---

Input:

sketch algorithm,	`if received` $?_1 \cdot n + ?_2 \cdot t + ?_3$ `echoes` then send echo to all
specification,	e.g., unforgeability, correctness & relay
resilience condition,	e.g., $n > 3t$, $t \geq f \geq 0$

Find (if exist): $a_1, \ldots, a_k \in \mathbb{Q}$ for $?_1, \ldots, ?_k$

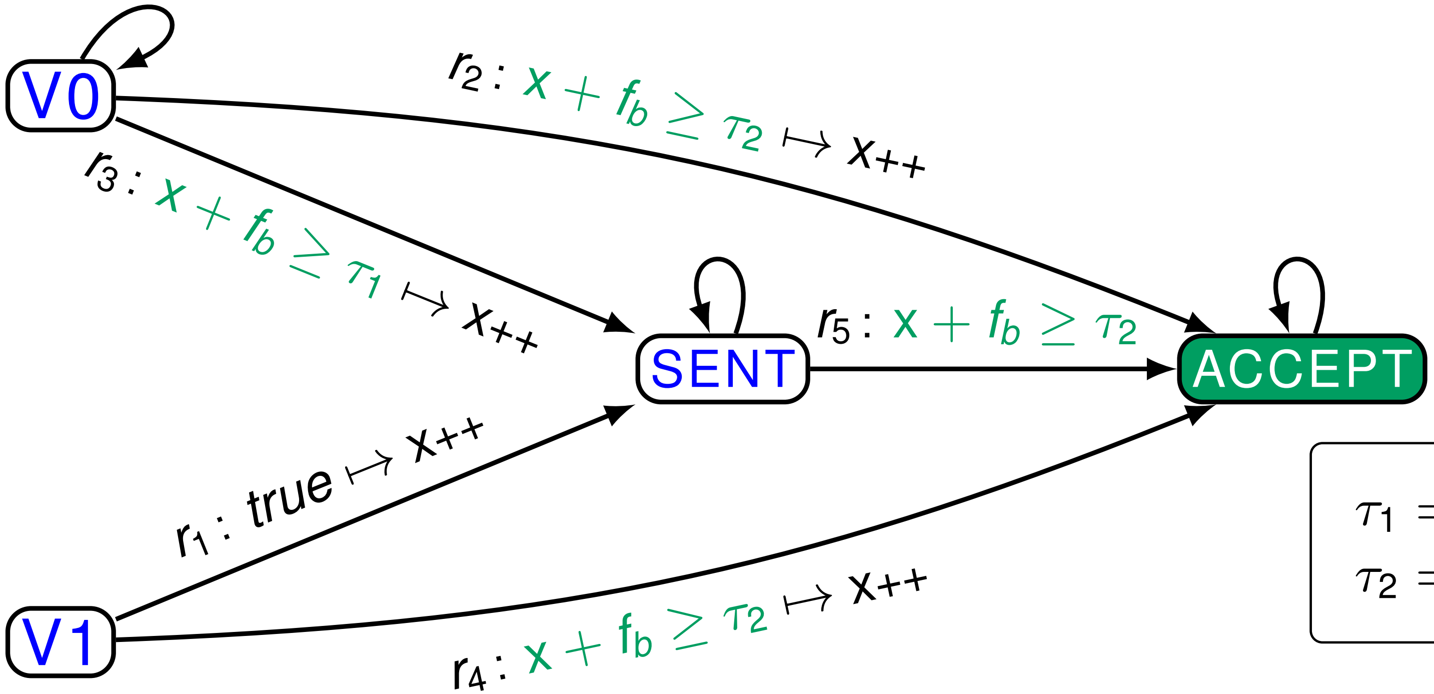
Formalizing pseudo-code

Sketch threshold automata to capture the pseudo-code

Linear temporal logic to formalize the specifications

Linear integer arithmetic to express the resilience condition

Sketch threshold automata



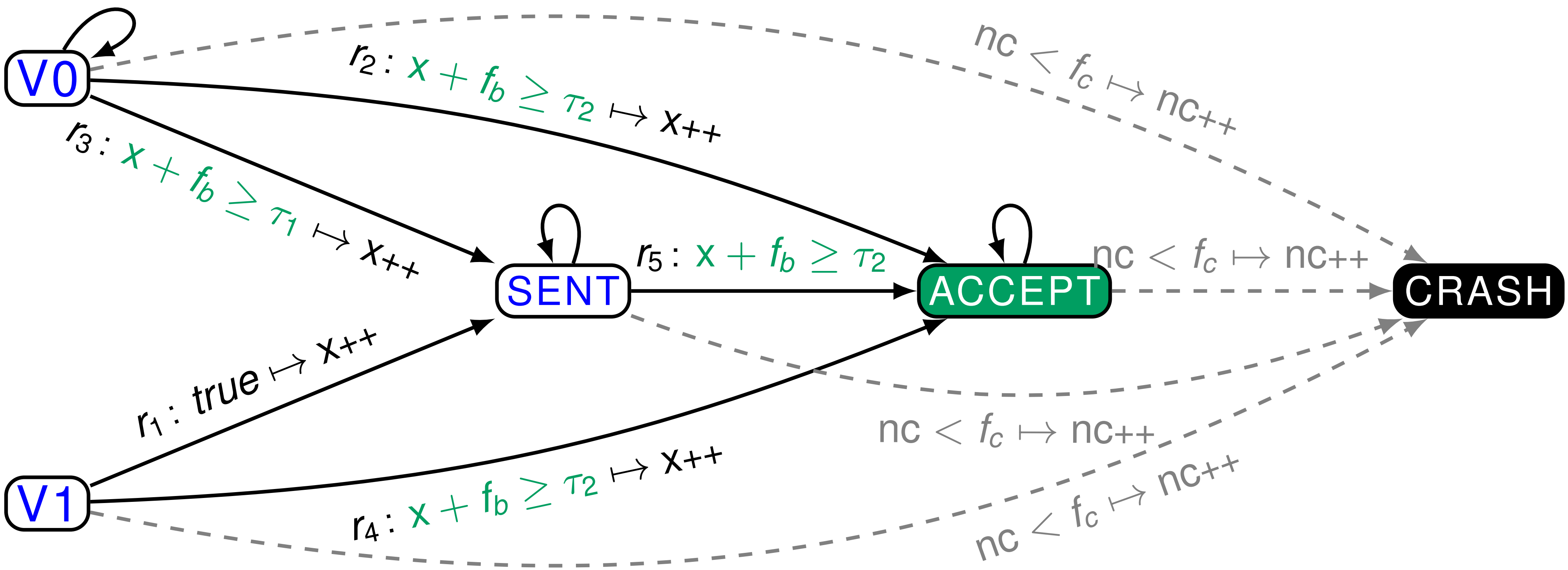
$$\tau_1 = \mathbf{?}_1 \cdot n + \mathbf{?}_2 \cdot t_b + \mathbf{?}_3$$
$$\tau_2 = \mathbf{?}_4 \cdot n + \mathbf{?}_5 \cdot t_b + \mathbf{?}_6$$

Byzantine faults:

run $n - f_b$ processes,

count messages modulo Byzantine processes, e.g., $x + f_b \geq \tau_2$

Sketch threshold automata



$f_b \leq t_b$ **Byzantine** and $f_c \leq t_c$ **crash faults:**

run $n - f_b$ processes,

resilience condition e.g. $n > 3t_b + 2t_c \ \wedge \ t_b \geq 0 \wedge t_c \geq 0$

Linear temporal logic

Relay:

if a correct process accepts **bcast**,
then all correct processes eventually accept **bcast**

Linear temporal logic

$\neg$ **Relay:**

~~if~~ a correct process accepts **bcast**,

~~then all correct processes eventually accept **bcast**~~

and at least one process never accept **bcast**

Linear temporal logic

¬ **Relay:**

~~if~~ a correct process accepts **bcast**

~~then all correct processes eventually accept **bcast**~~

and at least one process never accepts **bcast**

$$\mathbf{E}\left(\mathbf{F}\left(\boldsymbol{\kappa}_{\mathsf{ACCEPT}} \neq 0\right) \wedge \mathbf{G}\left(\boldsymbol{\kappa}_{\mathsf{V1}} \neq 0 \vee \boldsymbol{\kappa}_{\mathsf{V0}} \neq 0 \vee \boldsymbol{\kappa}_{\mathsf{SENT}} \neq 0\right)\right) \wedge \mathbf{G}\,\mathbf{F}\,\psi_{\mathsf{fair}}$$

Linear temporal logic

Relay:

~~if~~ a correct process accepts **bcast**

~~then all correct processes eventually accept **bcast**~~

and at least one process never accepts **bcast**

$$\mathbf{E}\,(\mathbf{F}\,(\boldsymbol{\kappa}_{\text{ACCEPT}} \neq 0) \wedge \mathbf{G}\,(\boldsymbol{\kappa}_{\text{V1}} \neq 0 \vee \boldsymbol{\kappa}_{\text{V0}} \neq 0 \vee \boldsymbol{\kappa}_{\text{SENT}} \neq 0)) \wedge \mathbf{G}\,\mathbf{F}\,\psi_{\text{fair}})$$

Propositional formulas:

(1) $\bigwedge_{\ell \in S} \boldsymbol{\kappa}_\ell = 0$

(2) $\bigvee_{\ell \in S} \boldsymbol{\kappa}_\ell \neq 0$

(3) $\bigwedge_{S \subseteq T} \bigvee_{\ell \in S} \boldsymbol{\kappa}_\ell \neq 0$

(4) $\text{Bool(Guards)} \rightarrow (1) \wedge (2) \wedge (3)$

Temporal formulas:

$\psi ::= \textit{prop} \mid \mathbf{G}\,\psi \mid \mathbf{F}\,\psi \mid \psi \wedge \psi$

We call this fragment ELTL_{FT}

Our solution to synthesis

Synthesis loop

Find $\mathbf{?}_1, \ldots, \mathbf{?}_k \in \mathbb{Q}$

coefficients

counterexample

Generator
infinite
search space

solution

Verifier
Byzantine MC

Termination? sane guards $\Rightarrow$ bounded search space

Efficiency? Generator learns from counterexamples

Sane guards: thresholds lie in the interval $[0, n]$

Classic threshold guards:

`if received` $\frac{n}{2}$ `messages...` ✓ — wait for a majority

`if received` $t+1$ `messages...` ✓ — wait for a correct process

`if received` $n-t$ `messages...` ✓ — wait for non-faulty processes

Syntactically correct but meaningless guards:

`if received` $2n$ `messages...` ✗

`if received` -5 `messages...` ✗

Search space for sane guards

Resilience condition: $n > 3t > 0$

Threshold: $0 \leq \mathbf{?}_a n + \mathbf{?}_b t + \mathbf{?}_c \leq n$

$$\Rightarrow \quad \begin{array}{rcl} 0 \leq & \mathbf{?}_a & \leq 1 \\ -4 \leq & \mathbf{?}_b & \leq 4 \\ -8 \leq & \mathbf{?}_c & \leq 8 \end{array}$$

Theorem

Assume: $n > \sum_{1 \leq i \leq k} \delta_i \cdot t_i$ and $\forall i.\ t_i \geq 0$ — resilience cond.

$0 \leq \mathbf{?}_a n + \sum_{1 \leq i \leq k} \mathbf{?}_{b_i} \cdot t_i + \mathbf{?}_c \leq n$ — threshold

Then:

$$\begin{cases} 0 \leq \mathbf{?}_a \leq 1 \\ -B_i \leq \mathbf{?}_{b_i} \leq B_i \quad \text{for } B_i = \delta_i + 1 \text{ and } 1 \leq i \leq k \\ -C \leq \mathbf{?}_c \leq C \quad \text{for } C = k + 1 + 2(\delta_1 + \cdots + \delta_k) \end{cases}$$

Search space for sane guards

Resilience condition: $n > 3t > 0$

Threshold: $0 \leq \mathbf{?}_a n + \mathbf{?}_b t + \mathbf{?}_c \leq n$

$\Rightarrow$

$$\begin{aligned} 0 &\leq \mathbf{?}_a \leq 1 \\ -4 &\leq \mathbf{?}_b \leq 4 \\ -8 &\leq \mathbf{?}_c \leq 8 \end{aligned}$$

Theorem

Assume: $n > \sum_{1 \leq i \leq k} \delta_i \cdot t_i$ and $\forall i.\ t_i \geq 0$ — resilience cond.

$$0 \leq \mathbf{?}_a n + \sum_{1 \leq i \leq k} \mathbf{?}_{b_i} \cdot t_i + \mathbf{?}_c \leq n$$ — threshold

Then:

$$\begin{cases} 0 \leq \mathbf{?}_a \leq 1 & \\ -B_i \leq \mathbf{?}_{b_i} \leq B_i & \text{for } B_i = \delta_i + 1 \text{ and } 1 \leq i \leq k \\ -C \leq \mathbf{?}_c \leq C & \text{for } C = k + 1 + 2(\delta_1 + \cdots + \delta_k) \end{cases}$$

From $\mathbb{Q}$ to a finite search space

Synthesizing thresholds of the form $\frac{n}{2}$ or $\frac{2n}{3}$:

Assume:

Resilience condition: $n > 3t \geq 0$

Threshold: $0 \leq \frac{\mathbf{?}'_a}{6} n + \frac{\mathbf{?}'_b}{6} t + \frac{\mathbf{?}'_c}{6} \leq n$

Then

$$0 \leq \mathbf{?}'_a \leq 6 \cdot 1$$
$$-6 \cdot 4 \leq \mathbf{?}'_b \leq 6 \cdot 4$$
$$-6 \cdot 8 \leq \mathbf{?}'_c \leq 6 \cdot 8$$

$$\mathbf{?}'_a, \mathbf{?}'_b, \mathbf{?}'_c \in \mathbb{Z}$$

Explicit enumeration?

coefficients $(\mathbf{?}_1, \mathbf{?}_2, \mathbf{?}_3, \mathbf{?}_4, \mathbf{?}_5, \mathbf{?}_6)$ in reliable broadcast:

$$(2 \cdot 9 \cdot 17)^2 \text{ vectors}$$

coefficients $(0, \mathbf{?}_2, 1, 0, \mathbf{?}_5, 1)$ in reliable broadcast:

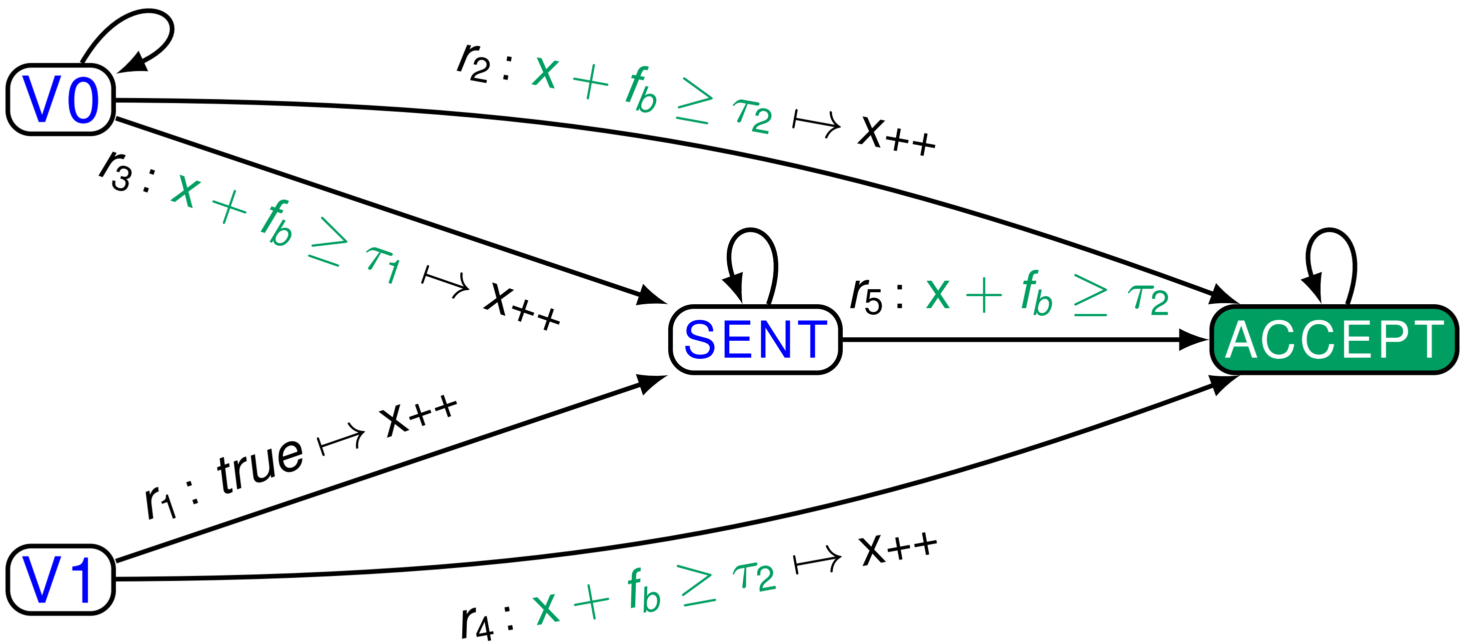
$$9 \cdot 9 \text{ vectors}$$

coefficients $(\mathbf{?}_1, \mathbf{?}_2, \dots, \mathbf{?}_9)$ in one-step consensus (BOSCO):

$$(3 \cdot 17 \cdot 33)^3 \text{ vectors}$$

The generator should learn from the counterexamples!

Sketch of reliable broadcast in 2D

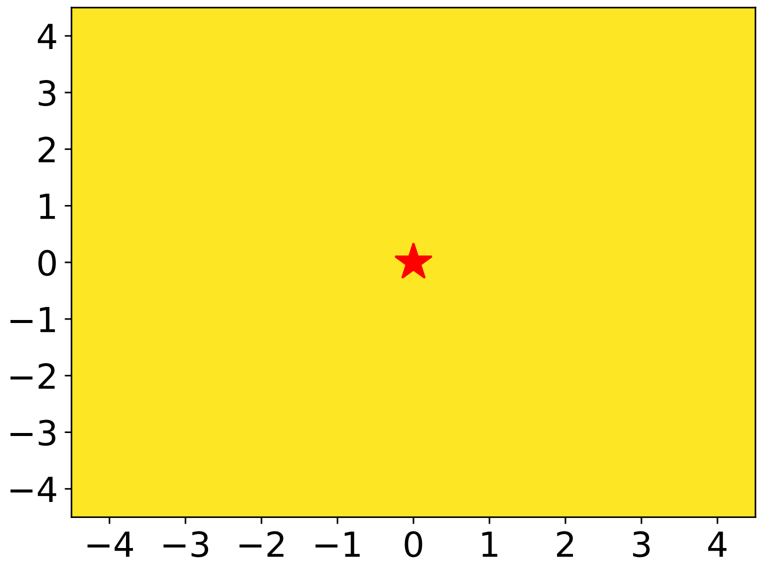


missing coefficients for t:

$\tau_1 = \mathbf{?}_2 \cdot t + 1$

$\tau_2 = \mathbf{?}_5 \cdot t + 1$

bounded search space



Counterexample to unforgeability

Model checker flags an error:

$n = 4$, $t = 1$, and $f = 1$

State 1. $\boldsymbol{\kappa}_{\mathsf{V0}} = 3$, other counters 0

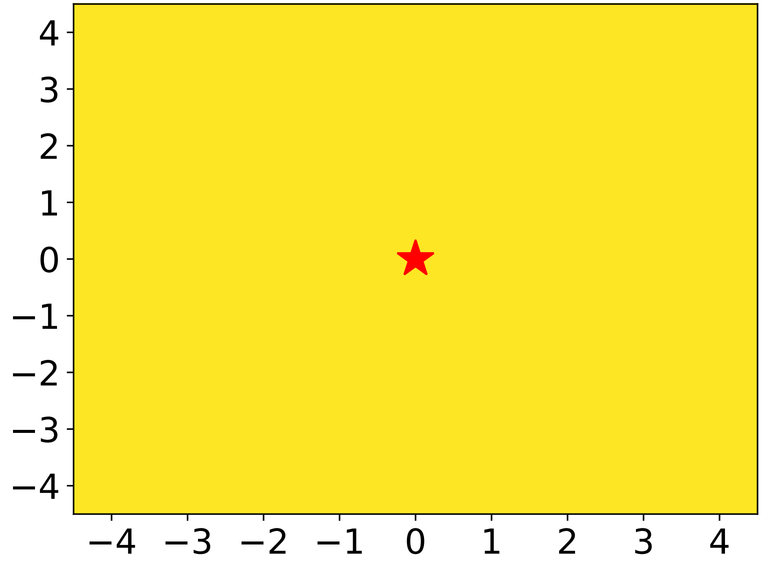
$$\Downarrow r_3 : 0 + 1 \geq 0 \cdot 1 + 1$$

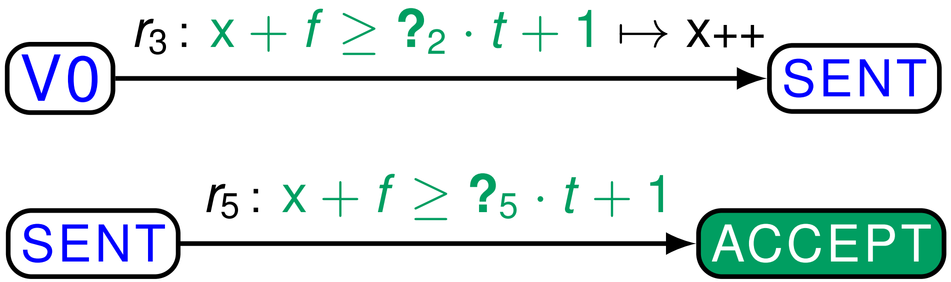
State 2. $\boldsymbol{\kappa}_{\mathsf{SENT}} = 1$, $x = 1$, $\boldsymbol{\kappa}_{\mathsf{V0}} = 2$

$$\Downarrow r_5 : 1 + 1 \geq 0 \cdot 1 + 1$$

State 3. $\boldsymbol{\kappa}_{\mathsf{ACCEPT}} = 1$, $\boldsymbol{\kappa}_{\mathsf{SENT}} = 0$

$\mathbf{?}_2 = 0$ and $\mathbf{?}_5 = 0$





Counterexample to unforgeability

Model checker flags an error:

$n = 4$, $t = 1$, and $f = 1$

State 1. $\kappa_{\mathrm{V0}} = 3$, other counters 0

~~$\Downarrow r_3 : 0 + 1 \geq 0 \cdot 1 + 1$~~

$\Downarrow r_3 : 0 + 1 \geq \mathbf{?}_2 \cdot 1 + 1$

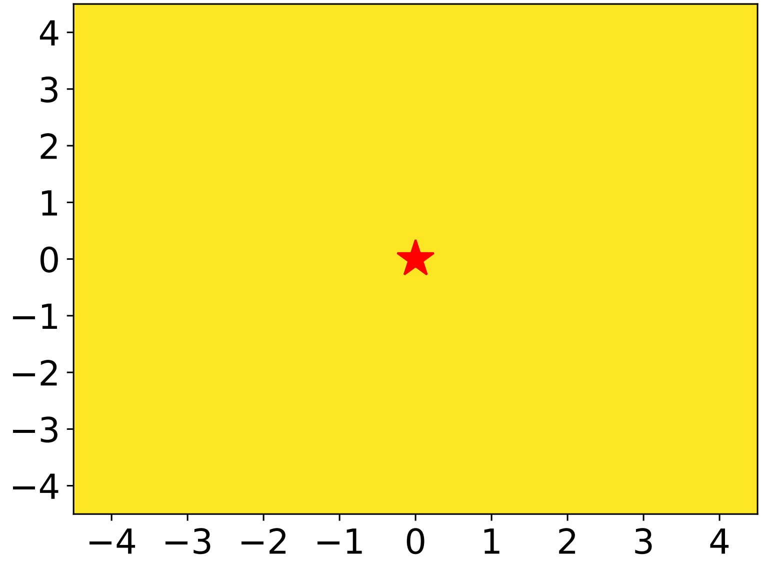
State 2. $\kappa_{\mathrm{SENT}} = 1$, $x = 1$, $\kappa_{\mathrm{V0}} = 2$

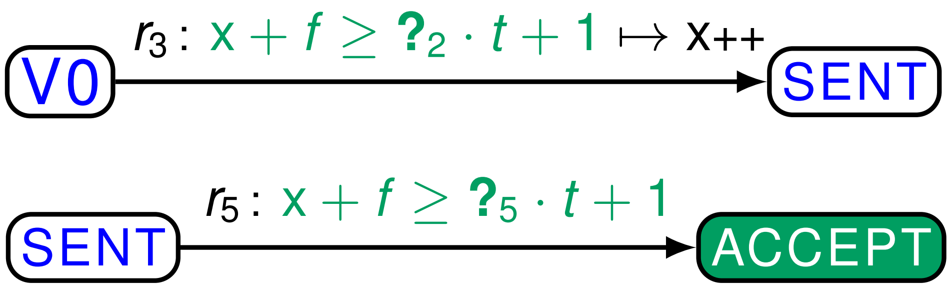
~~$\Downarrow r_5 : 1 + 1 \geq 0 \cdot 1 + 1$~~

$\Downarrow r_5 : 1 + 1 \geq \mathbf{?}_5 \cdot 1 + 1$

State 3. $\kappa_{\mathrm{ACCEPT}} = 1$, $\kappa_{\mathrm{SENT}} = 0$

$\mathbf{?}_2 = 0$ and $\mathbf{?}_5 = 0$





Counterexample to unforgeability

Model checker flags an error:

$$n = 4,\ t = 1, \text{ and } f = 1$$

State 1. $\kappa_{\mathrm{V0}} = 3$, other counters 0

~~$\Downarrow\ r_3 : 0 + 1 \geq 0 \cdot 1 + 1$~~

$\Downarrow\ r_3 : 0 + 1 \geq \mathbf{?}_2 \cdot 1 + 1$

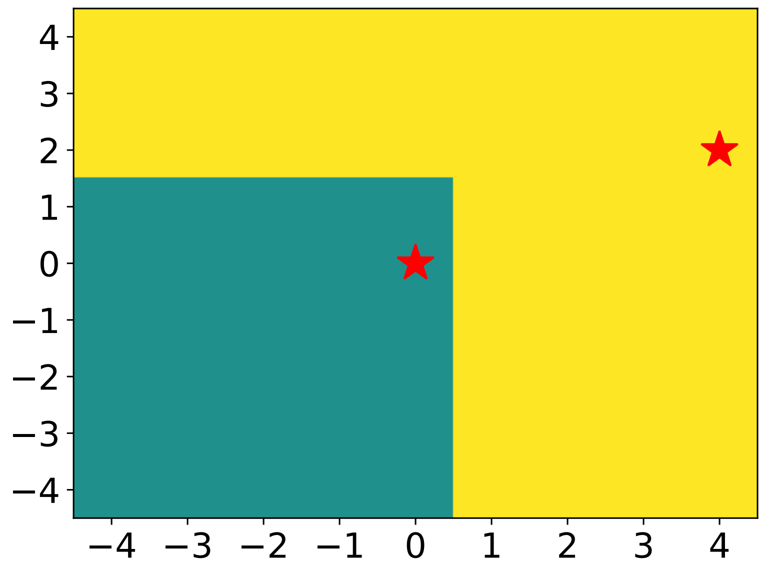
State 2. $\kappa_{\mathrm{SENT}} = 1,\ x = 1,\ \kappa_{\mathrm{V0}} = 2$

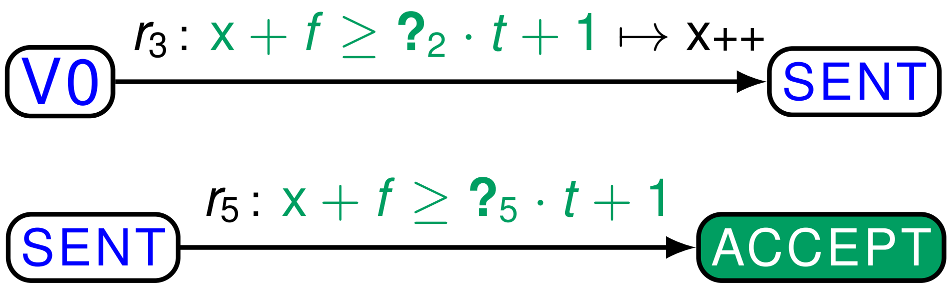
~~$\Downarrow\ r_5 : 1 + 1 \geq 0 \cdot 1 + 1$~~

$\Downarrow\ r_5 : 1 + 1 \geq \mathbf{?}_5 \cdot 1 + 1$

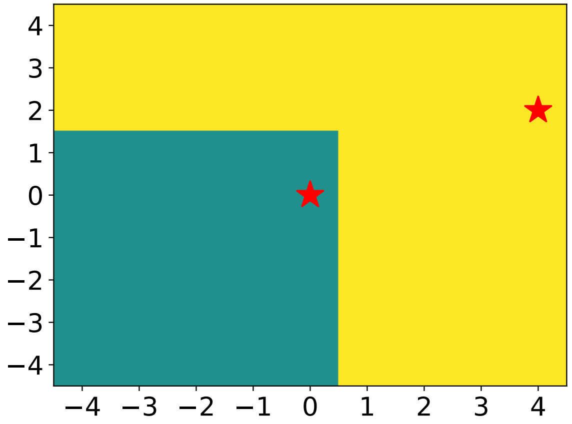
State 3. $\kappa_{\mathrm{ACCEPT}} = 1,\ \kappa_{\mathrm{SENT}} = 0$

$\mathbf{?}_2 = 0$ and $\mathbf{?}_5 = 0$

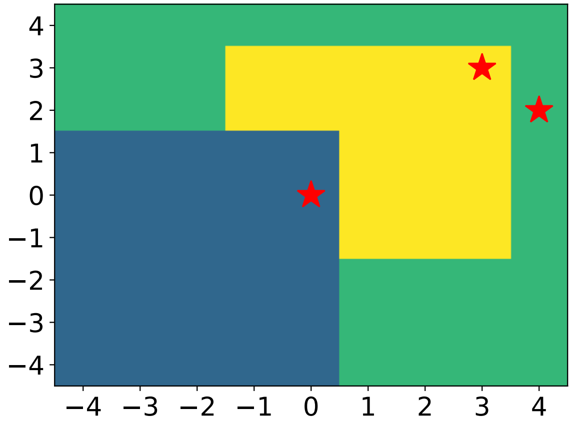




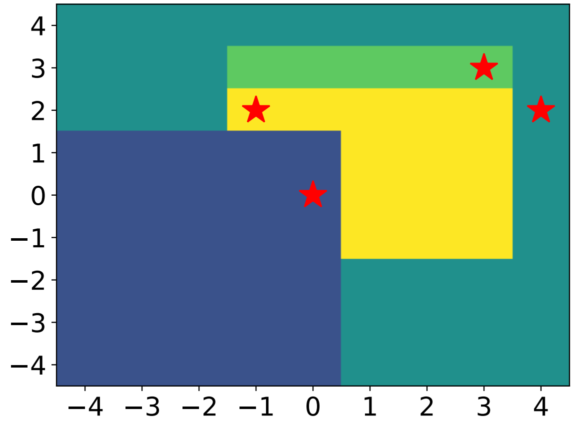
Learning from counterexamples



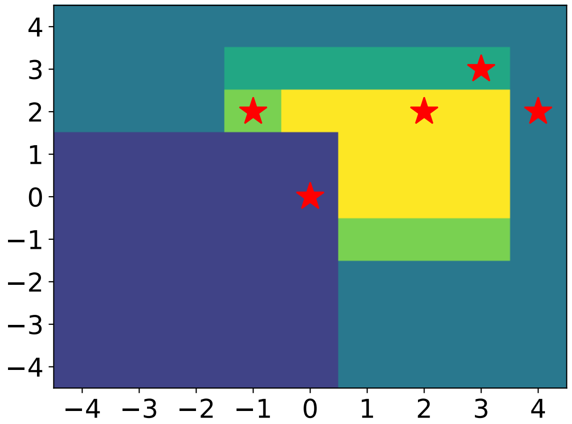
1. unforgeability



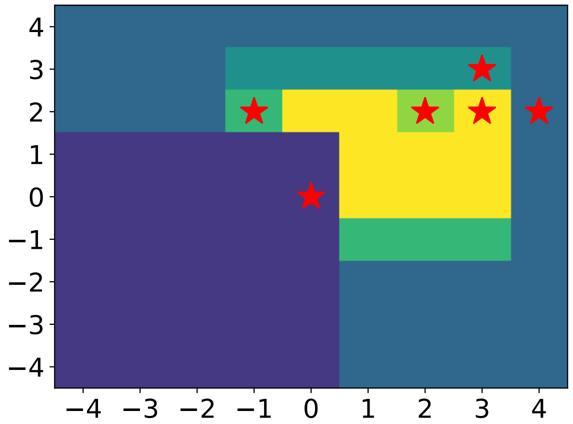
2. sanity



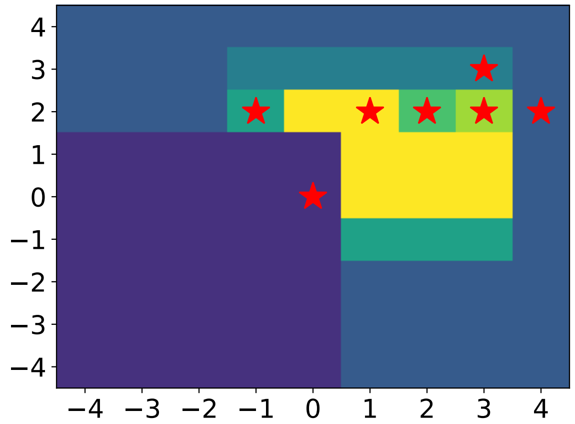
3. correctness



4. sanity



5. relay

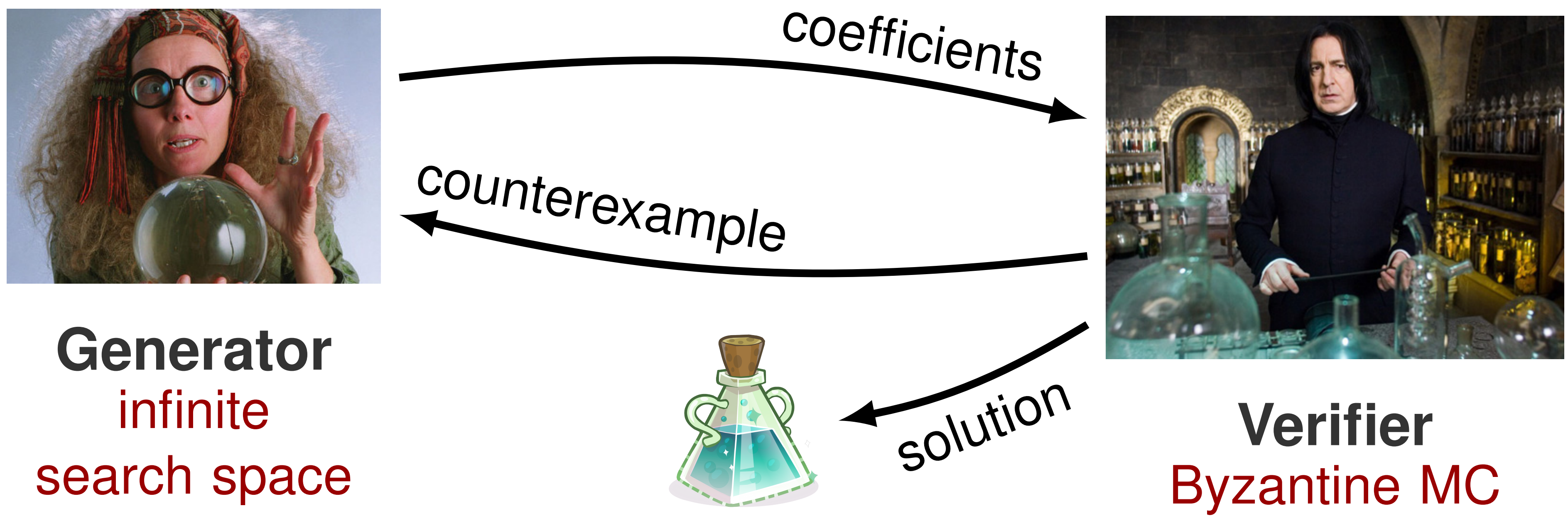


6. relay

found the solution $?_2 = 1$ and $?_5 = 2$

Synthesis loop

Find $?_1, \ldots, ?_k \in \mathbb{Q}$



Experiments

We have synthesized

reliable broadcast, hybrid broadcast, and

BOSCO (one-step consensus) using

&

©VSC / Claudia Blaas-Schenner

[forsyte.at/software/bymc]

SYNTHESIZED
Correct-By-Construction

Thresholds for Byzantine reliable broadcast

local $myval_i \in \{0, 1\}$

3 solutions
16 seconds
31 calls to verifier

while true **do**
  if $myval_i = 1$ **and not** sent ECHO before
  then send ECHO to all

  if received ECHO from at least t + 1 **distinct** processes
    and not sent ECHO before
  then send ECHO to all

  if received ECHO from at least n - t **distinct** processes
  then accept
od

Candidate thresholds for the first check: $t+1$, $n-2t$

Candidate thresholds for the second check: $2t+1$, $n-t$

✓ ✓ ✓

resilience: $n > 3t$, $f \leq t$ **Byzantine** faults

Thresholds for Byzantine reliable broadcast

```
local myval_i ∈ {0,1}

while true do
 if myval_i = 1 and not sent ECHO before
 then send ECHO to all

 if received ECHO from at least [✗] distinct processes
    and not sent ECHO before
 then send ECHO to all

 if received ECHO from at least [✗] distinct processes
 then accept
od
```

0 solutions
7 seconds
25 calls to verifier

$n \geq 3t$

resilience: $n > 3t$, $f \leq t$ **Byzantine** faults

Thresholds for hybrid reliable broadcast

local $myval_i \in \{0, 1\}$

while `true` **do**
 if $myval_i = 1$ **and not** `sent ECHO before`
 then `send ECHO to all`

 if `received ECHO from at least` $t_b + 1$ / $t_b + 1$ / $n - 2t_b - 2t_c$ **distinct** `processes`
  and not `sent ECHO before`
 then `send ECHO to all`

 if `received ECHO from at least` $2t_b + t_c + 1$ / $n - t_b + t_c$ / $n - t_b - t_c$ **distinct** `processes`
 then `accept`
od

3 solutions
50 seconds
34 calls to verifier

✓ ✓ ✓

resilience: $n > 3t_b + 2t_c$ $f_b \leq t_b$ **Byzantine** and $f_c \leq t_c$ **crash** faults

Thresholds for hybrid reliable broadcast

```
local myval_i ∈ {0, 1}

while true do
  if myval_i = 1 and not sent ECHO before
  then send ECHO to all

  if received ECHO from at least  ✕  distinct processes
     and not sent ECHO before
  then send ECHO to all

  if received ECHO from at least  ✕  distinct processes
  then accept
od
```

0 solutions
24 seconds
29 calls to verifier

$n > 3t_b + t_c$

resilience: $n > 3t_b + 2t_c$ $f_b \leq t_b$ **Byzantine** and $f_c \leq t_c$ **crash** faults

Reliable broadcast: changing specifications

$\leq X$

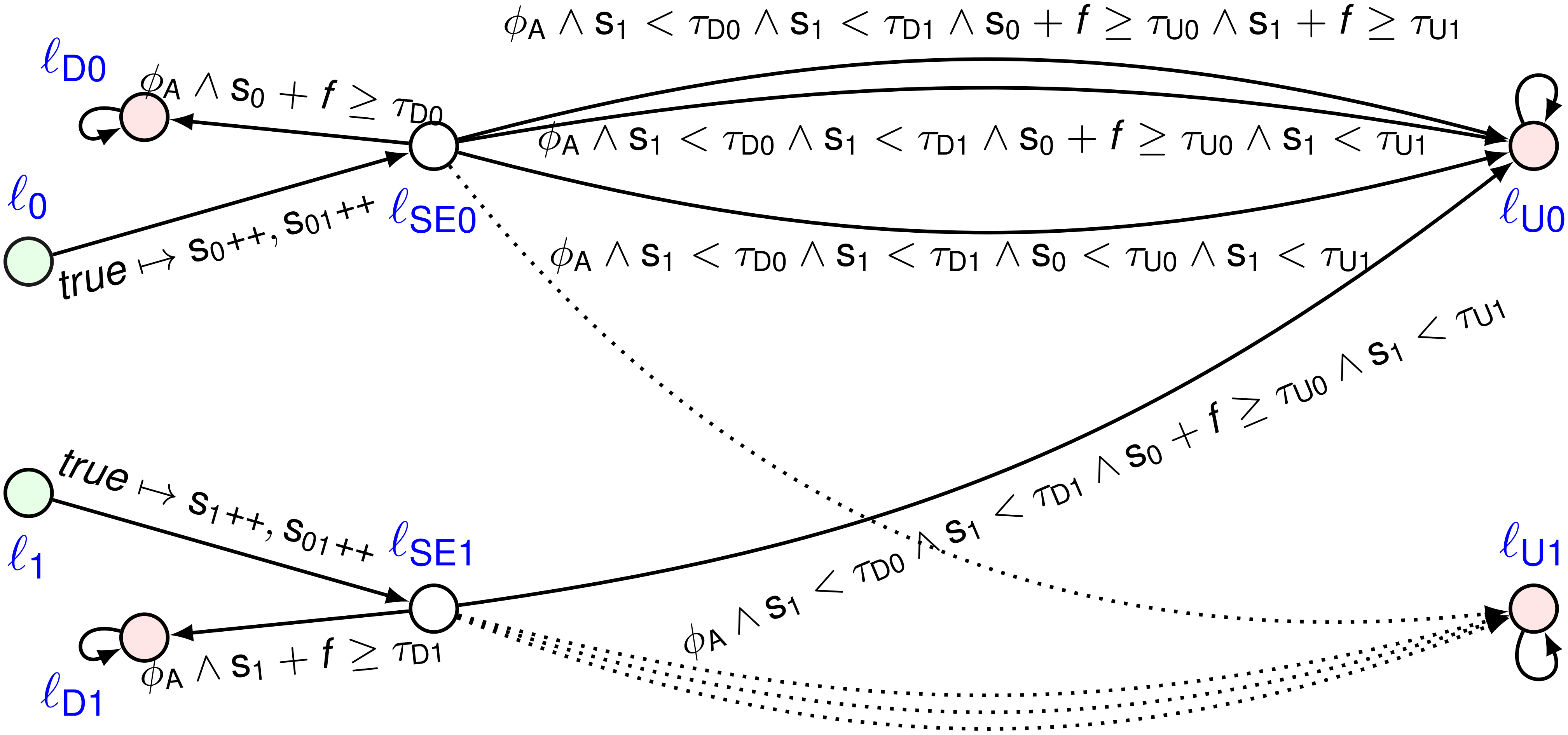
unforgeability: if ~~no~~ correct process received **bcast**,
then no correct process ever accepts **bcast**

correctness and **relay** like before

$\underline{X = 2}$	$\underline{X = t}$
no solutions for $n > 3t$	no solutions for $n > 3t$
3 solutions for $n > 3t + 2$	3 solutions for $n > 4t$

each answer found in less than 12 sec.

BOSCO (one-step consensus)



BOSCO: synthesis results

$$\left.\begin{array}{r}\textbf{Agreement}\\ \textbf{Termination}\end{array}\right\}\text{ when } n > 3t$$

$$\left.\begin{array}{r}\textbf{One step}\\ \textbf{Fast termination}\end{array}\right\}\text{ when } n > 7t \quad \text{or} \quad n > 5t, f = 0$$

Found 4 solutions using 4 cluster nodes = 64 cores **24 min.**

No solutions for $n \geq 5t, f = 0$ and $n \geq 7t$ **40 min.**

(the conditions $n > 5t$ and $n > 7$ are tight)

Conclusions

we can **verify** and **synthesize** distributed algorithms that are:

- asynchronous and parameterized,
- subject to faults, sending to all, and
- counting messages and comparing to threshold guards

next steps:

- learning from positive examples,
- geometrical structure of learned regions,
- synthesizing threshold automata, not just thresholds

[forsyte.at/software/bymc]