Model Checking of Fault-Tolerant Distributed Algorithms

Part V: On the Completeness of Bounded Model Checking: Reachability

Igor Konnov Helmut Veith Josef Widder

VTSA 2014, Luxembourg

Back to the big picture

Why fault-tolerant (FT) distributed algorithms

faults not in the control of system designer

- bit-flips in memory
- power outage
- disconnection from the network
- intruders take control over some computers

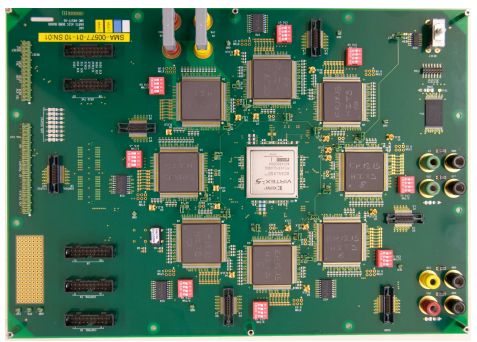
Why fault-tolerant (FT) distributed algorithms

faults not in the control of system designer

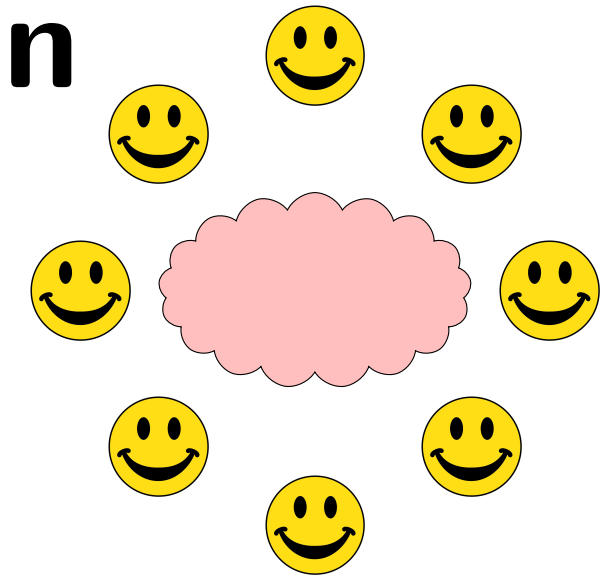
- bit-flips in memory
- power outage
- disconnection from the network
- intruders take control over some computers

distributed algorithms intended to make systems more reliable even in the presence of faults

- replicate processes
- exchange messages
- do coordinated computation
- goal: keep replicated processes in "good state"

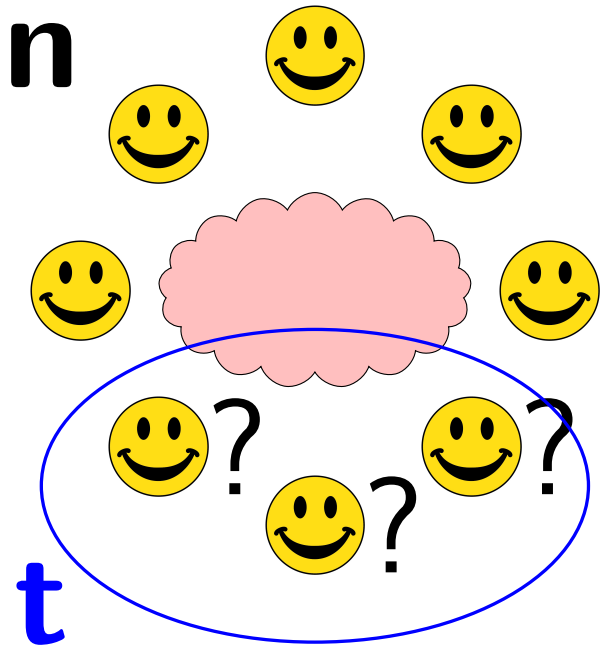


Fault-tolerant distributed algorithms



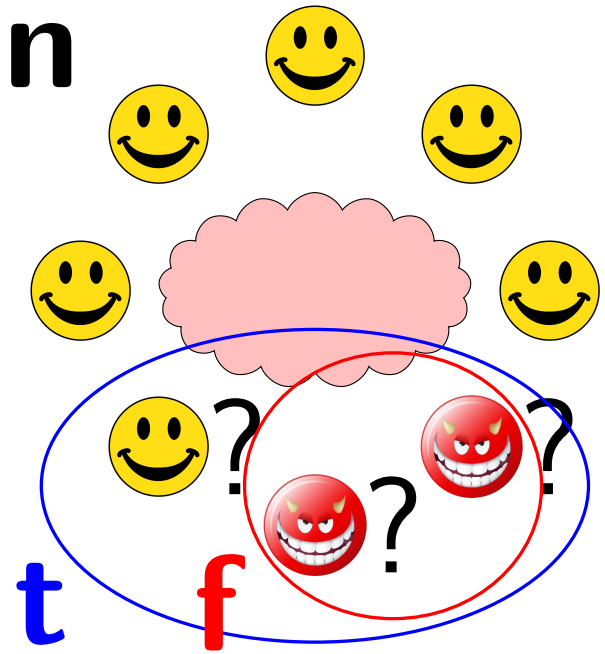
- n processes communicate by messages

Fault-tolerant distributed algorithms



- n processes communicate by messages
- all processes know that at most t of them might be faulty

Fault-tolerant distributed algorithms



- n processes communicate by messages
- all processes know that at most t of them might be faulty
- f are actually faulty, e.g., Byzantine
- resilience condition, e.g., $n > 3t \wedge t \geq f \geq 0$
- no masquerading: the processes know the origin of incoming messages

Distributed algorithms: computational model and faults

In previous parts, we considered algorithms operating
in the classic model by [Fischer, Lynch, Paterson'85]

Environment:

- Asynchronous processes (interleaving semantics)
- Reliable asynchronous message passing (non-blocking send and receive)

Faults:

- crashes and clean crashes,
- omission faults,
- symmetric faults,
- Byzantine faults

Threshold-based fault-tolerant distributed algorithms

- The parameters (n, t, f) are fixed in each run
- Main loop with the body executed atomically
- Processes are anonymous (no identifiers)

- Receiving messages, counting them and comparing to thresholds, e.g.,

  `if received <ECHO> from` $t + 1$ `distinct processes then ...`

- Sending messages to all processes, e.g.,

  `send <ECHO> to all`

Case studies: asynchronous threshold-based FTDAs

- Folklore reliable broadcast (FRB) [Chandra, Toueg'96]

  6 counters
- Consistent broadcast (STRB) [Srikanth, Toueg'87]

  7 counters
- Byzantine agreement (ABA) [Bracha, Toueg'85]

  case 1: **37** counters, case 2: **61** counters
- Condition-based consensus (CBC)

  [Mostefaoui, Nourgaya, Parvedy, Raynal'03]

  case 1: **71** counters, case 2: **115** counters
- Non-blocking atomic commitment (NBAC and NBACC)

  [Raynal'97], [Guerraoui'02]

  case 1: **77** counters, case 2: **109** counters

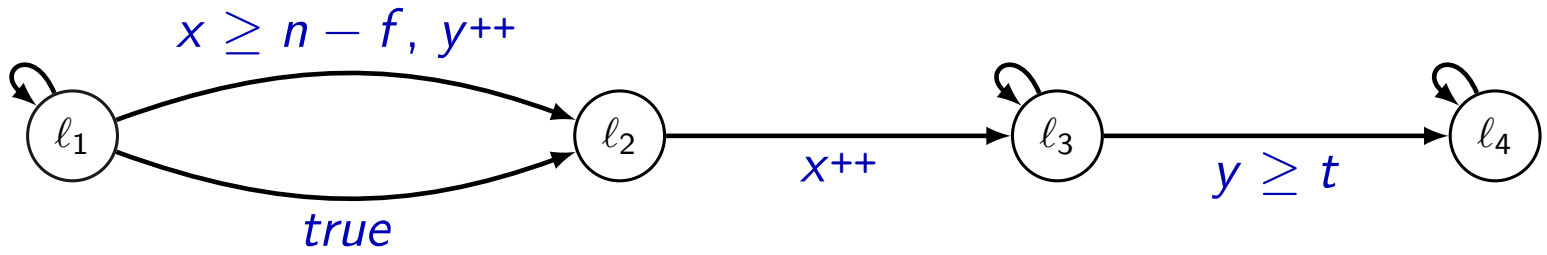
Part V: Outline

1. Yet another abstract model: threshold automata
2. Counter systems with acceleration
3. Parameterized reachability
4. Bounded model checking and its completeness
5. Parameterized bounded model checking and its completeness
6. **Main result:**
   diameter of accelerated counter systems (of threshold automata)

Threshold automata and parameterized reachability

Threshold automata (TA)

Every correct process follows the control flow graph (L, E):



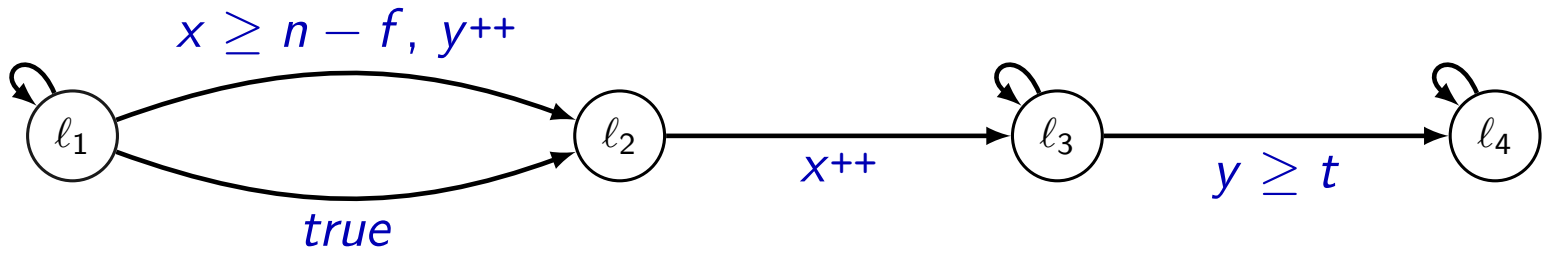
Processes move from one location to another along the edges labeled with:

- **Threshold conditions:**
  - Comparison of a shared variable to linear combinations of parameters,
    e.g., $x \geq t + 1$.
  - Conjunction of comparisons,
    e.g., $x \geq t + 1 \wedge x < n - t$.
- **Updates:**

  Increment shared variables (or do nothing),
  e.g., x^{++}.

Threshold automata (TA)

Every correct process follows the control flow graph (L, E):



Processes move from one location to another along the edges labeled with:

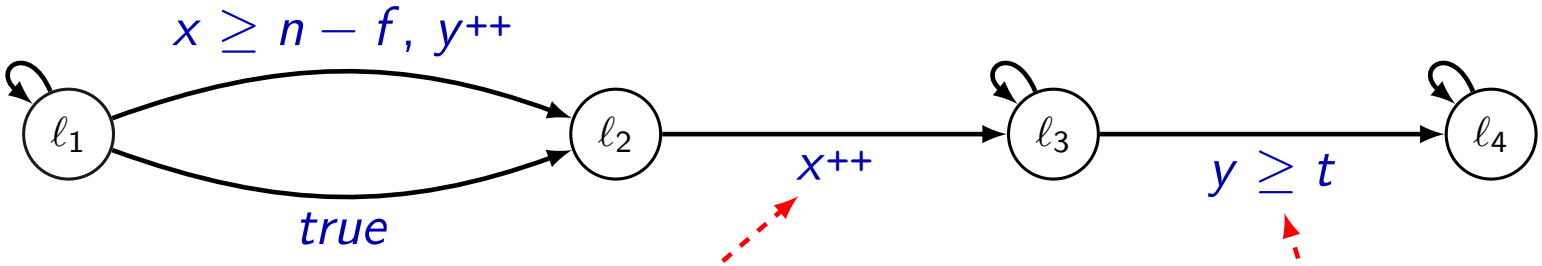
- **Threshold conditions:**
  - Comparison of a shared variable to linear combinations of parameters,
    e.g., $x \geq t + 1$.
  - Conjunction of comparisons,
    e.g., $x \geq t + 1 \wedge x < n - t$.
- **Updates:**

  Increment shared variables (or do nothing),
  e.g., x^{++}.

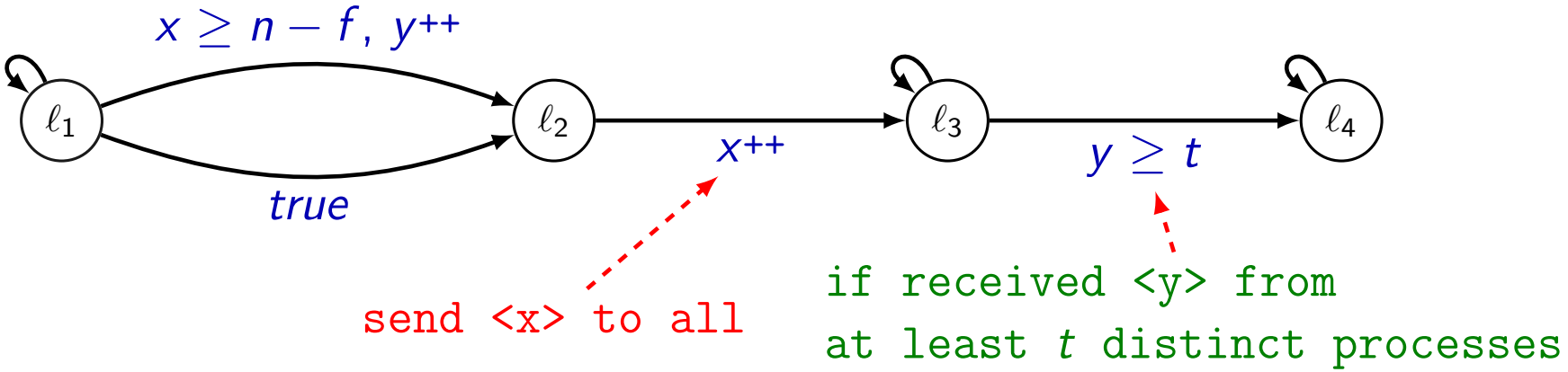
The case studies lead us to the natural restriction on the cycles:

Restriction: the edges in cycles do not change the shared variables.

Intuition: threshold automata and threshold-based DAs?

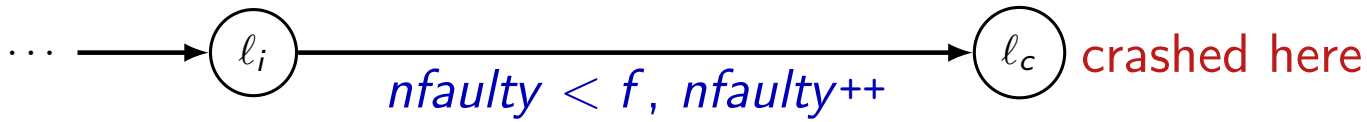


Intuition: threshold automata and threshold-based DAs?



Crash faults:

run n processes,

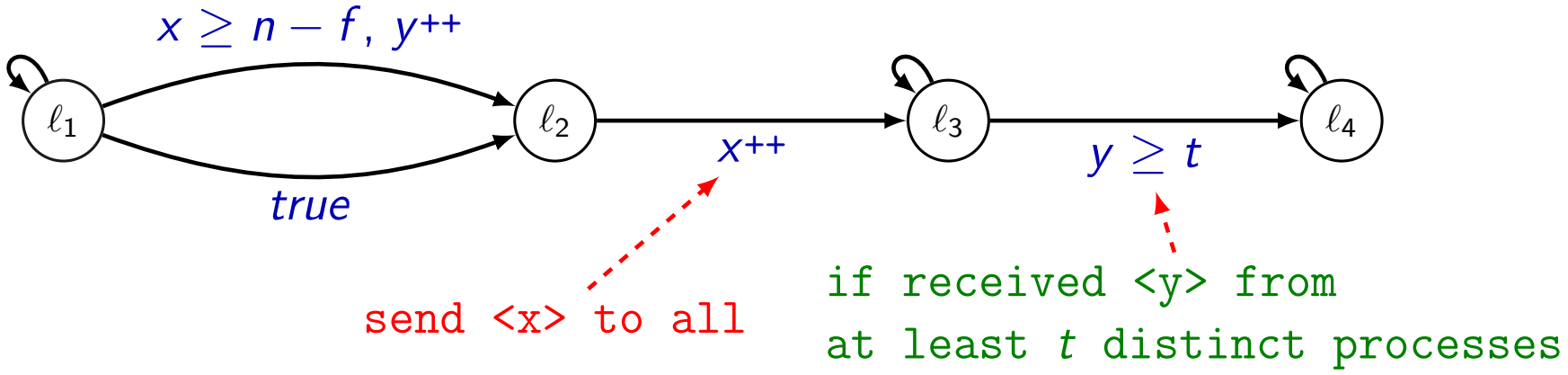


Byzantine faults:

run $n - f$ processes,

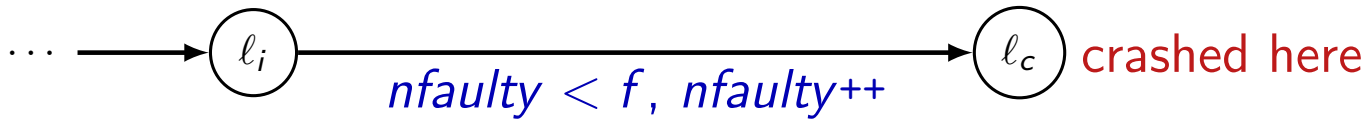
count messages modulo Byzantine processes, e.g., $x \geq (t + 1) - f$

Intuition: threshold automata and threshold-based DAs?



Crash faults:

run n processes,



Byzantine faults:

run $n - f$ processes,

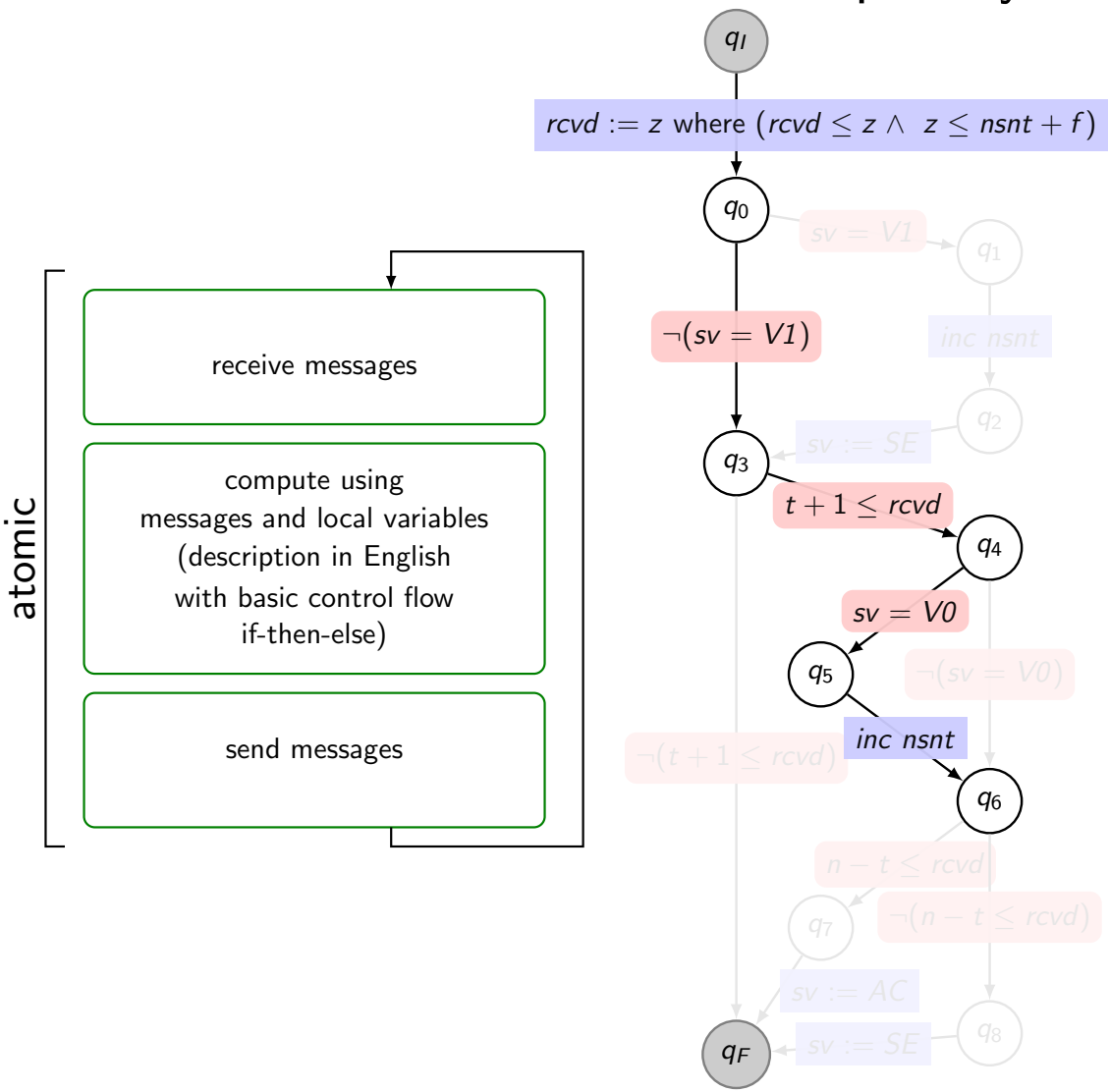
count messages modulo Byzantine processes, e.g., $x \geq (t + 1) - f$

Warning:

Preliminary abstraction is needed as described in Parts II, III.

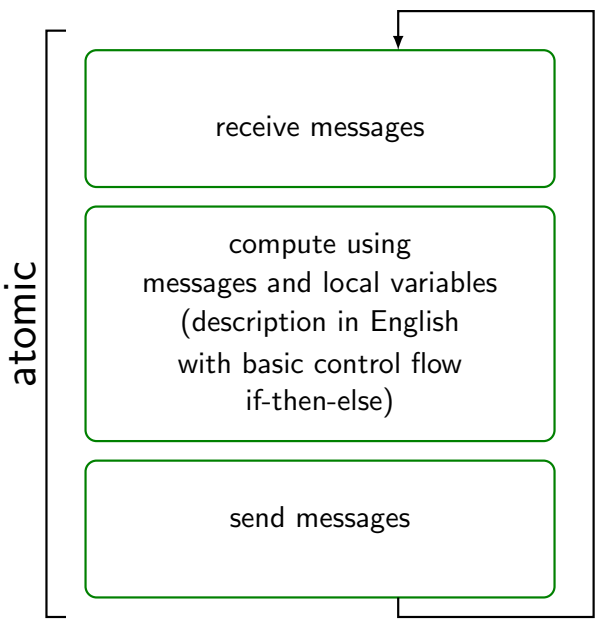
Refresher: control flow automata and their abstraction

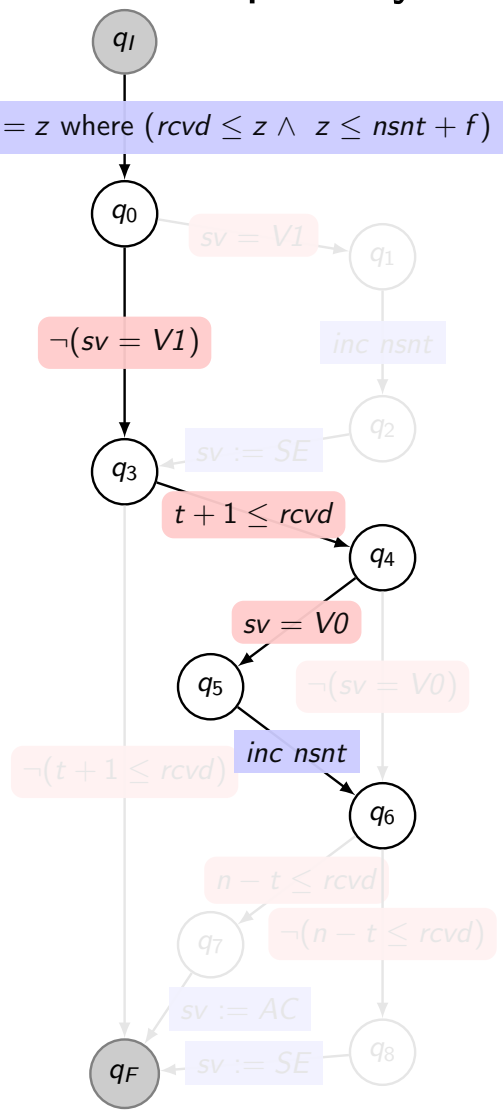
In Parts II, III, we encoded the loop body as a CFA:

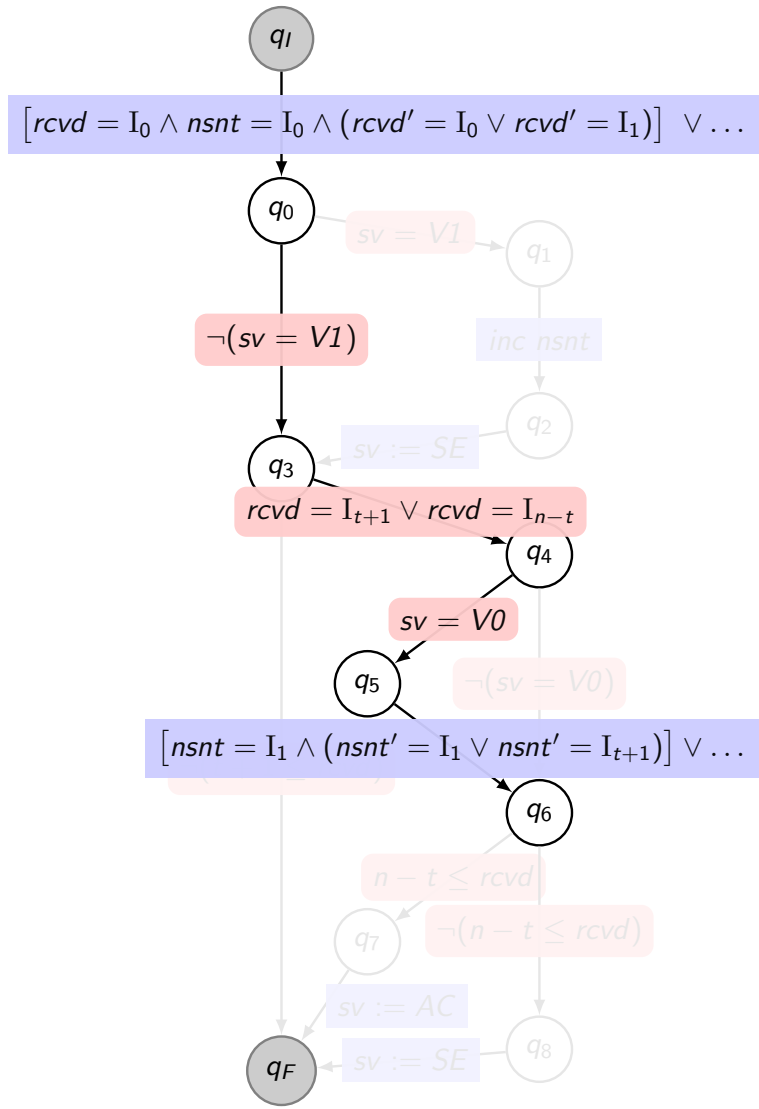


Refresher: control flow automata and their abstraction

In Parts II, III, we encoded the loop body as a CFA:







Intuition: from CFA to TA

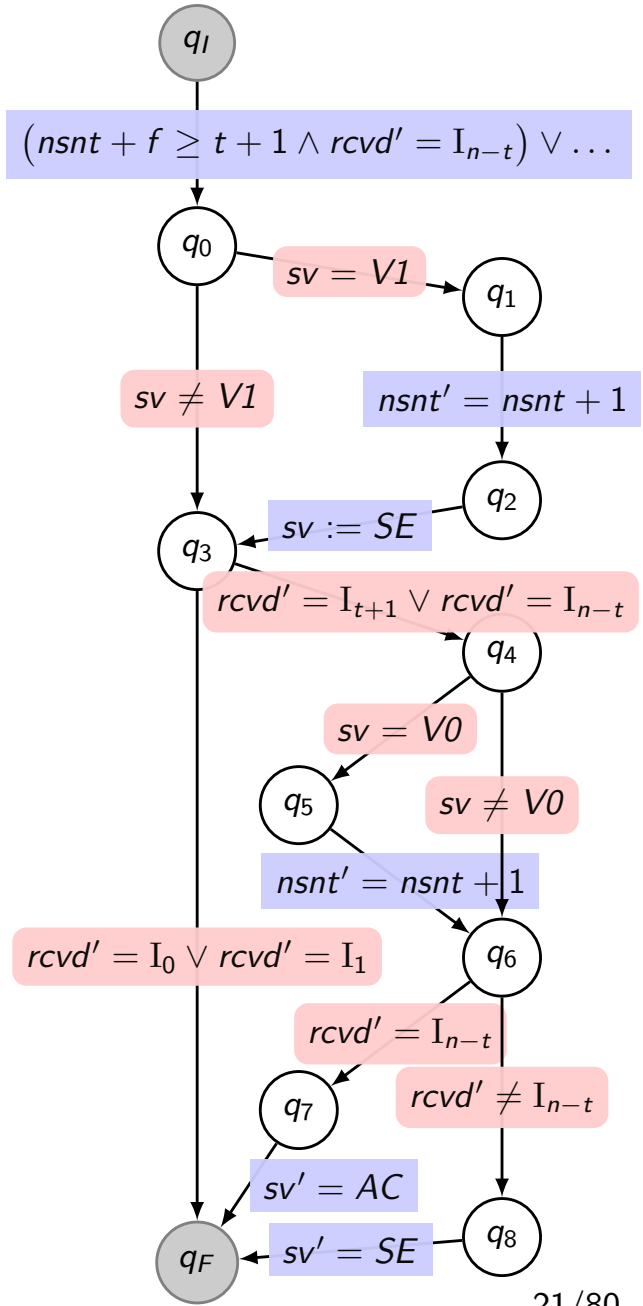
How to construct TA from CFA?

- Apply parametric interval abstraction only to the local variables, e.g., *rcvd*
- Shared variables, e.g., *nsnt*, are still unbounded

Intuition: from CFA to TA

How to construct TA from CFA?

- Apply parametric interval abstraction only to the local variables, e.g., *rcvd*
- Shared variables, e.g., *nsnt*, are still unbounded

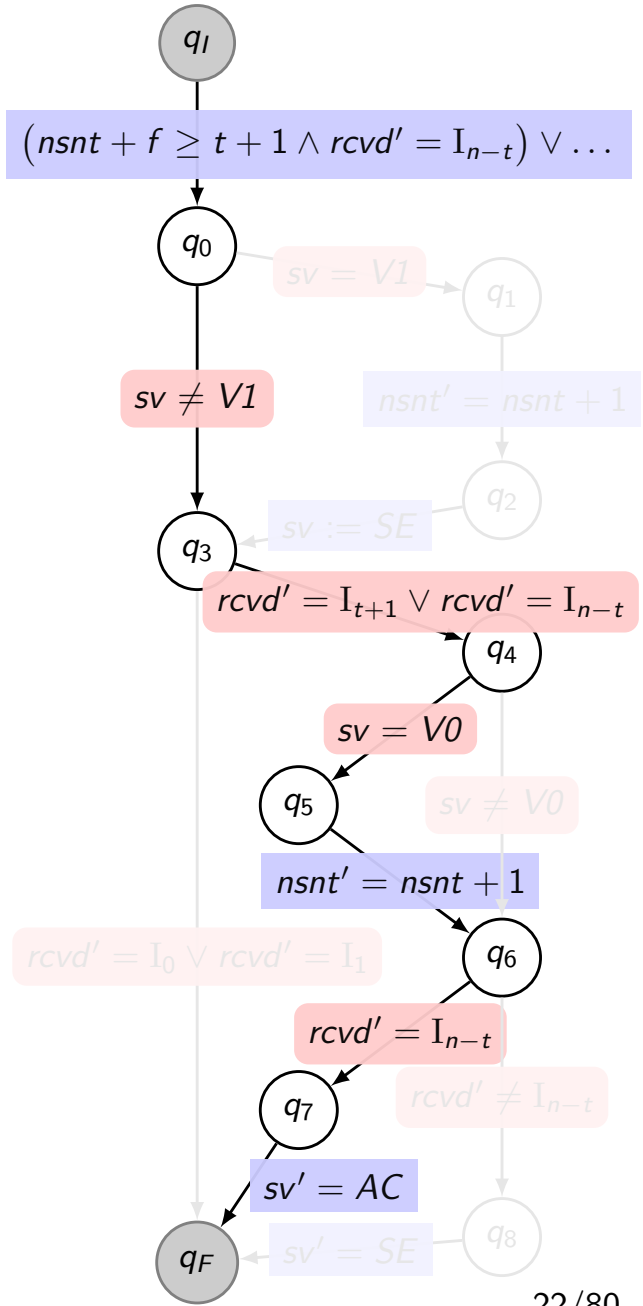


Intuition: from CFA to TA

How to construct TA from CFA?

- Apply parametric interval abstraction only to the local variables, e.g., *rcvd*
- Shared variables, e.g., *nsnt*, are still unbounded

- Enumerate all symbolic paths in CFA

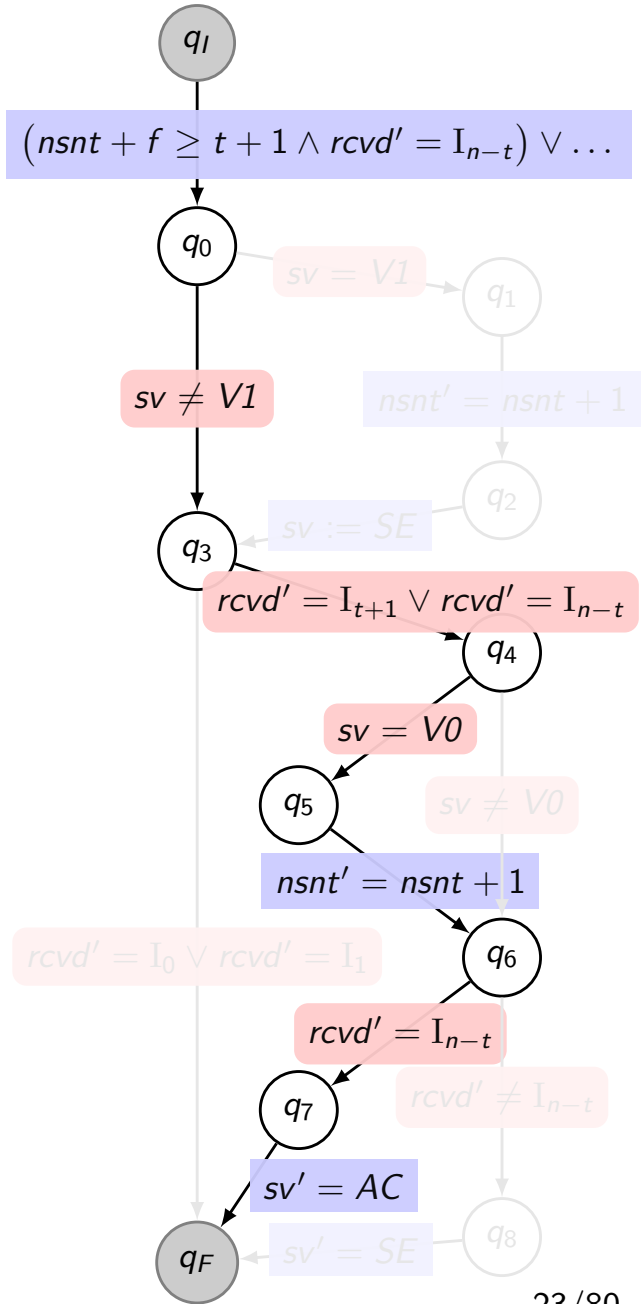


Intuition: from CFA to TA

How to construct TA from CFA?

- Apply parametric interval abstraction only to the local variables, e.g., *rcvd*
- Shared variables, e.g., *nsnt*, are still unbounded

- Enumerate all symbolic paths in CFA
- Use SMT to find all satisfying assignments of local variables
- Each of them gives a TA rule

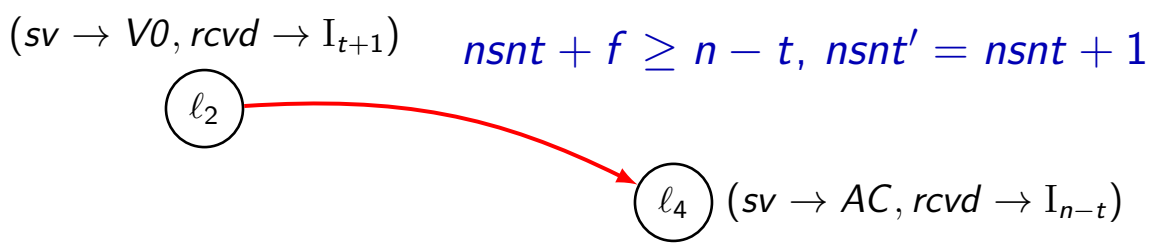


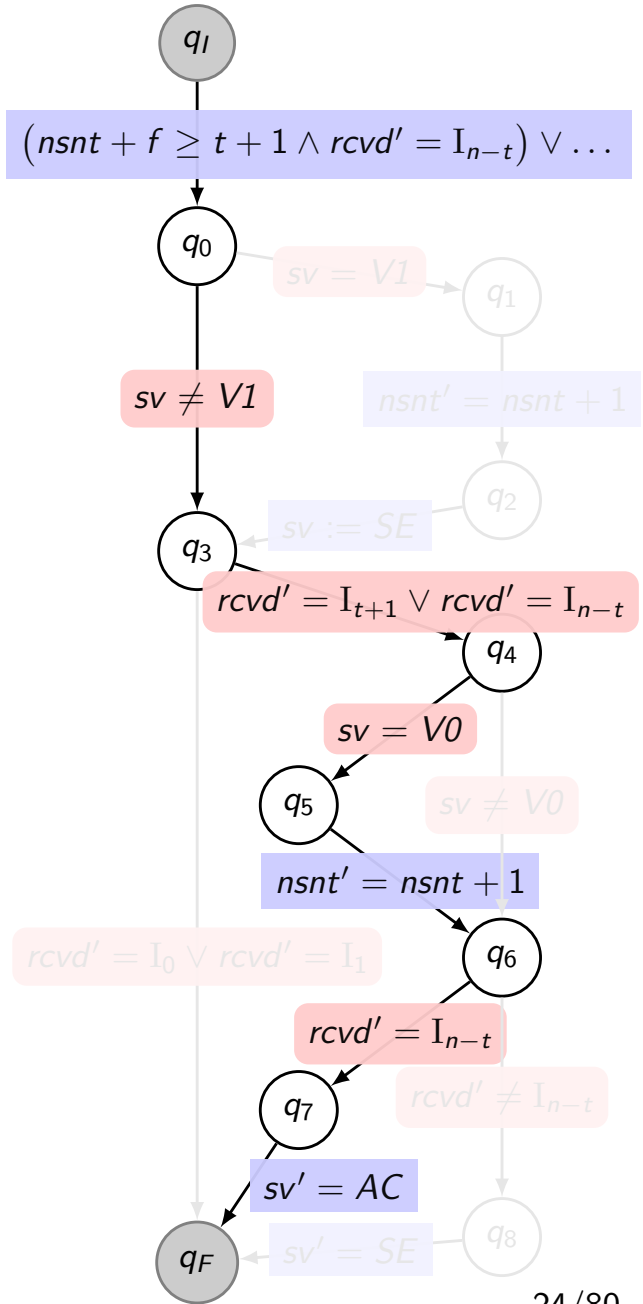
Intuition: from CFA to TA

How to construct TA from CFA?

- Apply parametric interval abstraction only to the local variables, e.g., *rcvd*
- Shared variables, e.g., *nsnt*, are still unbounded

- Enumerate all symbolic paths in CFA
- Use SMT to find all satisfying assignments of local variables
- Each of them gives a TA rule



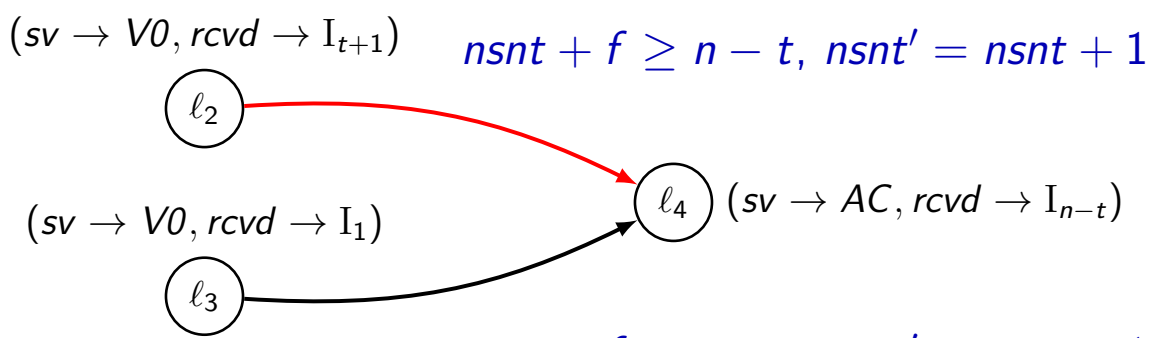


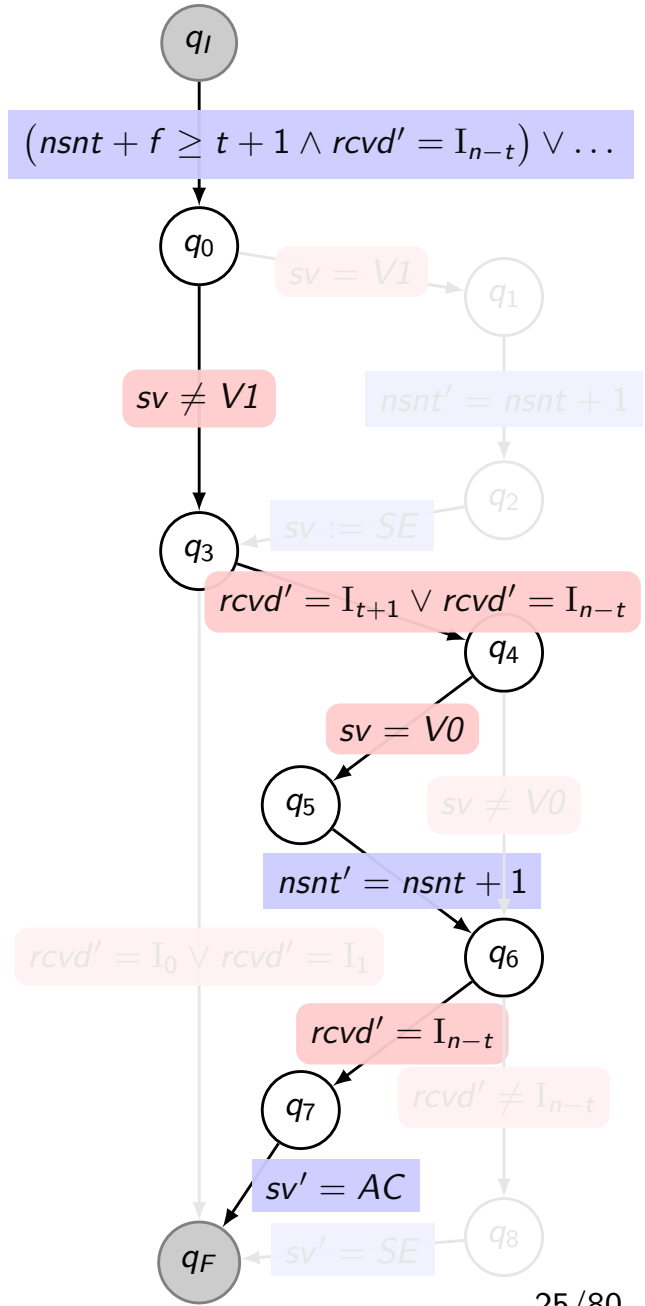
Intuition: from CFA to TA

How to construct TA from CFA?

- Apply parametric interval abstraction only to the local variables, e.g., *rcvd*
- Shared variables, e.g., *nsnt*, are still unbounded

- Enumerate all symbolic paths in CFA
- Use SMT to find all satisfying assignments of local variables
- Each of them gives a TA rule





Threshold Automaton of ST87 (after PIA data abstraction)

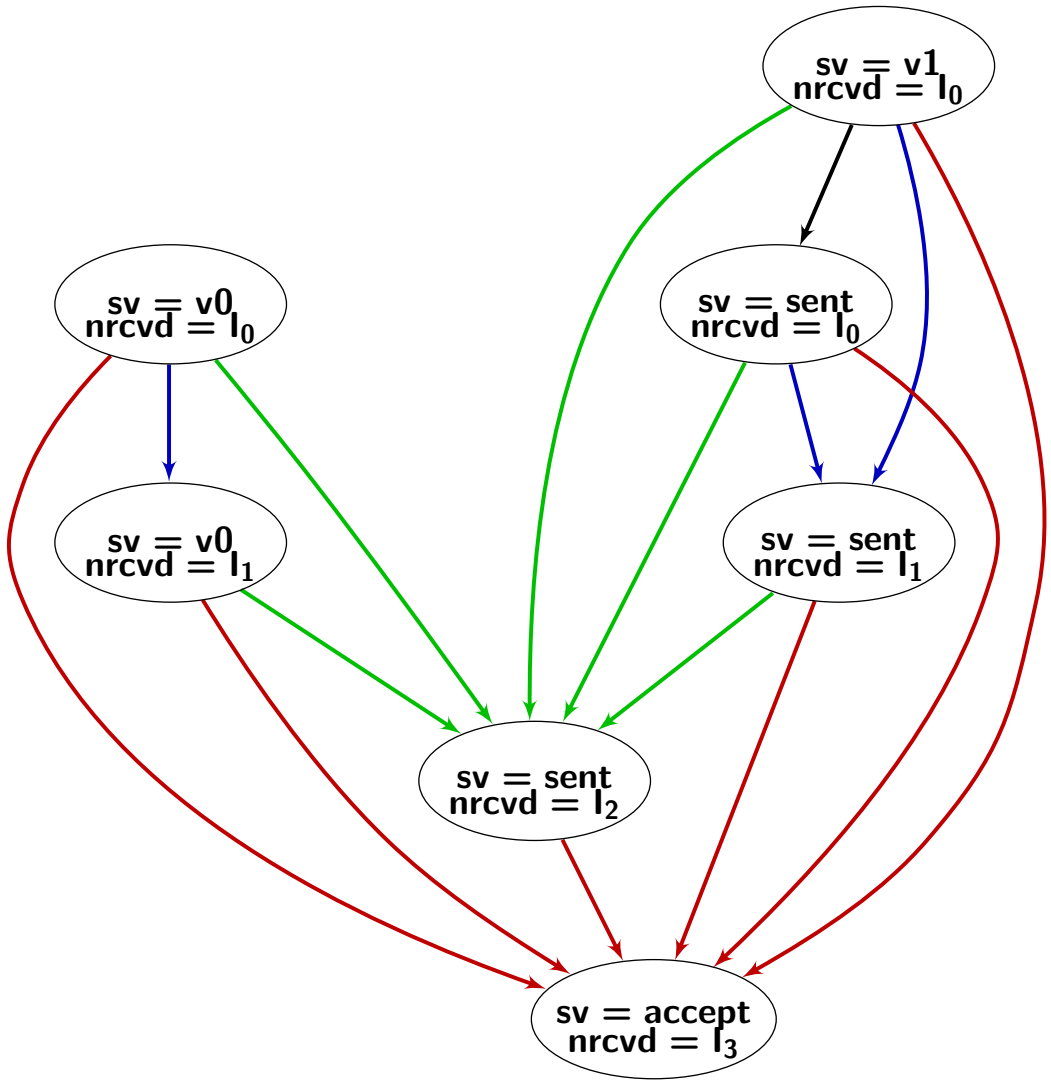
We automatically summarize process code from Part III:

7 locations, **15** rules
(+ self-loops)

Guards:

- black edges: *true*
- blue edges: $nsnt + f \geq 1$
- green edges: $nsnt + f \geq t + 1$
- red edges: $nsnt + f \geq n - t$

Actions increment *nsnt* iff:
$sv \in \{v0, v1\}$ to
$sv' \in \{sent, accept\}$



System of N processes

Having a threshold automaton P, fix:

- $\vec{p}$ are parameters satisfying the resilience condition $RC(\vec{p})$,
- $N(\vec{p})$ is a size function.

e.g., $\vec{p} = (n, t, f)$ and $N(\vec{p}) = n - f$ and $RC : n > 3t \wedge t \geq f \geq 0$.

and define a parallel composition $P(\vec{p})^{N(\vec{p})}$
(as a transition system with standard interleaving semantics).

However, we have a parameterized family of finite-state systems:

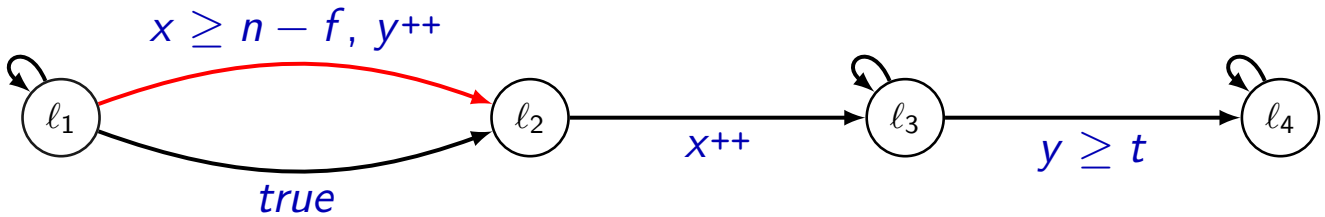
$$\{P(\vec{p})^{N(\vec{p})} \mid RC(\vec{p})\}$$

Counter system with acceleration!

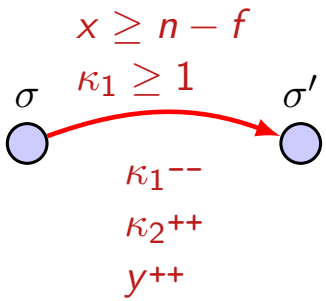
Counter system is a transition system simulating every system $P(\vec{p})^{N(\vec{p})}$.

Configuration $\sigma = (\vec{\kappa}, \vec{g}, \vec{p})$:

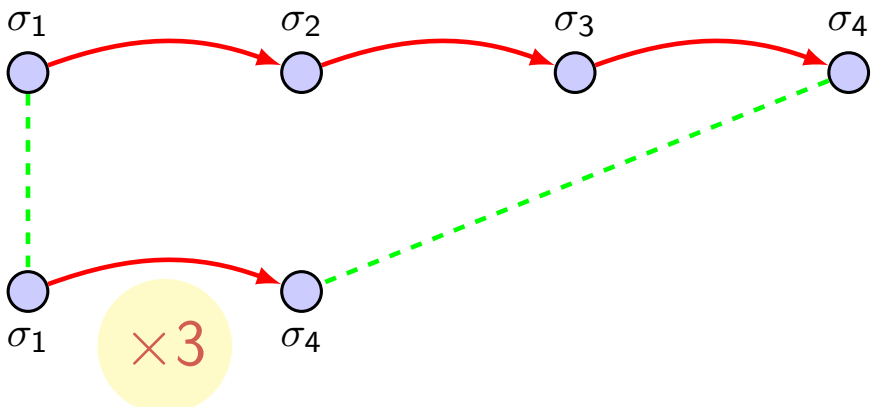
- κ_i counts processes at location ℓ_i with $\kappa_1 + \cdots + \kappa_{|L|} = N(\vec{p})$,
- g_j is the value of the shared variable x_j,
- $\vec{p}$ are the values of the parameters.



one transition (interleaving):



accelerated transition:



More formally: counter system

Counter system is a transition system that simulates every system $P^{N(\vec{p})}$.

Configuration $\sigma = (\vec{\kappa}, \vec{g}, \vec{p})$:

- κ_i counts processes at location ℓ_i,
- $\kappa_1 + \cdots + \kappa_{|L|} = N(\vec{p})$,
- g_j is the value of the shared variable x_j,
- $\vec{p}$ are the values of the parameters.

Transition from $\sigma = (\vec{\kappa}, \vec{g}, \vec{p})$ to $\sigma' = (\vec{\kappa}', \vec{g}', \vec{p})$:

there is an edge from ℓ to ℓ' labeled with condition φ and update vector $\vec{u}$:

- update counters: $\kappa_\ell \geq 1$ and $\kappa'_\ell = \kappa_\ell - 1$ and $\kappa'_{\ell'} = \kappa_{\ell'} + 1$
- check threshold condition: $\vec{g} \models \varphi$
- update shared variables: $\vec{g}' = \vec{g} + \vec{u}$
- the other counters κ_j stay unchanged

More formally: counter system with acceleration!

Counter system is a transition system that simulates every system $P^{N(\vec{p})}$.

Configuration $\sigma = (\vec{\kappa}, \vec{g}, \vec{p})$:

- κ_i counts processes at location ℓ_i,
- $\kappa_1 + \cdots + \kappa_{|L|} = N(\vec{p})$,
- g_j is the value of the shared variable x_j,
- $\vec{p}$ are the values of the parameters.

Transition from $\sigma = (\vec{\kappa}, \vec{g}, \vec{p})$ to $\sigma' = (\vec{\kappa}', \vec{g}', \vec{p})$ with factor $\delta \geq 1$:
there is an edge from ℓ to ℓ' labeled with condition φ and update vector $\vec{u}$:

- update counters: $\kappa_\ell \geq \delta$ and $\kappa'_\ell = \kappa_\ell - \delta$ and $\kappa'_{\ell'} = \kappa_{\ell'} + \delta$
- check threshold condition: $\vec{g} \models \varphi$ and $\vec{g} + (\delta - 1) \cdot \vec{u} \models \varphi$
- update shared variables: $\vec{g}' = \vec{g} + \delta \cdot \vec{u}$
- the other counters κ_j stay unchanged

Reachability and parameterized reachability

Reachability (fixed parameters):

Fix the parameters, e.g., $n = 4$, $t = 1$, $f = 1$, $N = n - f = 3$.

Fix configurations σ and σ' of P^N.

Question: is σ' reachable from σ in P^N?

Reachability and parameterized reachability

Reachability (fixed parameters):

Fix the parameters, e.g., $n = 4$, $t = 1$, $f = 1$, $N = n - f = 3$.

Fix configurations σ and σ' of P^N.

Question: is σ' reachable from σ in P^N?

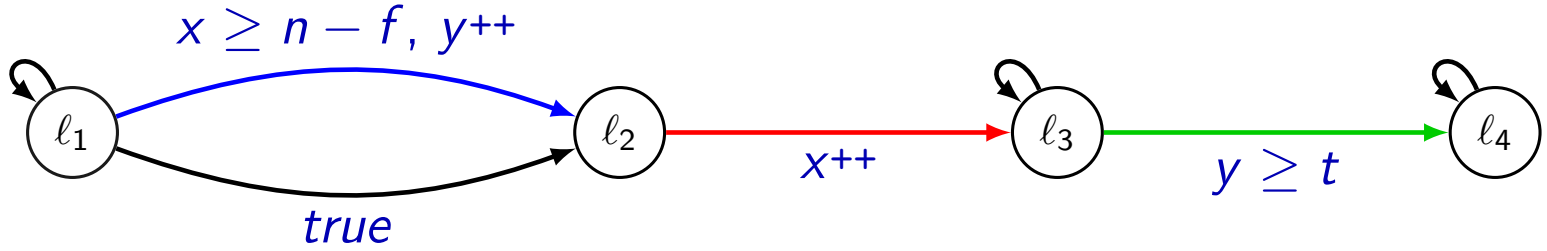
Parameterized reachability:

Fix properties S and S' on configurations,
e.g., $S : \kappa_1 = N(\vec{p})$ and $S' : \kappa_4 \neq 0$.

Question: are there parameter values $\vec{p}$ and configurations σ, σ' of $P^{N(\vec{p})}$:

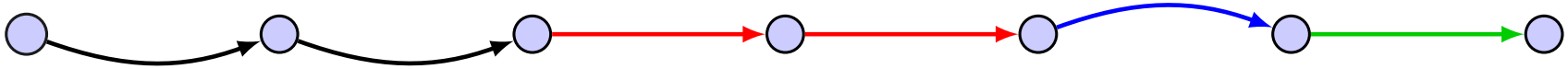
- parameters $\vec{p}$ satisfy the resilience condition $RC(\vec{p})$,
- $\sigma \models S$ and $\sigma' \models S'$,
- σ' is reachable from σ in $P^{N(\vec{p})}$.

Parameterized reachability: Example



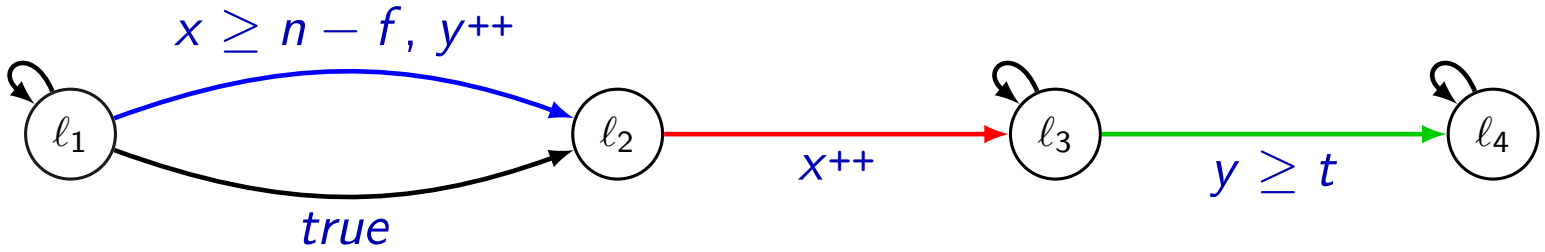
Resilience condition 1: $n > t \geq f$ and $t > 0$.

Is ℓ_4 reachable, if all processes start at ℓ_1? YES



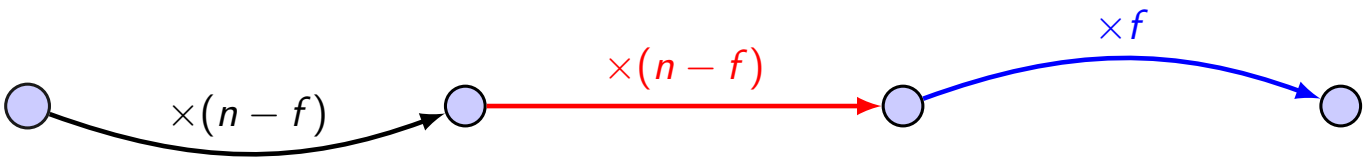
$\kappa_1 = 3$	$\kappa_1 = 1$	$\kappa_1 = 1$	$\kappa_1 = 0$	$\kappa_1 = 0$
$\kappa_2 = 0$	$\kappa_2 = 2$	$\kappa_2 = 0$	$\kappa_2 = 1$	$\kappa_2 = 1$
$\kappa_3 = 0$	$\kappa_3 = 0$	$\kappa_3 = 2$	$\kappa_3 = 2$	$\kappa_3 = 1$
$\kappa_4 = 0$	$\kappa_4 = 0$	$\kappa_4 = 0$	$\kappa_4 = 0$	$\kappa_4 = 1$
$x = 0$	$x = 0$	$x = 2$	$x = 2$	$x = 2$
$y = 0$	$y = 0$	$y = 0$	$y = 1$	$y = 1$

Parameterized reachability: Example 2



Resilience condition 2: $n > t > f$ and $t > 0$.

Is ℓ_4 reachable, if all processes start at ℓ_1? NO



$\kappa_1 = n$	$\kappa_1 = f$	$\kappa_1 = f$	$\kappa_1 = 0$
$\kappa_2 = 0$	$\kappa_2 = n - f$	$\kappa_2 = 0$	$\kappa_2 = 0$
$\kappa_3 = 0$	$\kappa_3 = 0$	$\kappa_3 = n - f$	$\kappa_3 = n$
$\kappa_4 = 0$	$\kappa_4 = 0$	$\kappa_4 = 0$	$\kappa_4 = 0$
$x = 0$	$x = 0$	$x = n - f$	$x = n - f$
$y = 0$	$y = 0$	$y = 0$	$y = f$

Parameterized & bounded model checking

Bounded Model Checking

Model checking without BDDs [Biere, Cimatti, Clarke'99]

Bounded Model Checking

Model checking without BDDs [Biere, Cimatti, Clarke'99]

Encode as a boolean formula:

- the transition relation $T(\overline{x}, \overline{x}')$,
- the set of initial states $I(\overline{x})$,
- the set of bad states $B(\overline{x})$.

Given a bound k,
construct a model checking problem for paths of length k:

$$f_k \equiv I(\vec{x}_0) \wedge T(\vec{x}_0, \vec{x}_1) \wedge T(\vec{x}_1, \vec{x}_2) \cdots \wedge T(\vec{x}_{k-1}, \vec{x}_k) \wedge B(\vec{x}_k)$$

Bounded Model Checking

Model checking without BDDs [Biere, Cimatti, Clarke'99]

Encode as a boolean formula:

- the transition relation $T(\overline{x}, \overline{x}')$,
- the set of initial states $I(\overline{x})$,
- the set of bad states $B(\overline{x})$.

Given a bound k,
construct a model checking problem for paths of length k:

$$f_k \equiv I(\vec{x}_0) \wedge T(\vec{x}_0, \vec{x}_1) \wedge T(\vec{x}_1, \vec{x}_2) \cdots \wedge T(\vec{x}_{k-1}, \vec{x}_k) \wedge B(\vec{x}_k)$$

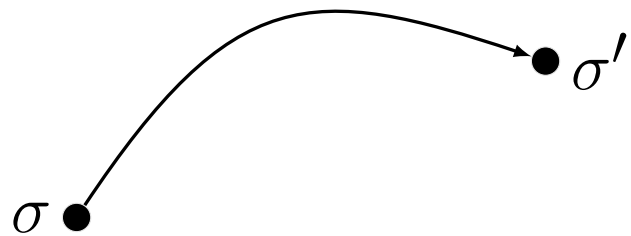
Check f_k with a SAT solver.

Tools that implement BMC: NuSMV, CBMC, and many other.

Diameter of a system

Consider configurations σ and σ'

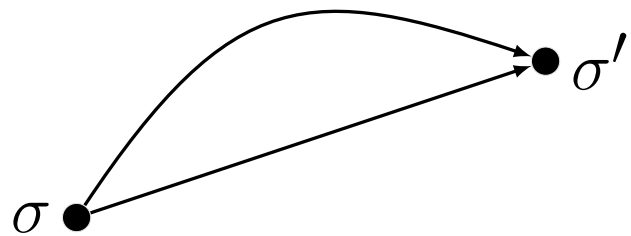
- if σ' is reachable from σ



Diameter of a system

Consider configurations σ and σ'

- if σ' is reachable from σ
- then distance $dist(\sigma, \sigma')$ is the length of the shortest path from σ to σ'

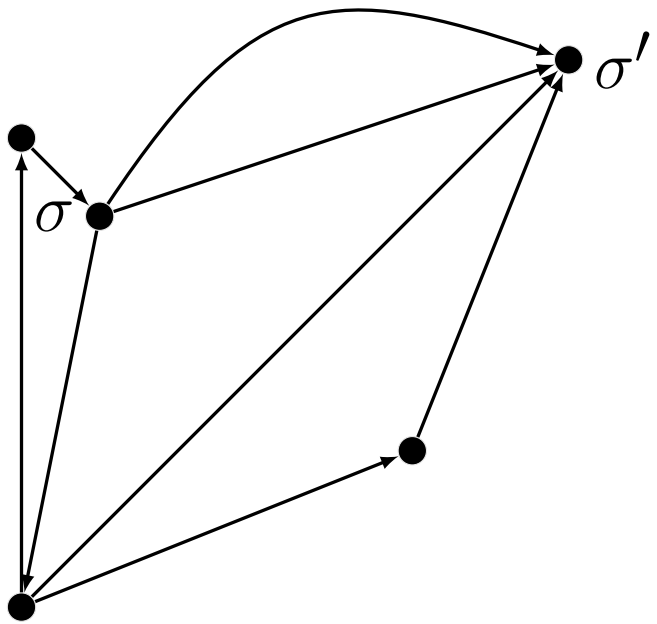


Diameter of a system

Consider configurations σ and σ'

- if σ' is reachable from σ
- then distance $dist(\sigma, \sigma')$ is the length of the shortest path from σ to σ'

Consider distances between
all pairs of states



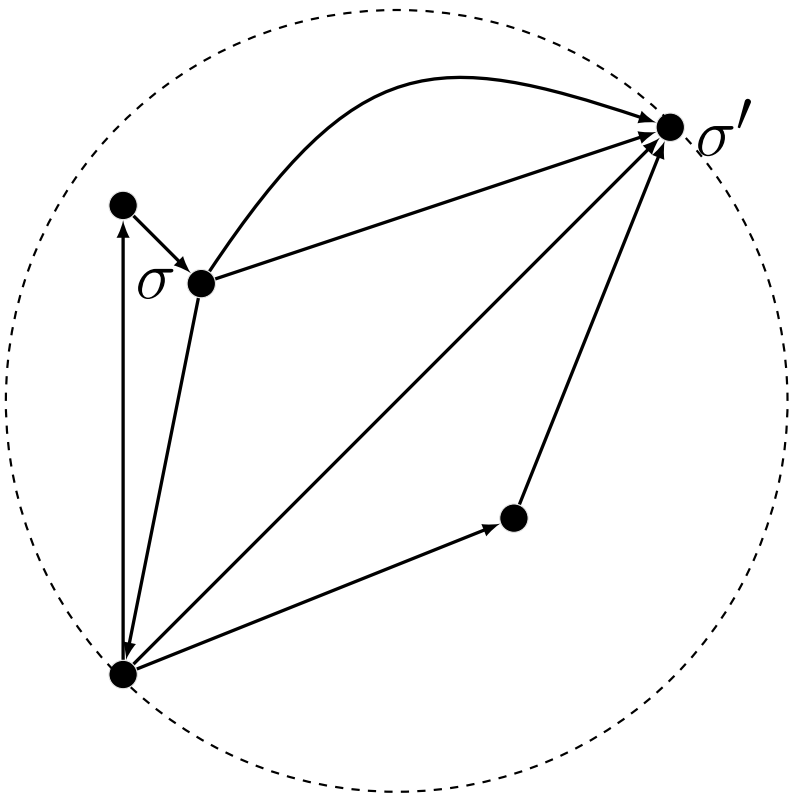
Diameter of a system

Consider configurations σ and σ'

- if σ' is reachable from σ
- then distance $dist(\sigma, \sigma')$ is the length of the shortest path from σ to σ'

Consider distances between all pairs of states

The diameter is the longest distance among all pairs of states



Complete bounded model checking (reachability)

Bounded model checking explores executions up to a given length k.

To make it complete for reachability properties,
set k to the diameter of the transition system [Biere, Cimatti, Clarke'99]

If we know the diameter d of the accelerated counter system,
then for every combination of the parameters $\vec{p}$,

$$\text{diameter of unaccelerated } P^{N(\vec{p})} \leq d \cdot N(\vec{p})$$

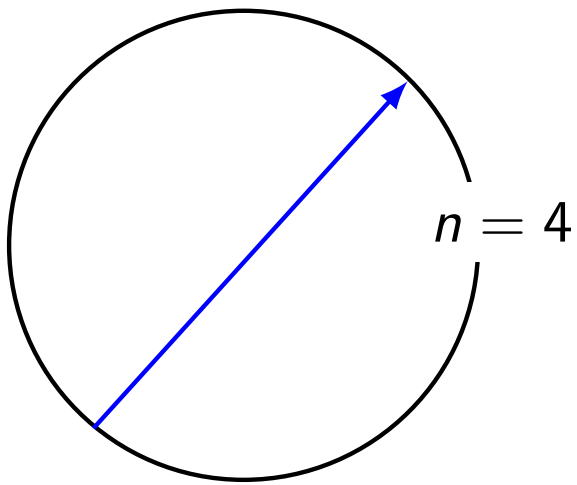
Diameter is the greatest distance between any pair of configurations.
Distance between two configurations is the length of the shortest path.

Diameter of a fixed-size system

Fix the parameters, e.g., $n = 4, t = 1, f = 1$.

All variables are bounded, the state set is finite.

The diameter is bounded by the number of states.



Diameter of a fixed-size system

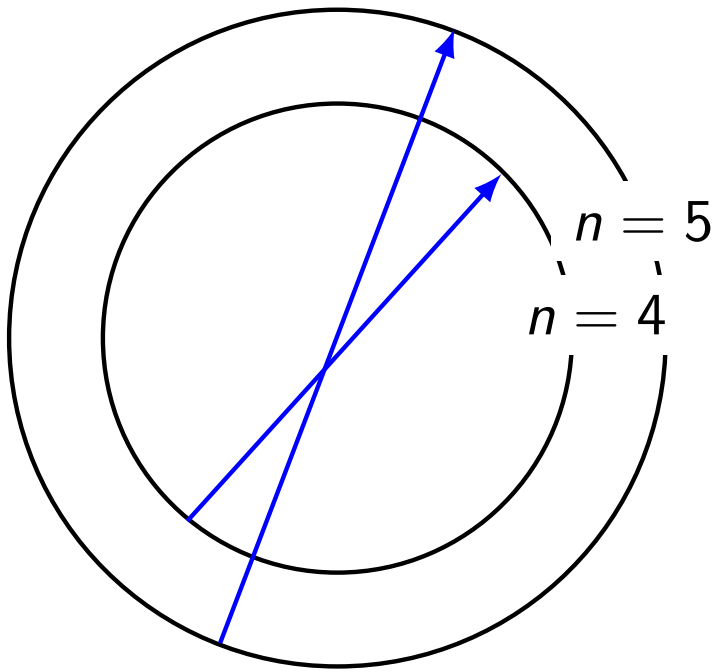
Fix the parameters, e.g., $n = 4, t = 1, f = 1$.

All variables are bounded, the state set is finite.

The diameter is bounded by the number of states.

Increase the system size

The diameter grows...



Diameter of a fixed-size system

Fix the parameters, e.g., $n = 4, t = 1, f = 1$.

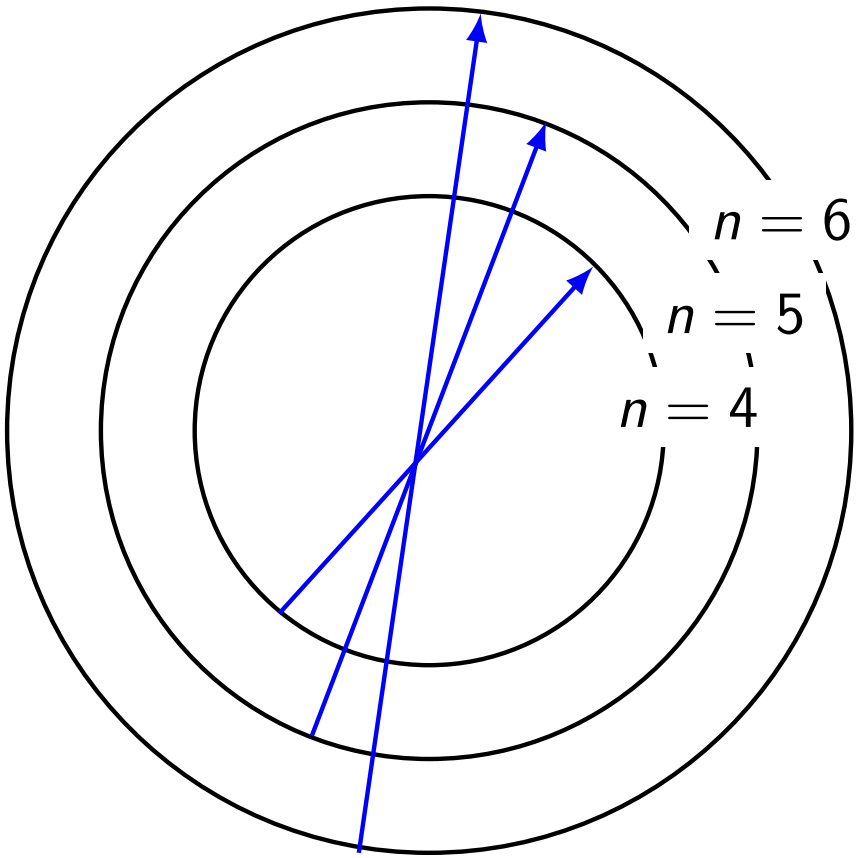
All variables are bounded, the state set is finite.

The diameter is bounded by the number of states.

Increase the system size

The diameter grows...

Can acceleration help?



Diameter of a fixed-size system

Fix the parameters, e.g., $n = 4, t = 1, f = 1$.

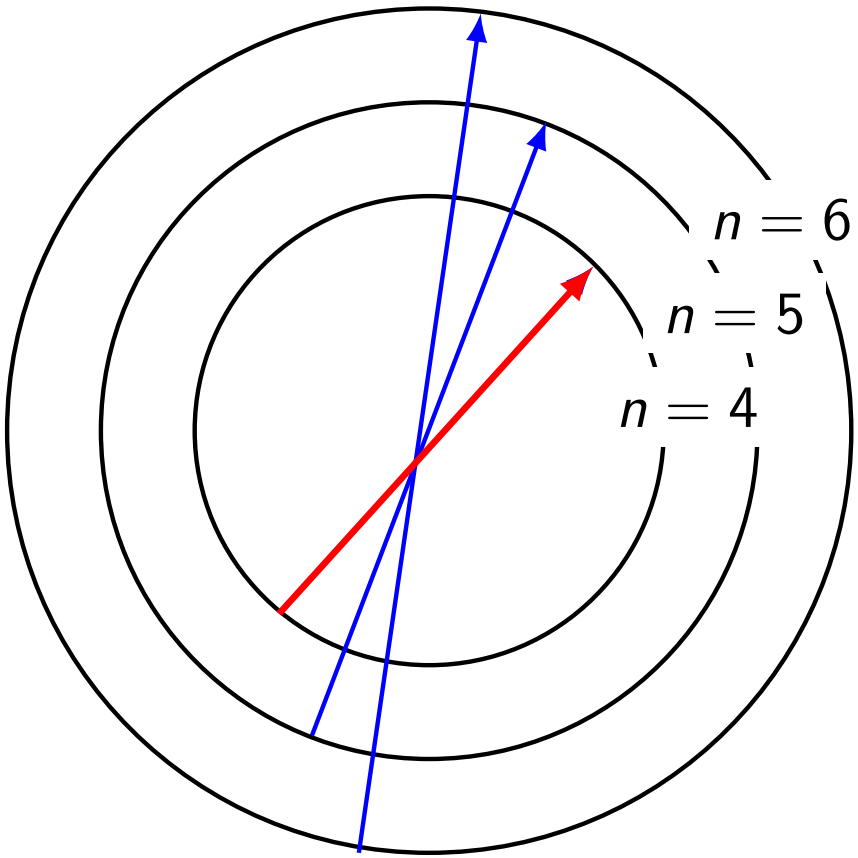
All variables are bounded, the state set is finite.

The diameter is bounded by the number of states.

Increase the system size

The diameter grows...

Can acceleration help?



Diameter of a fixed-size system

Fix the parameters, e.g., $n = 4, t = 1, f = 1$.

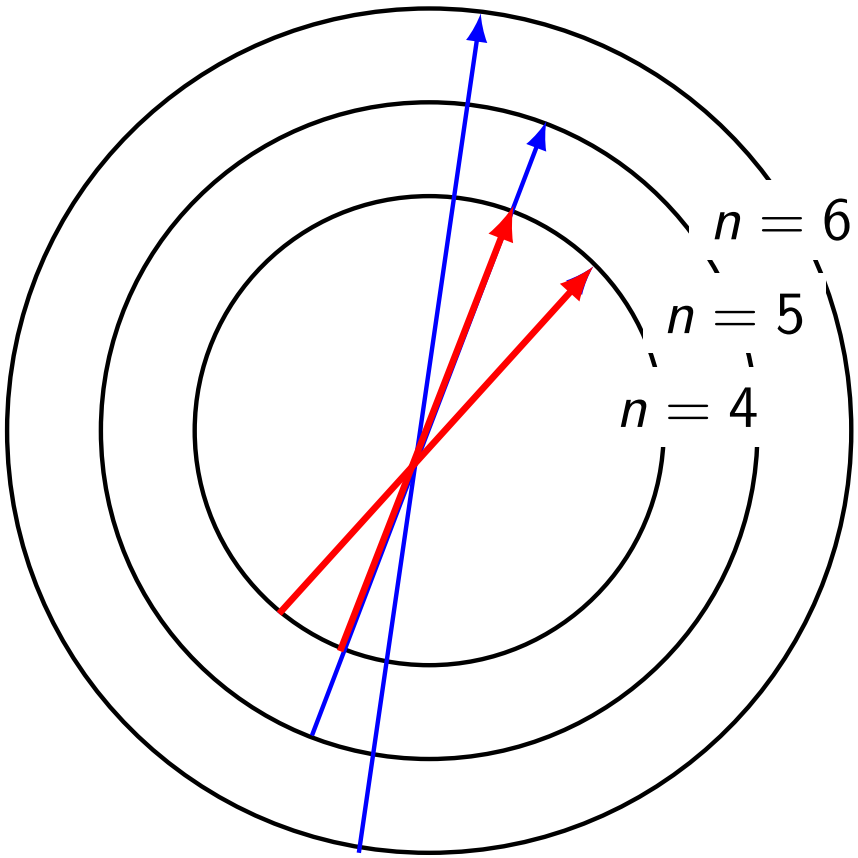
All variables are bounded, the state set is finite.

The diameter is bounded by the number of states.

Increase the system size

The diameter grows...

Can acceleration help?



Diameter of a fixed-size system

Fix the parameters, e.g., $n = 4, t = 1, f = 1$.

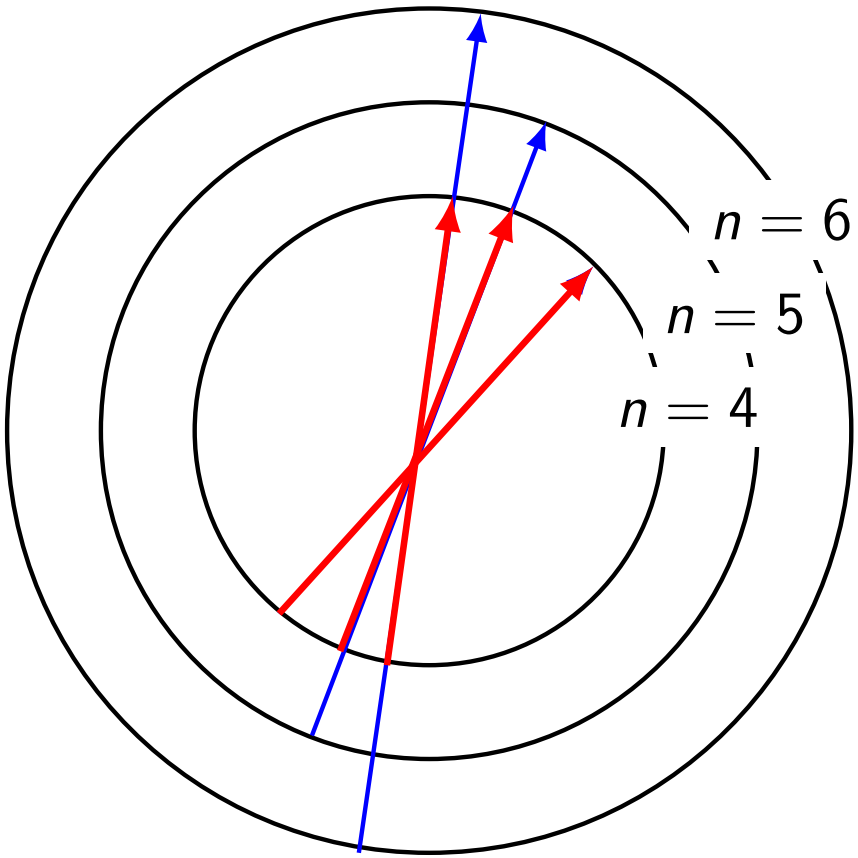
All variables are bounded, the state set is finite.

The diameter is bounded by the number of states.

Increase the system size

The diameter grows...

Can acceleration help?

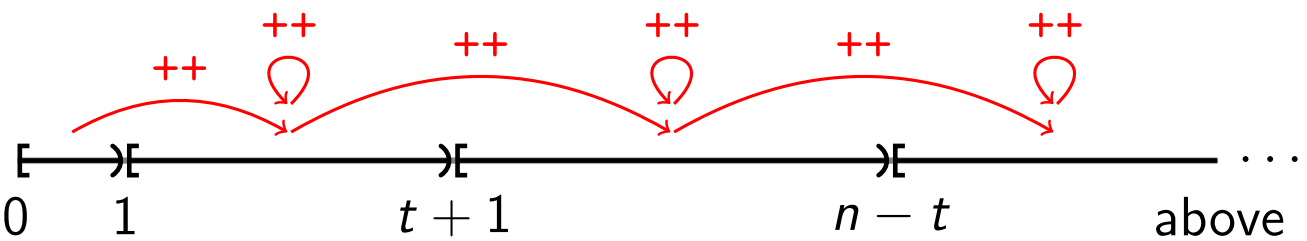


Complete parameterized bounded model checking

Use counter abstraction to get a finite system $\mathcal{A}$.

Counters κ_i are mapped to a finite domain $\widehat{D}$, e.g.,

- $\{0, 1, \infty\}$ by [Pnueli, Xu, Zuck'02].
- Domain of parametric intervals extracted from thresholds,
  e.g., $\{[0, 1), [1, t+1), [t+1, n-t), [n-t, \infty)\}$, see [FMCAD'13].



If we know the diameter d of the accelerated counter system, then

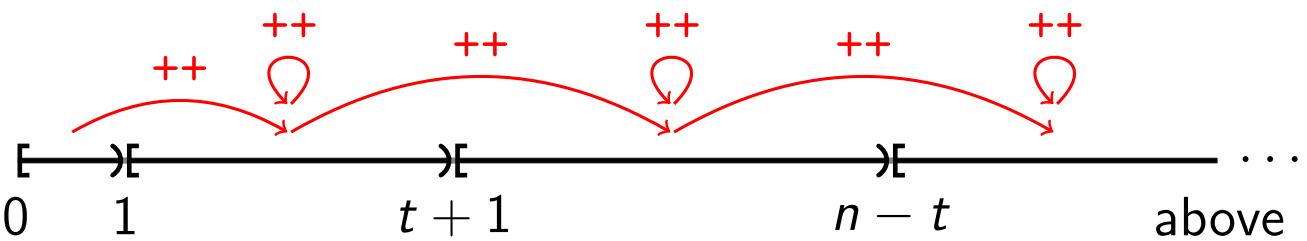
$$diam(\mathcal{A}) \leq d \cdot (|\widehat{D}| - 1)$$

Complete parameterized bounded model checking

Use counter abstraction to get a finite system $\mathcal{A}$.

Counters κ_i are mapped to a finite domain $\widehat{D}$, e.g.,

- $\{0, 1, \infty\}$ by [Pnueli, Xu, Zuck'02].
- Domain of parametric intervals extracted from thresholds,
  e.g., $\{[0, 1), [1, t+1), [t+1, n-t), [n-t, \infty)\}$, see [FMCAD'13].



If we know the diameter d of the accelerated counter system, then

$$diam(\mathcal{A}) \leq d \cdot (|\widehat{D}| - 1)$$

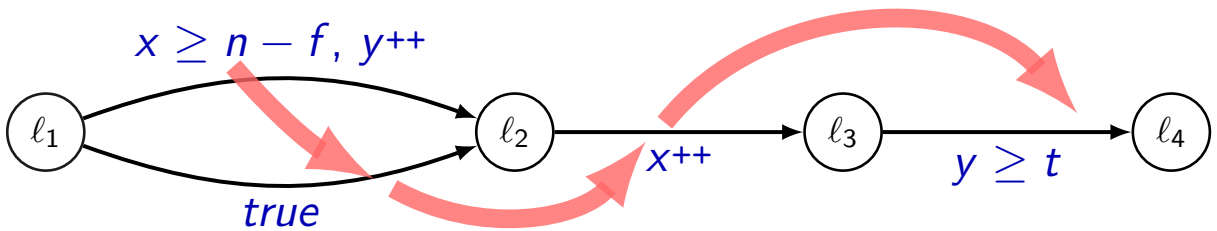
Warning: completeness may require abstraction refinement

The diameter of the accelerated system?

Partial orders on TA rules

The control flow defines a partial order.

Fix a total order $\preceq_P^{lin} \subseteq E \times E$ on the edges (rules):

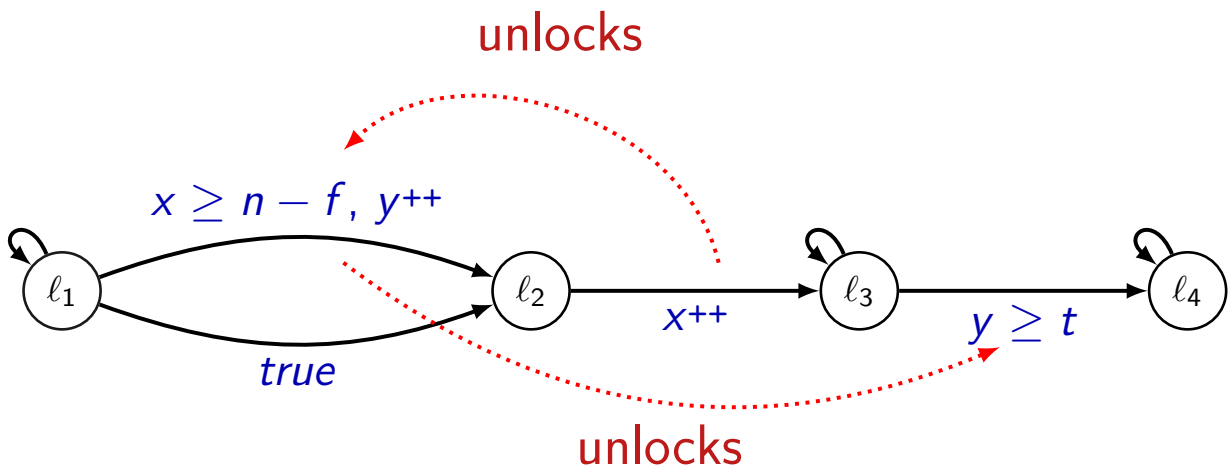


Partial orders on TA rules (cont.)

Define a partial order $\preceq_U \subseteq E \times E$ on the edges (rules):

$r_1 \prec_U r_2$ iff there is
a vector of shared variables $\vec{g} \in \mathbb{N}_0^{|\Gamma|}$ and parameter values $\mathbf{p} \in \mathbf{P}_{RC}$ with:

- $(\vec{g}, \mathbf{p}) \models r_1.\varphi$
- $(\vec{g}, \mathbf{p}) \not\models r_2.\varphi$
- $(\vec{g} + r_1.\vec{u}, \mathbf{p}) \models r_2.\varphi$

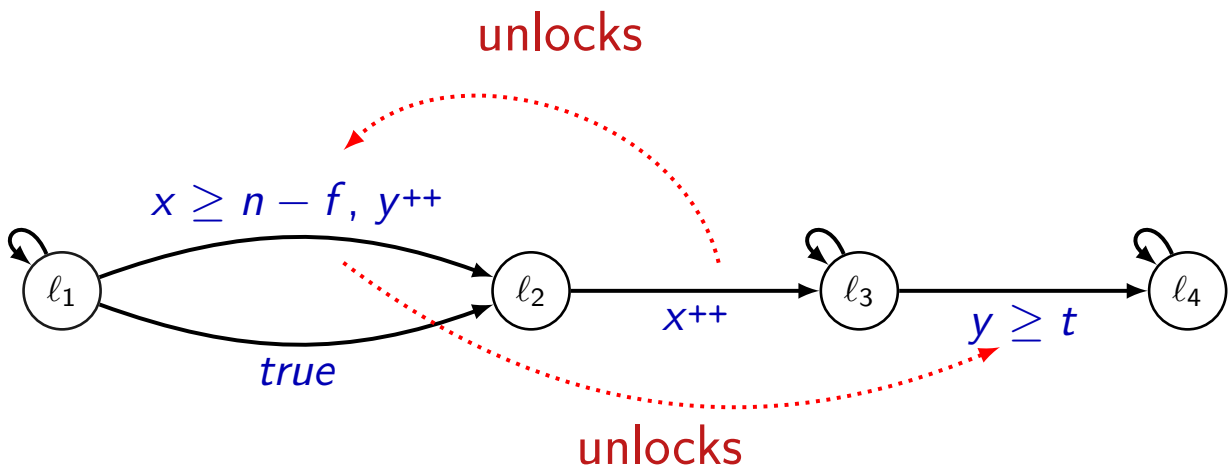


Partial orders on TA rules (cont.)

Define a partial order $\preceq_U \subseteq E \times E$ on the edges (rules):

$r_1 \prec_U r_2$ iff there is
a vector of shared variables $\vec{g} \in \mathbb{N}_0^{|\Gamma|}$ and parameter values $\mathbf{p} \in \mathbf{P}_{RC}$ with:

- $(\vec{g}, \mathbf{p}) \models r_1.\varphi$
- $(\vec{g}, \mathbf{p}) \not\models r_2.\varphi$
- $(\vec{g} + r_1.\vec{u}, \mathbf{p}) \models r_2.\varphi$



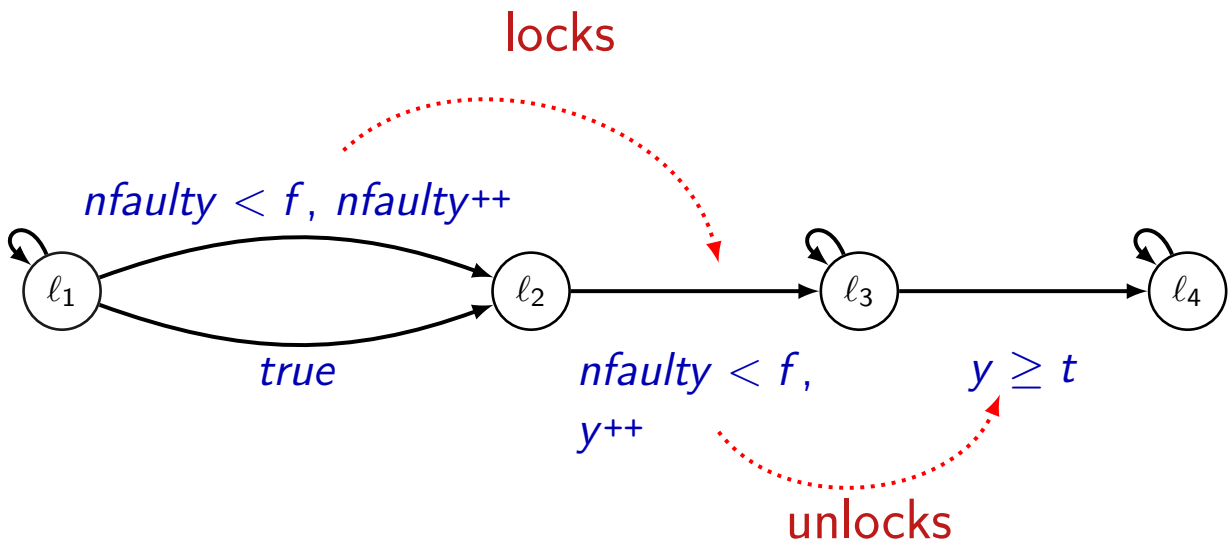
We can check the conditions with SMT

Partial orders on TA rules (cont.)

Define a partial order $\preceq_L \subseteq E \times E$ on the edges (rules):

$r_1 \prec_L r_2$ iff there is
a vector of shared variables $\vec{g} \in \mathbb{N}_0^{|\Gamma|}$ and parameter values $\mathbf{p} \in \mathbf{P}_{RC}$ with:

- $(\vec{g}, \mathbf{p}) \models r_1.\varphi$
- $(\vec{g}, \mathbf{p}) \models r_2.\varphi$
- $(\vec{g} + r_1.\vec{u}, \mathbf{p}) \not\models r_2.\varphi$



Our main result

Fix a threshold automaton TA and a size function N.

Theorem

For each $\vec{p}$ with $RC(\vec{p})$, the diameter of an accelerated counter system is independent of parameters and is less than or equal to $|E| \cdot (|\mathcal{C}| + 1) + |\mathcal{C}|$:

- $|E|$ is the number of edges in TA (self-loops excluded).
- $|\mathcal{C}|$ is the number of edge conditions in TA that can be unlocked (locked) by an edge appearing later (resp. earlier) in the control flow, or by a parallel edge.

Our main result

Fix a threshold automaton TA and a size function N.

Theorem

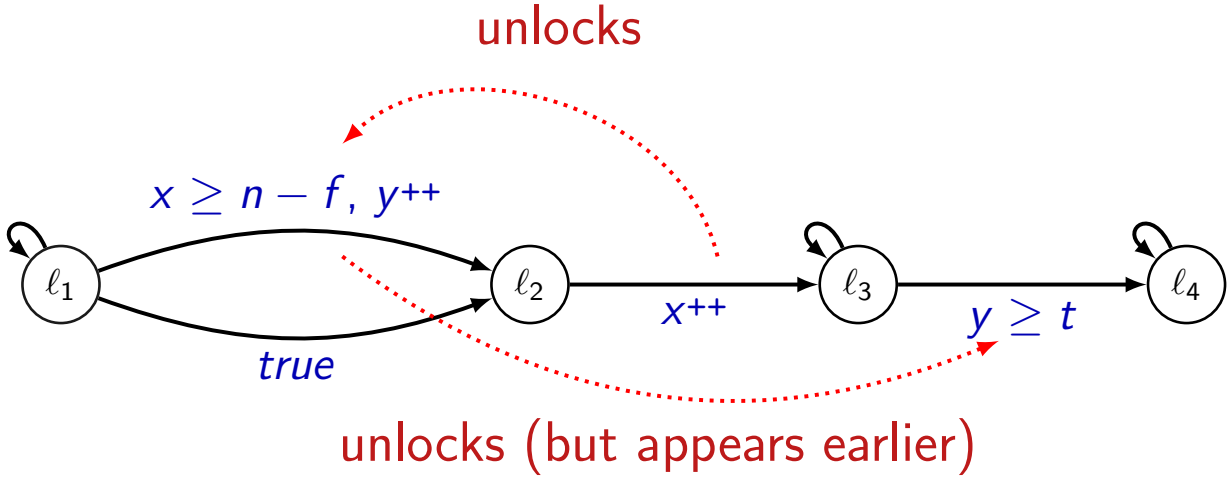
For each $\vec{p}$ with $RC(\vec{p})$, the diameter of an accelerated counter system is independent of parameters and is less than or equal to $|E| \cdot (|\mathcal{C}| + 1) + |\mathcal{C}|$:

- $|E|$ is the number of edges in TA (self-loops excluded).
- $|\mathcal{C}|$ is the number of edge conditions in TA that can be unlocked (locked) by an edge appearing later (resp. earlier) in the control flow, or by a parallel edge.

In our example:

$|E| = 4$, $|\mathcal{C}| = 1$.

Thus, $d \leq 9$.



Proof idea

Central idea

For each run that connects two configurations we construct a short run by:

- swapping transitions,
- and accelerating them

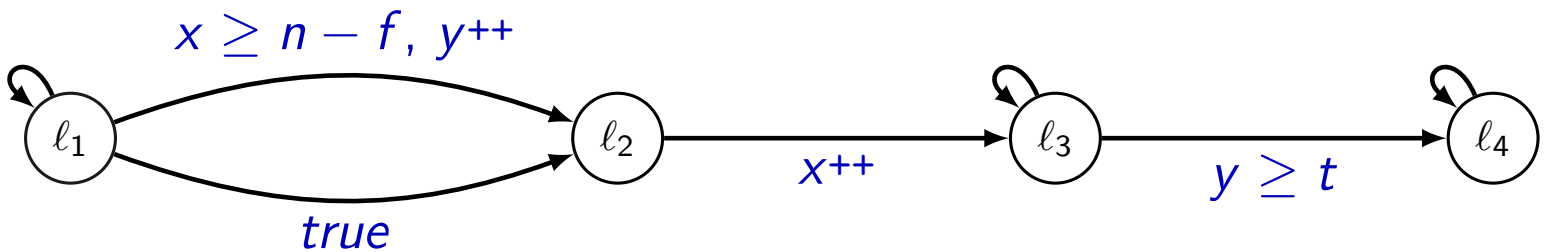
Central idea

For each run that connects two configurations
we construct a short run by:

- swapping transitions,
- and accelerating them

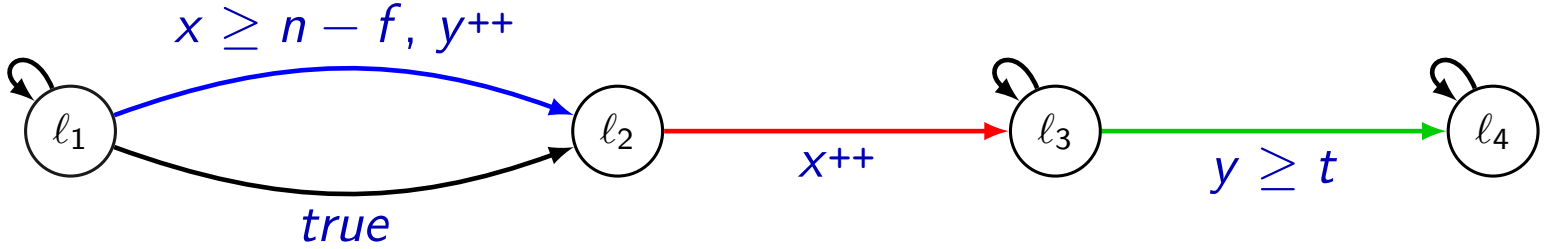
Shared variables are only incremented.

Valuation of each comparison changes at most once along every execution.

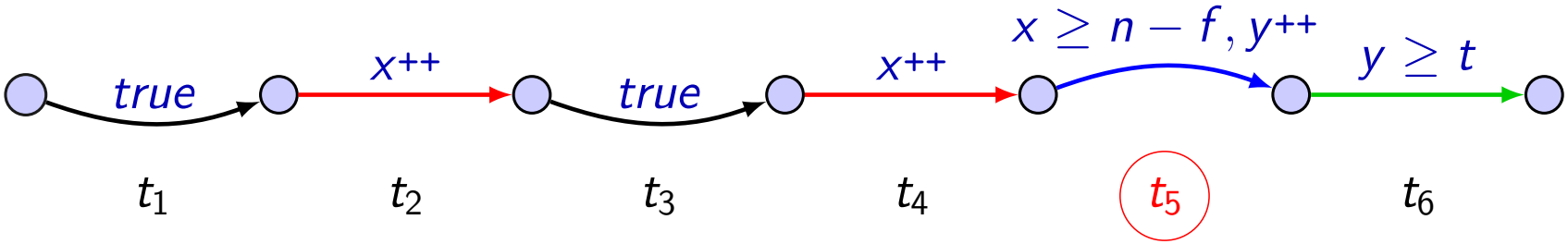


E.g., once $x \geq n - f$ and $y \geq t$ hold true, they will remain true.

Milestones



Consider an execution for $n = 3$, $t = 1$, $f = 1$:

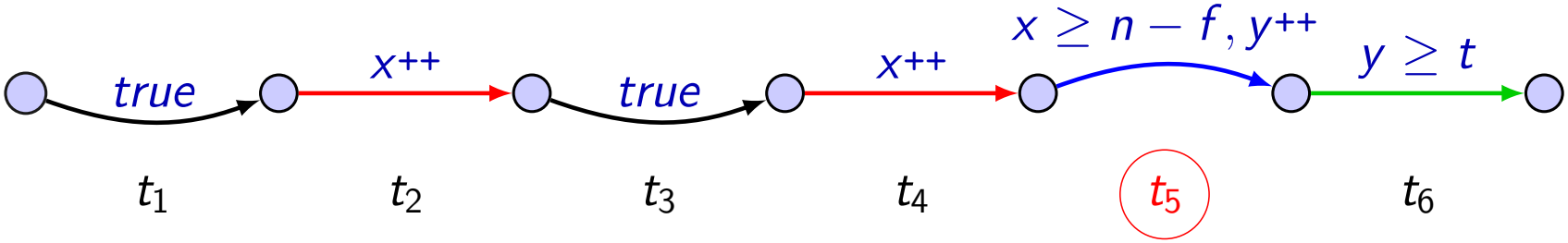


Transition t_5 is a milestone (and t_6 is not):

- its condition is unlocked by t_4, i.e., $t_4 \prec_U t_5$
- the rule of t_5 precedes the edge of t_4 in the control flow, i.e., $t_5 \prec_P^+ t_4$

Observation: a milestone cannot be swapped with any other transition.

Milestones formally



Transition t_5 is a left milestone.

Definition (Left Milestone)

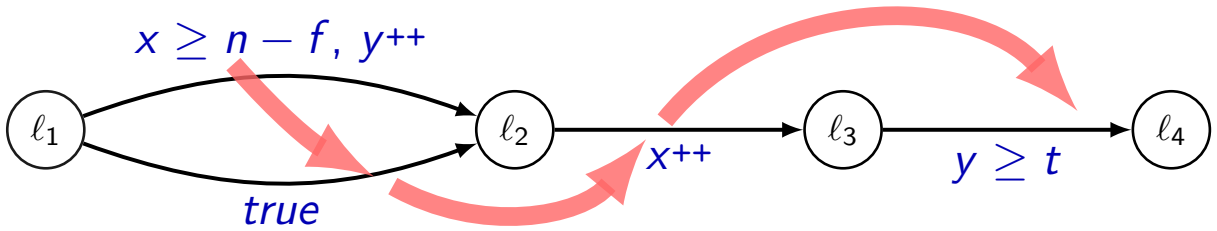
Given a configuration σ and a sequence of rules $\tau = \tau' \cdot t \cdot \tau''$ applicable to σ, the transition t is a left milestone for σ and τ, if

1. there is a transition t' in τ' satisfying $t' \not\leftarrow_P^+ t \ \wedge \ t' \prec_U t$,
2. $t.\varphi^{\leq}$ is locked in σ, and
3. for all t' in τ', $t'.\varphi^{\leq} \neq t.\varphi^{\leq}$.

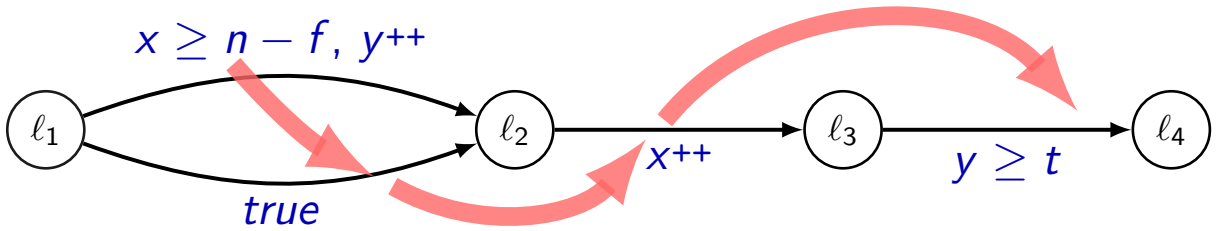
($t.\varphi^{\leq}$ is a conjunction of conditions like $x \geq a_0 \cdot n + a_1 \cdot t$))

(A right milestone is defined similarly w.r.t. $\prec_L$.)

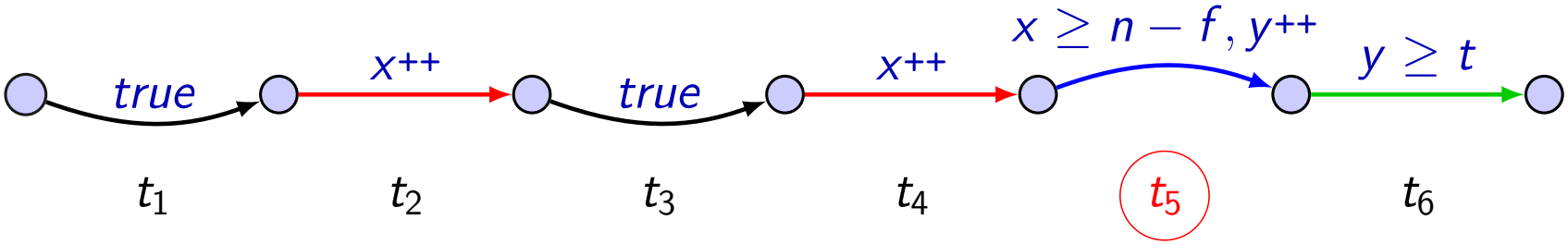
Sorting the transitions (with acyclic control flow)



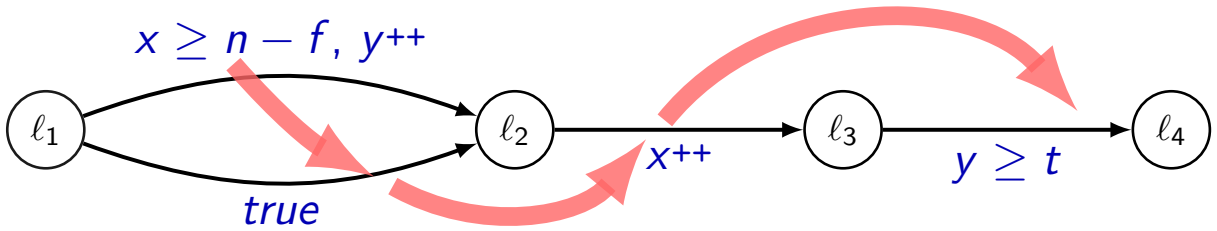
Sorting the transitions (with acyclic control flow)



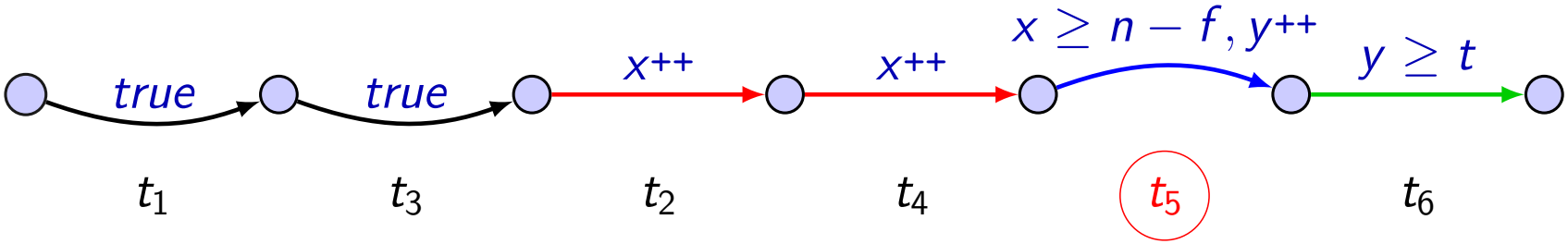
Sort the transitions between the milestones:



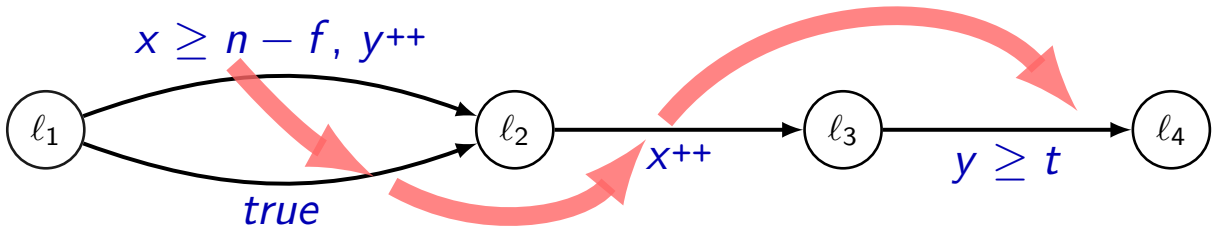
Sorting the transitions (with acyclic control flow)



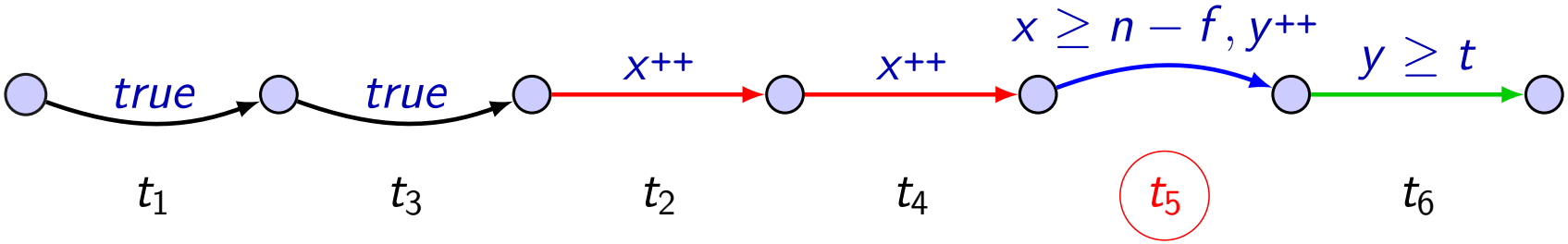
Sort the transitions between the milestones:



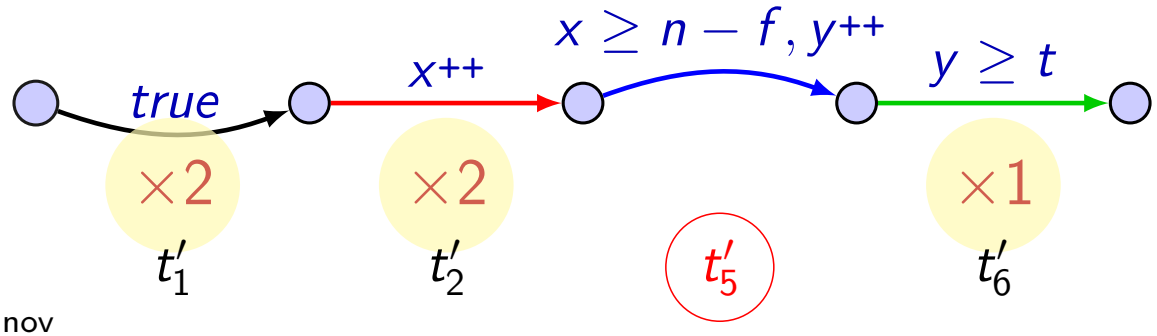
Sorting the transitions (with acyclic control flow)



Sort the transitions between the milestones:



Accelerate adjacent transitions of the same type:



How long is an accelerated execution?

The number of milestones is bounded with $|\mathcal{C}|$:

> the number of edge conditions that can be unlocked (locked) by an edge appearing later (resp. earlier) in the control flow, or by a parallel edge.

The length of each segment (sorted and accelerated) is bounded with $|E|$:

> the number of edges in the threshold automaton

The length of an accelerated execution is bounded with:

$$\underbrace{|E|}_{\text{length of each segment}} \times \underbrace{(|\mathcal{C}|+1)}_{\text{number of segments}} + \underbrace{|\mathcal{C}|}_{\text{number of milestones}}$$

So is the diameter of the accelerated counter system.

Evaluation

Case studies: asynchronous threshold-based FTDAs

- Toy example (Toy) [we made it up]

  5 locations, **8** rules

- Folklore reliable broadcast (FRB) [Chandra, Toueg'96]

  6 locations, **15** rules

- Consistent broadcast (STRB) [Srikanth, Toueg'87]

  7 locations, **21** rule

- Byzantine agreement (ABA) [Bracha, Toueg'85]

  case 1: **37** counters, **202** rules; case 2: **61** locations, **425** rules

- Condition-based consensus (CBC)

  [Mostefaoui, Nourgaya, Parvedy, Raynal'03]

  case 1: **71** counter, **408** rules; case 2: **115** counters and **991** rule

- Non-blocking atomic commitment (NBAC and NBACC)

  [Raynal'97], [Guerraoui'02]

  case 1: **77** counters, **1356** rules; case 2: **109** counters, **1831** rule

Case studies: asynchronous threshold-based FTDAs

- Toy example (Toy).

  [we made it up]

- Folklore reliable broadcast (FRB).

  [Chandra, Toueg'96]

- Consistent broadcast (STRB).

  [Srikanth, Toueg'87]

- Byzantine agreement (ABA).

  [Bracha, Toueg'85]

- Condition-based consensus (CBC).

  [Mostefaoui, Nourgaya, Parvedy, Raynal'03]

- Non-blocking atomic commitment (NBAC and NBACC).

  [Raynal'97], [Guerraoui'02]

Implementation

We encode the distributed algorithms in parameteric PROMELA

Our tool BYMC implements counter abstraction/refinement loop

NUSMV does bounded model checking of the counter abstraction:

- either with MINISAT,
- or PLINGELING (multicore SAT solver)

Everything is available at: [http://forsyte.at/software/bymc]

Can we reach the bound with NuSMV?

Timeout in abstraction refinement: NBAC (13200) and NBACC (16500).

Conclusions for Part V

Polynomial bound on the diameter of accelerated counter systems
(for threshold automata)

Our results allow us to use bounded model checking as a complete method for reachability in systems of threshold automata of:

- a fixed size,
- a parameterized size

Conclusions for Part V

Polynomial bound on the diameter of accelerated counter systems
(for threshold automata)

Our results allow us to use bounded model checking as a complete method for reachability in systems of threshold automata of:

- a fixed size,
- a parameterized size

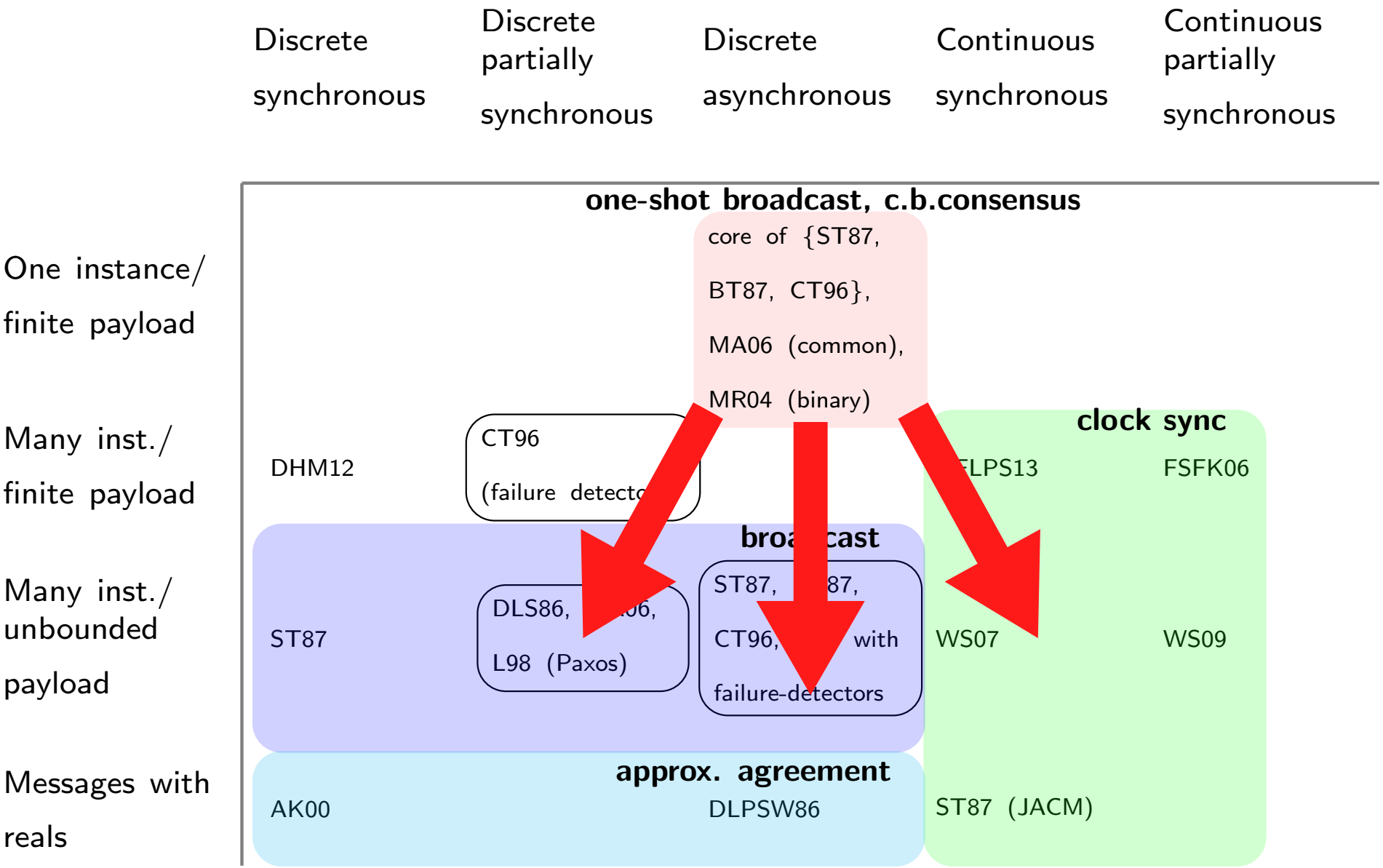
Bounds for liveness properties?

Better implementation?

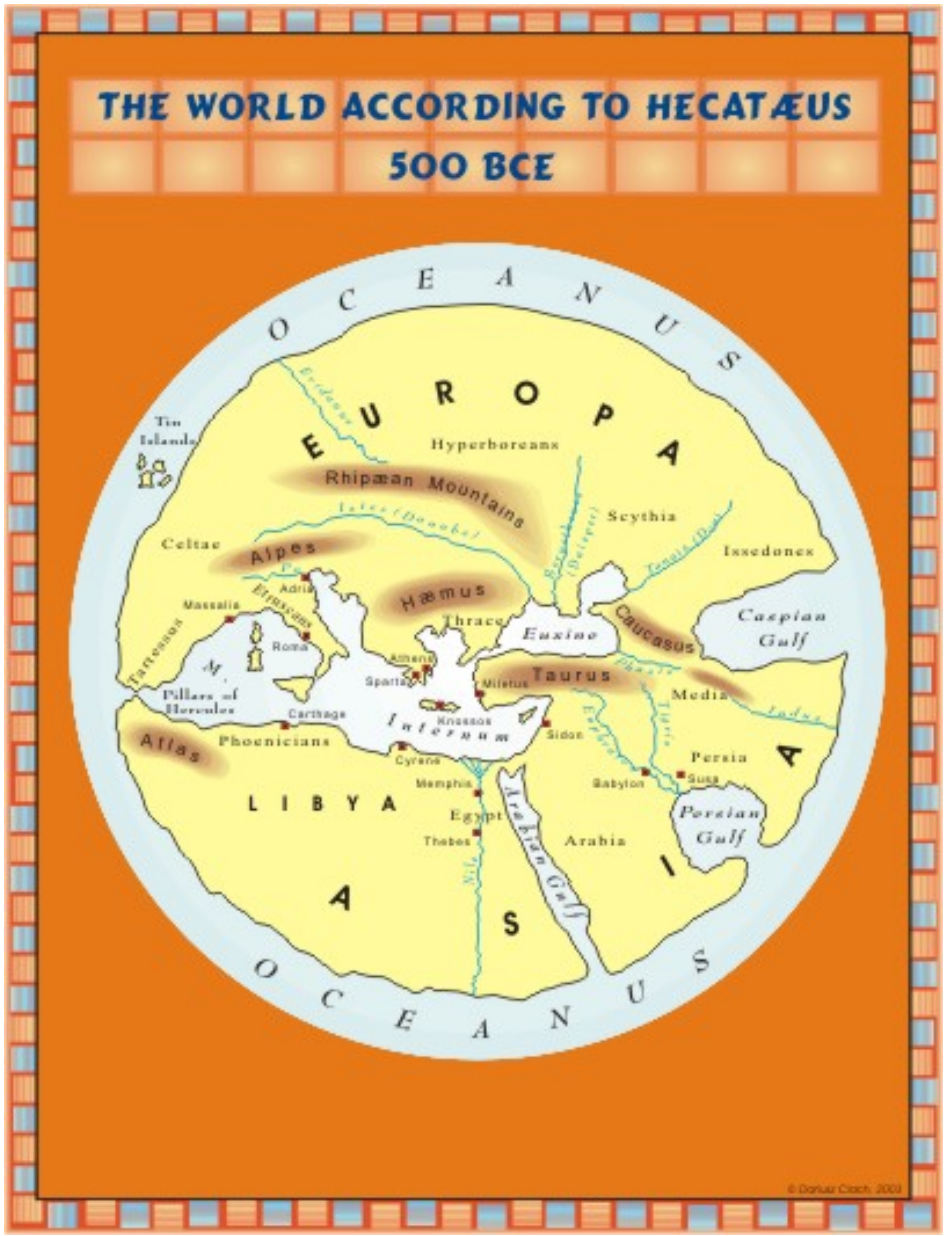
Our current work

	Discrete synchronous	Discrete partially synchronous	Discrete asynchronous	Continuous synchronous	Continuous partially synchronous
One instance/ finite payload	**one-shot broadcast, c.b.consensus** core of {ST87, BT87, CT96}, MA06 (common), MR04 (binary)				
Many inst./ finite payload					
Many inst./ unbounded payload					
Messages with reals					

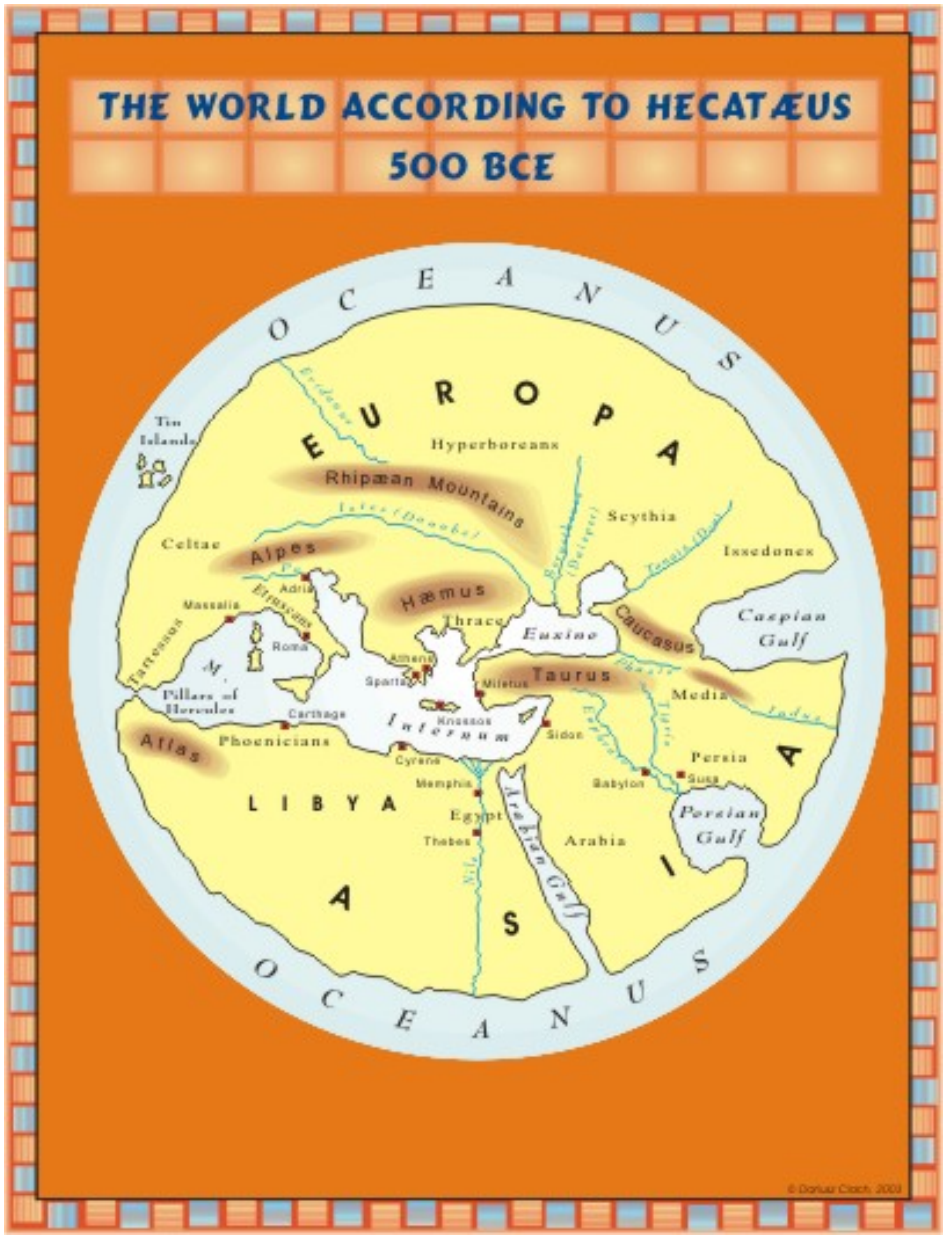
Future work: threshold guards + orthogonal features

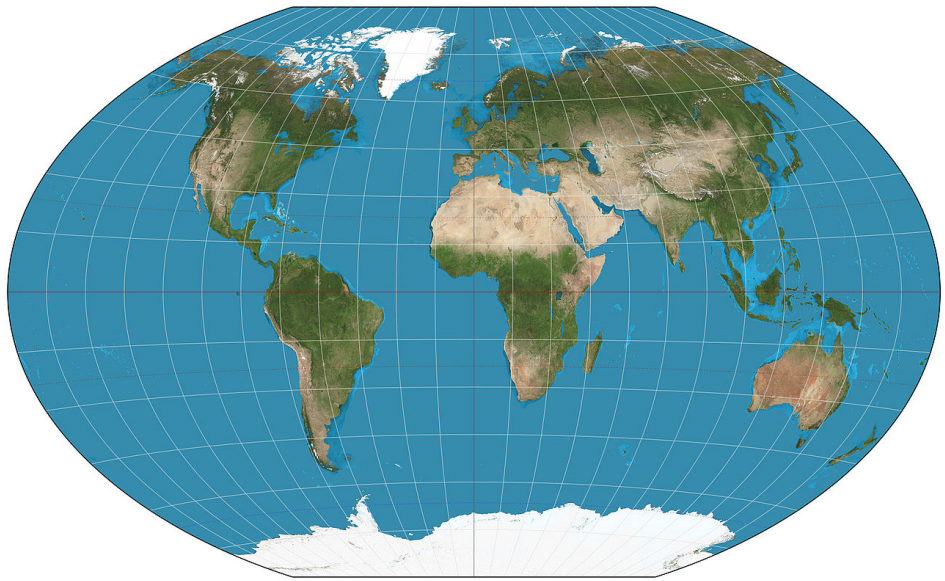


Fault-tolerant distributed algorithms: a big exciting world



Fault-tolerant distributed algorithms: a big exciting world





Thank you!

[http://forsyte.at/software/bymc]

Dealing with cycles: the idea

Recall that cycles do not update shared variables.

Find strongly connected components in the control flow graph
and define equivalence classes of edges.

When sorting the segments,
preserve the relative order of transitions within the equivalence classes.

After sorting, remove the cycles.

The length of an acyclic accelerated execution is bounded as before.

Explicit encoding of counter abstraction in Promela

```
/* number of processes in each local state */
int k[16];
/* the number of send-to-all's */
int nsnt = 0;
active[1] proctype CtrAbs() {
  int pc = 0, nrcvd = 0;
  int next_pc = 0, next_nrcvd = 0;
  /* init */
loop: /* select */
    /* receive-compute-send from data abstraction: */
        /* 1. receive */
        /* 2. compute */
        /* 3. send */
    /* update counters */
  goto loop;
}
```

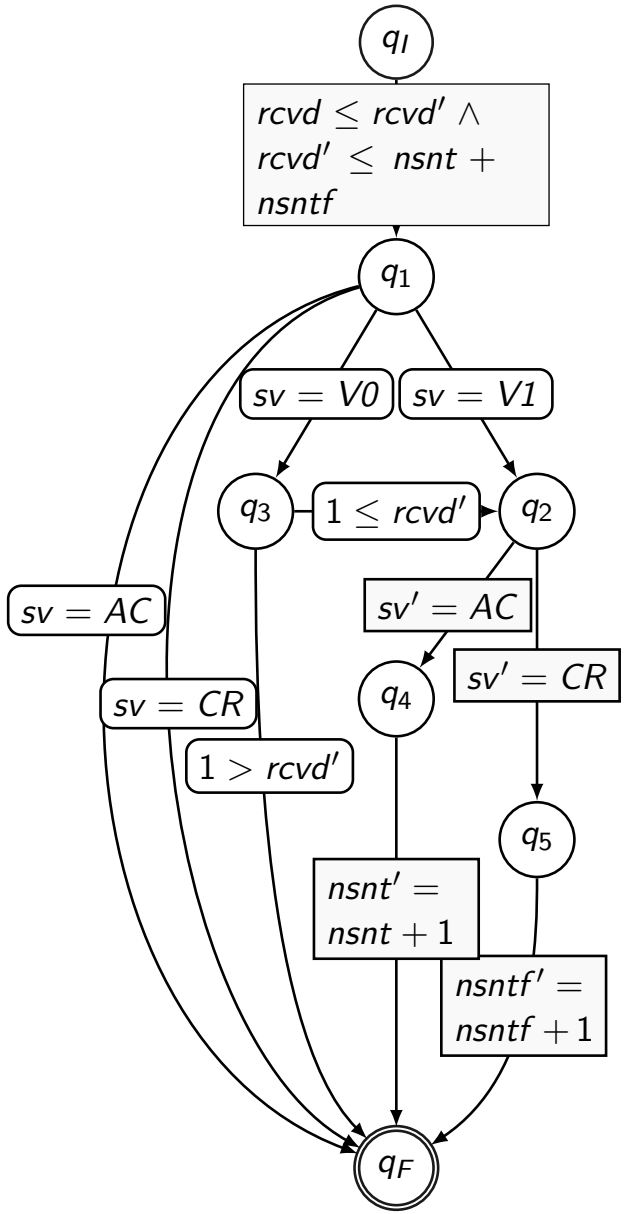
Diameters of counter systems

Our bound on the diameter of an (accelerated) counter system of a threshold automaton is $|E| \cdot (|\mathcal{C}| + 1) + |\mathcal{C}|$, or $O(|E|^2)$.

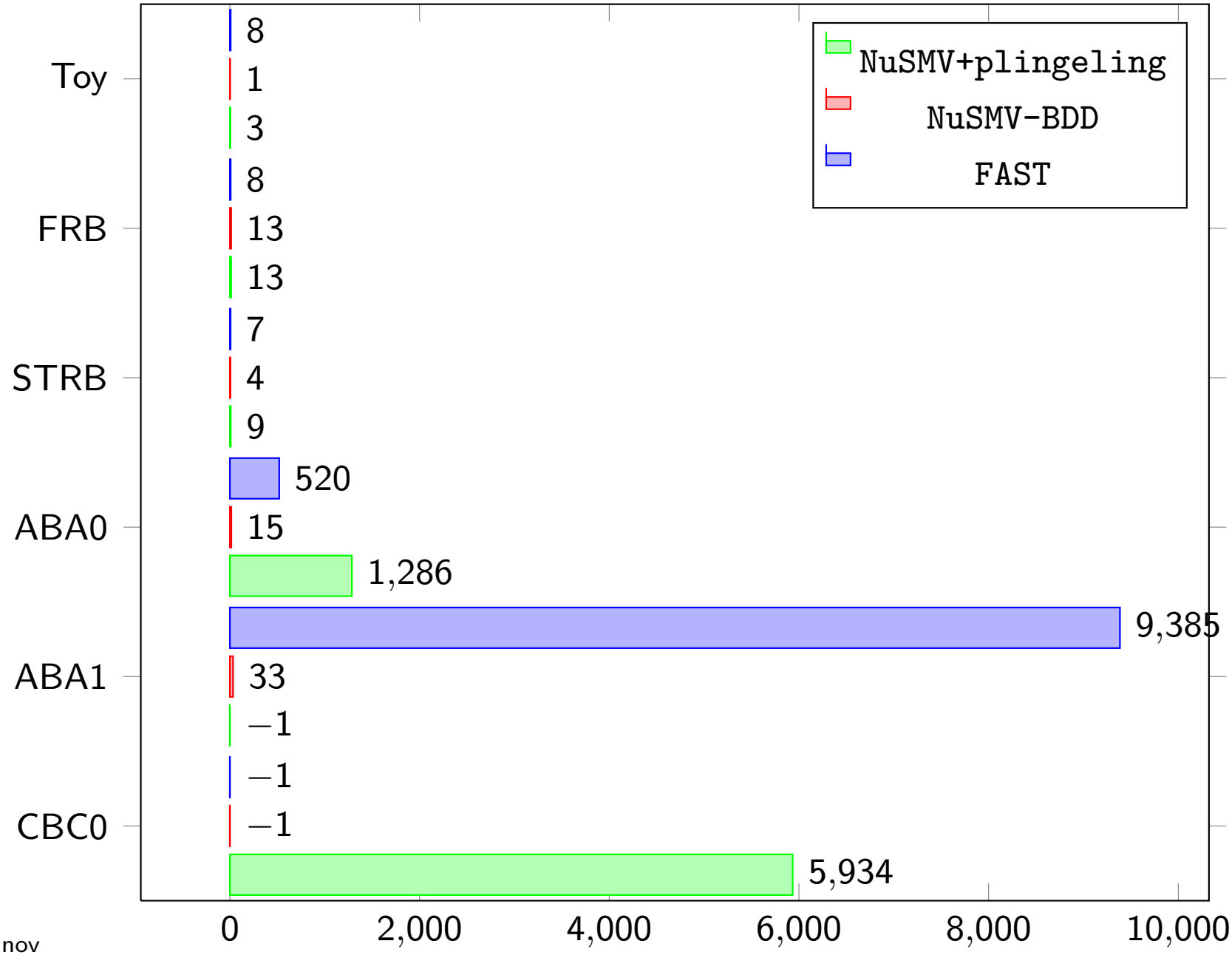
The number of conditions $|\mathcal{C}|$ is usually small, so we can bound the diameter with $O(|E|)$.

Forklore Reliable Broadcast

- crash faults,
- regular model checking for FTDA [Fisman, Kupferman, Lustig 2008],
- our technique also works with $\mathrm{I}_0 = [0; 1)$ and $\mathrm{I}_1 = [1; \infty)$.



Running time in comparison to other tools?



The diameter and refinement

The diameter does not grow up in the course of refinement!

Petri nets?