Reachability in Networks of Register Protocols under Stochastic Schedulers

Patricia Bouyer[1] Nicolas Markey[1] Mickael Randour[2]
Arnaud Sangnier[3] Daniel Stan[1]

[1]LSV - CNRS & ENS Cachan, France

[2]ULB, Belgium

[3]IRIF - CNRS & Université Paris Diderot, France

April 07, 2016 - IRISA - INRIA Rennes - *68NQRT seminar*

The talk in one slide

Networks of *arbitrarily many* identical processes:

- processes = non-deterministic automata,
- communication via a *shared register* (read and write),
- **fair** (stochastic) scheduler.

Question:

Is it the case that *almost-surely* one of the processes reaches a final state for a network of N processes?

▷ Existence of a **cut-off property** (constant answer for large N).

▷ EXPSPACE algorithm based on a *symbolic graph*.

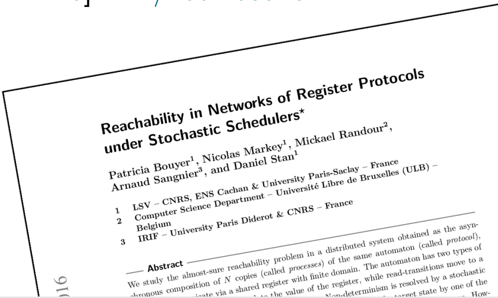
▷ **Cut-offs can be exponential**.

The talk in one slide. . . OK, two ☺

Goal of this talk:

- highlight the particularities of our model and their impact,
- understand typical examples,
- sketch the cornerstones of our solution.

Full paper available on arXiv [BMR+16]: abs/1602.05928



Reachability in Networks of Register Protocols under Stochastic Schedulers*

Patricia Bouyer[1], Nicolas Markey[1], Mickael Randour[2], Arnaud Sangnier[3], and Daniel Stan[1]

1 LSV – CNRS, ENS Cachan & University Paris-Saclay – France
2 Computer Science Department – Université Libre de Bruxelles (ULB) – Belgium
3 IRIF – University Paris Diderot & CNRS – France

Abstract
We study the almost-sure reachability problem in a distributed system obtained as the asyn-
chronous composition of N copies (called *processes*) of the same automaton (called *protocol*),
... a shared register with finite domain. The automaton has two types of
... the value of the register, while read-transitions move to a
... Non-determinism is resolved by a stochastic
... target state by one of the ... How-

1 Networks of register protocols

2 Almost-sure reachability

3 Cut-offs: existence and decision algorithm

4 Conclusion

Context: distributed systems

Goal

Study distributed systems composed of *many identical components* running concurrently.

Useful for distributed algorithms, ad-hoc networks, communication protocols, etc.

$\Longrightarrow$ Instead of fixing a bound on the number of components, we use **parameterized verification**.

Parameterized verification

Parameterized verification

Take the number of components as a parameter and identify an infinite set of parameter values for which the system is correct, if such a set exists.

E.g., *all networks of* $\geq N$ *components satisfy a given property*.

Advantages:

- general approach covering all parameter values,
- can be more efficient than checking the system for very large values as it involves orthogonal techniques (e.g., reducing the size of the network using structural arguments).

Parameterized networks

Every process follow the same protocol (usually, a finite-state automaton).

Different means of communication $\implies$ different models.

E.g.,

- *Rendez-vous* communication [GS92],
- broadcast communication [EFM99, DSZ10],
- token-passing [CTTV04, AJKR14],
- message passing [BGS14],
- shared register or memory [ABG15, EGM13].

$\implies$ **Minor changes in the setting can drastically change the complexity of verification problems.**

See Esparza's survey in STACS'14 [Esp14].

Our model in a nutshell

Processes

- *Protocol*: non-deterministic finite-state automaton.
- *Communication*: **non-atomic** read and write operations on a shared register (see [Hag11, EGM13, DEGM15]).

Some known results:

- ▷ Deciding if one process can reach a control state takes polynomial time (adapting [DSTZ12]).
- ▷ With a *leader* implementing a different protocol, NP-complete problem [EGM13].

Scheduler's role

In many works, the scheduler actually **helps** in reaching the target state: i.e., the question is whether there exists a scheduling such that a process reaches the target.

Our model in a nutshell

Scheduler

$\Longrightarrow$ Here, we want to get rid of this strong assumption.

$\Longrightarrow$ **Introduction of a fair scheduler.**

Two flavors of fairness:

1. *Temporal logic property* on executions (e.g., every action available infinitely often is performed infinitely often) (e.g., [GS92, AJK16]).
2. *Stochastic scheduler* (w.l.o.g. uniform distribution).

$\Longrightarrow$ **The stochastic scheduler breaks regular patterns (e.g., round-robin) and considers all possible interleaving with probability one in the long run.**

$\Longrightarrow$ **Important property for our approach.**

Related work

In [BFS14], Bertrand et al. study networks with

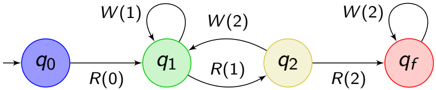
- stochastic protocols,
- communication via broadcast,
- an "helping scheduler".

One studied question is the existence of a network size and a scheduler granting almost-sure reachability of a control state: it turns out to be a coNP-complete problem.

$\Longrightarrow$ Despite apparent similarities, **the models are difficult to compare**: different use of probabilities, different communication mechanism, different role of the scheduler.

Our protocols

Definition



Register protocol with $D = \{0, 1, 2\}$.

Definition: register protocol

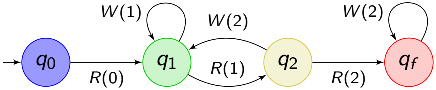
$\mathcal{P} = \langle Q, D, q_0, T \rangle$

- Q finite set of control locations;
- D finite alphabet of data for the shared register;
- $q_0 \in Q$ initial location;
- $T \subseteq Q \times \{R, W\} \times D \times Q$ set of transitions of the protocol.

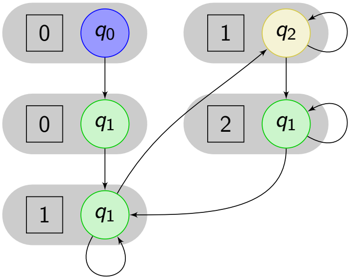
No deadlock and if R then all values in D can be read (omitted = self-loops).

Our protocols

Example



Imagine that our network contains a single process.



$\Longrightarrow$ **A single process cannot reach q_f.**

Our networks

Sketch

We study **distributed systems**:

- asynchronous composition of k copies of the protocol,
- non-determinism (inside the protocol and choice of process) resolved by a stochastic scheduler (uniform).

$\Longrightarrow$ Markov chain over the set of **configurations** $\Gamma = \mathbb{N}^Q \times D$ (multiset + data), finite if k is fixed.

$\Longrightarrow$ No creation/deletion of processes.

Notations:

- ▷ $\mathcal{S}_\mathcal{P}$ distributed system,
- ▷ $\mathcal{S}_\mathcal{P}^k$ distributed system of size k,
- ▷ $\gamma_0 \to \gamma_1 \ldots \to \gamma_n$ sequence of configurations, also $\gamma_0 \to^* \gamma_n$

Almost-sure reachability

For $q_f \in Q$:

- $[\![q_f]\!]$ = configurations covering q_f, i.e., γ s.t. $st(\gamma)(q_f) > 0$.
- $[\![\Diamond q_f]\!]$ = paths $\gamma_0 \rightarrow^* \gamma_n$ s.t. $\exists\, i \in [0; n]$, $st(\gamma_i)(q_f) > 0$.

  $\implies$ Paths covering q_f.
- $\mathbb{P}(\gamma, [\![\Diamond q_f]\!])$ = probability to cover q_f starting in γ.

$\implies$ **We seek cut-off properties for almost-sure reachability.**

Cut-off

Definition: cut-off

An integer $k \in \mathbb{N}$ is a *cut-off for almost-sure reachability* for $\mathcal{P}$, d_0 and q_f if one of the following two properties holds:

- for all $h \geq k$, we have $\mathbb{P}(\langle q_0^h, d_0 \rangle, [\![\Diamond q_f]\!]) = 1$. In this case k is a *positive* cut-off;
- for all $h \geq k$, we have $\mathbb{P}(\langle q_0^h, d_0 \rangle, [\![\Diamond q_f]\!]) < 1$. Then k is a *negative* cut-off.

An integer k is a *tight* cut-off if it is a cut-off and $k - 1$ is not.

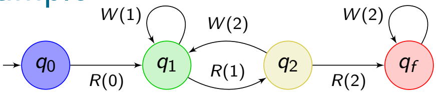
⚠ Cut-offs need not exist from the definition

and

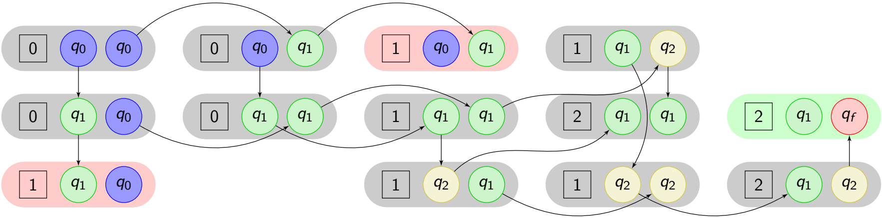
$\nexists$ positive $\not\Rightarrow$ $\exists$ negative.

$\hookrightarrow$ **We will prove that they always exist!**

Back to the example



Network for two processes (self-loops omitted).



$\Longrightarrow$ **From here, the process in q_0 is trapped hence the other one is alone and will never reach q_f.**

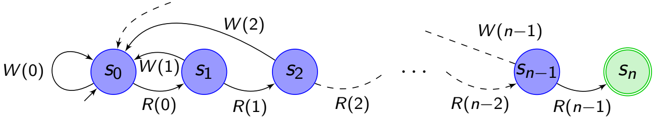
$\Longrightarrow$ **From here, non-exhaustive construction.**

$\Longrightarrow$ **With ≥ 2 processes, q_f reached with probability > 0 but < 1!**

$\Longrightarrow$ **$k = 1$ is a negative cut-off.**

Other examples

Positive cut-off



"Filter" protocol $\mathcal{F}_n$ for $n > 0$.

For protocol $\mathcal{F}_n$,

- networks of size $\geq n$ cover s_n with probability 1,
- networks of size $< n$ cannot cover s_n.

No deadlock can ever occur as all processes can always go back to the initial state.

$\Longrightarrow$ **Tight positive cut-off equal to n, i.e., linear in the protocol size.**

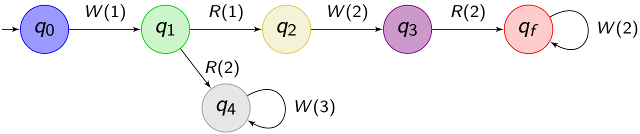
Other examples

Lack of monotonicity for small network sizes

Observation

When considering an "helping scheduler" as in many models, increasing the network size is never a bad thing (as the scheduler can decide not to activate the additional processes at all).

$\Longrightarrow$ **Not true anymore with our fair scheduler!**



$\Longrightarrow$ **Additional processes can create new deadlocks!**

$\Longrightarrow$ **We need new techniques to detect such behaviors.**

Existence of a cut-off

Main result

Theorem

For any register protocol $\mathcal{P}$, any initial register value d_0 and any target location q_f, **there always exists a cut-off for almost-sure reachability**, whose value is at most doubly-exponential in the size of $\mathcal{P}$. Whether it is a positive or a negative cut-off can be decided in EXPSPACE, and is PSPACE-hard.

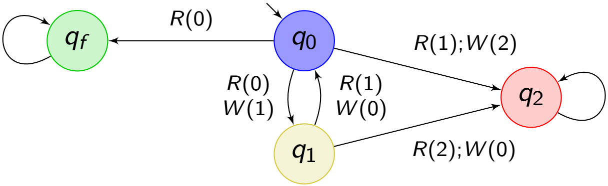
⚠ This result strongly relies on the "regularity-breaking" aspect of our stochastic scheduler and on the non-atomicity of read/write operations.

The non-atomicity guarantees that when a process takes a transition, all processes in the same transition can also take the same transition (with a non-zero probability).

$\Longrightarrow$ **Crucial to obtain a copycat lemma.**

Existence of a cut-off

Atomic read/write $\rightsquigarrow$ no cut-off



$\Longrightarrow$ State q_f is reached with probability one if and only if the network size is odd.

Existence of a cut-off

Proof sketch (1/3)

1. Partial order $\preceq$ over configurations s.t. $\langle \mu, d \rangle \preceq \langle \mu', d' \rangle$ iff $d = d'$, the multisets have the same support and $\mu \sqsubseteq \mu'$.

   $\Longrightarrow$ **$\langle \Gamma, \preceq \rangle$ is a wqo.**

2. For $k > 0$,

   $$\mathbb{P}(\langle q_0^k, d_0 \rangle, [\![\Diamond q_f]\!]) = 1 \Leftrightarrow \mathrm{Post}^*(\{\langle q_0^k, d_0 \rangle\}) \subseteq \mathrm{Pre}^*([\![q_f]\!]).$$

   $\Longrightarrow$ **Cut-off k_0 if for all $k \geq k_0$, either the inclusion is always true or it is always false.**

3. *Copycat lemma*: if $\gamma_1 \rightarrow^* \gamma_2$ and $\gamma_2 \preceq \gamma_2'$, then there exists γ_1' such that $\gamma_1' \rightarrow^* \gamma_2'$ and $\gamma_1 \preceq \gamma_1'$.

   $\Longrightarrow$ **Monotonicity property.**

4. $\mathrm{Post}^*(\uparrow\{\langle q_0, d_0 \rangle\})$ and $\mathrm{Pre}^*([\![q_f]\!])$ are **upward-closed sets**.

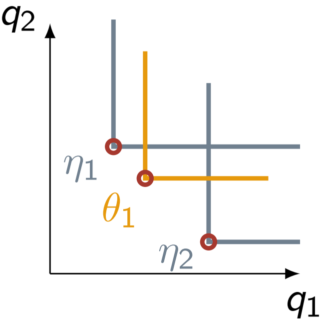
   $\Longrightarrow$ **Can be represented by minimal elements!**

Existence of a cut-off

Proof sketch (2/3)

5. $\mathrm{Post}^*(\uparrow\{\langle q_0, d_0\rangle\}) = \uparrow\{\theta_1, \ldots, \theta_n\}$ and $\mathrm{Pre}^*(\llbracket q_f \rrbracket) = \uparrow\{\eta_1, \ldots, \eta_m\}$.
6. *Is $\mathrm{Post}^*(\uparrow\{\langle q_0, d_0\rangle\})$ included to $\mathrm{Pre}^*(\llbracket q_f \rrbracket)$ modulo single-state incrementation?*

$\Longrightarrow$ **A bit technical. . .**



. . . intuitively, the goal is to check if elements of $\mathrm{Post}^*(\uparrow\{\langle q_0, d_0\rangle\})$ can enter $\mathrm{Pre}^*(\llbracket q_f \rrbracket)$ by adding sufficiently many processes in a given state.

Existence of a cut-off

Proof sketch (3/3)

7. If $\textsc{No}$, then there is a **negative cut-off**.

   $\hookrightarrow$ For each k sufficiently large, we can build a configuration that is in $\mathrm{Post}^*(\{\langle q_0^k, d_0\rangle\})$ but not in $\mathrm{Pre}^*([\![q_f]\!])$ $\Longrightarrow \mathbb{P}(\langle q_0^k, d_0\rangle, [\![\Diamond q_f]\!]) < 1$.

8. If $\textsc{Yes}$, then there is a **positive cut-off**.

   $\hookrightarrow$ For k sufficiently large, every configuration in $\mathrm{Post}^*(\{\langle q_0^k, d_0\rangle\})$ is also in $\mathrm{Pre}^*([\![q_f]\!])$ $\Longrightarrow \mathbb{P}(\langle q_0^k, d_0\rangle, [\![\Diamond q_f]\!]) = 1$.

$\Longrightarrow$ **There is always a cut-off!**

$\Longrightarrow$ **Value of the cut-off at most polynomial in the size of the minimal elements...**

Deciding the nature of the cut-off

Goal

Decide if the system admits a *negative* cut-off. If not, then there is a *positive* one.

Idea

Abstract *arbitrarily large* systems by a **symbolic graph** of bounded size and study this graph to conclude.

$\Longrightarrow$ **The crux is to maintain enough information!**

Symbolic graph

Traditional approach: using only supports (1/2)

Fully symbolic graph:

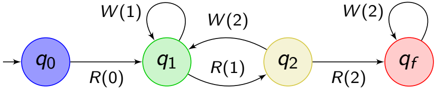
- We totally abstract the number of processes in each state by keeping only *supports* of configurations.
- Sufficient abstraction in simpler models.

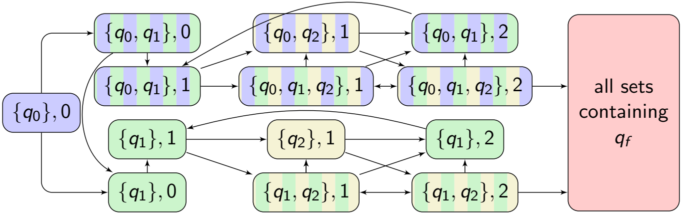
Hope (soon to be crushed)

State q_f is almost-surely covered if and only if supports containing q_f are reachable from all reachable states in the symbolic graph.

Symbolic graph

Traditional approach: using only supports (2/2)





What can we conclude from the symbolic graph?

q_f **is reachable from everywhere, so positive cut-off?**

No! We saw that $k = 1$ is a negative cut-off!

Symbolic graph

Extending this approach

Is this graph useless?

$\Longrightarrow$ **No! One direction of the equivalence holds.**

Observation

If the symbolic graph contains a deadlock (i.e., a reachable state from which q_f is not reachable), then there is a negative cut-off.

This holds because *from any run in the symbolic graph, one can build a mimicking one in the real system* given a sufficient number of processes.

$\Longrightarrow$ **To obtain the other direction, we need to add information in the symbolic graph.**

$\Longrightarrow$ **We introduce a concrete part to track precisely the behavior of a bounded number of processes.**

Symbolic graph

Adding a concrete part

Definition: symbolic graph of index k

$\mathcal{G} = \langle V, v_0, E\rangle$ where

- $V = \mathbb{N}_k^Q \times 2^Q \times D$: concrete part keeping track of a fixed set of k processes, abstract part encoding the arbitrarily many remaining processes, data;
- $v_0 = \langle q_0^k, \{q_0\}, \{d_0\}\rangle$;
- $\langle \mu, S, d\rangle \to \langle \mu', S', d'\rangle$ for each $(q, O, d'', q') \in T$ such that $d = d' = d''$ if $O = R$ and $d' = d''$ if $O = W$, and one of the following two conditions holds:
  - either $S' = S$ and $q \sqsubseteq \mu$ and $\mu' = \mu \ominus q \oplus q'$;
  - or $\mu = \mu'$ and $q \in S$ and $S' \in \{S \setminus \{q\} \cup \{q'\}, S \cup \{q'\}\}$.

$\hookrightarrow$ Transitions either impact the concrete part or the symbolic part, not both (i.e., no exchange of processes).

Symbolic graph

Toward a correct and complete algorithm

Recall that $\mathrm{Pre}^*(\llbracket q_f \rrbracket) = \uparrow\{\eta_i \mid 1 \leq i \leq m\}$. We show that the symbolic graph abstraction is complete for $k = K \cdot |Q|$, where $K = \max\{st(\eta_i)(q) \mid q \in Q,\ 1 \leq i \leq m\}$.

$\Longrightarrow$ **Intuitively, the concrete part must be large enough to capture executions involving minimal elements of $\mathrm{Pre}^*(\llbracket q_f \rrbracket)$.**

Theorem

There is a negative cut-off for $\mathcal{P}$, d_0 and q_f if, and only if, there is a node in the symbolic graph of index $K \cdot |Q|$ that is reachable from $\langle q_0^{K \cdot |Q|}, \{q_0\}, d_0 \rangle$ but from which no configuration involving q_f is reachable.

Complexity (1/2)

Upper bounds

- ▷ Using results by Rackoff on the coverability problem in VAS [Rac78, DJLL13], we bound K (hence the size of the graph since we use multisets and not vectors) by a double-exponential in the size of the protocol.
- ▷ Reachability in NLOGSPACE [Sip97] w.r.t. the graph $\Longrightarrow$ NEXPSPACE w.r.t. the protocol $\Longrightarrow$ EXPSPACE by Savitch's theorem [Sip97].
- ▷ Doubly-exponential upper bounds on cut-off values.

Complexity (2/2)

Lower bounds

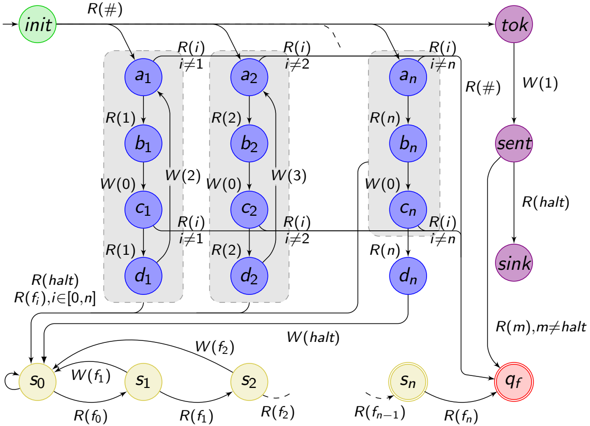
- PSPACE-hardness via linear-bounded Turing machine [Sip97]: we build a protocol for which there is a negative cut-off iff the machine reaches its final state q_{halt}.
- Best lower bound for positive cut-offs so far: linear (cf. "filter" protocol).

$\Longrightarrow$ **Huge gap!**

- Best lower bound for negative cut-offs so far: exponential.

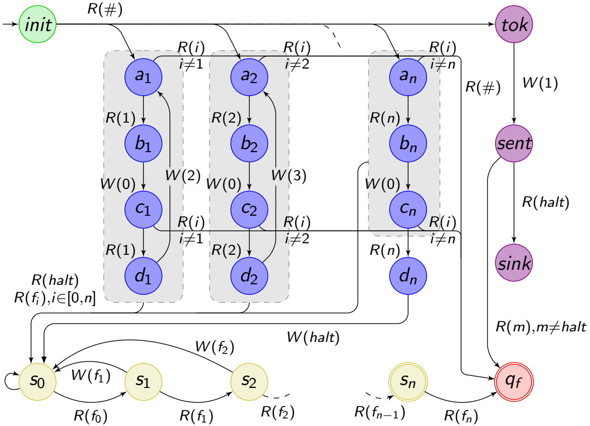
$\Longrightarrow$ **Shares ideas with PSPACE-hardness proof. Let's discuss it now.**

Exponential negative cut-off



Different parts: simulating a counter over n bits, producing tokens needed for the simulation, filter protocol, $d_0 = \#$, target q_f.

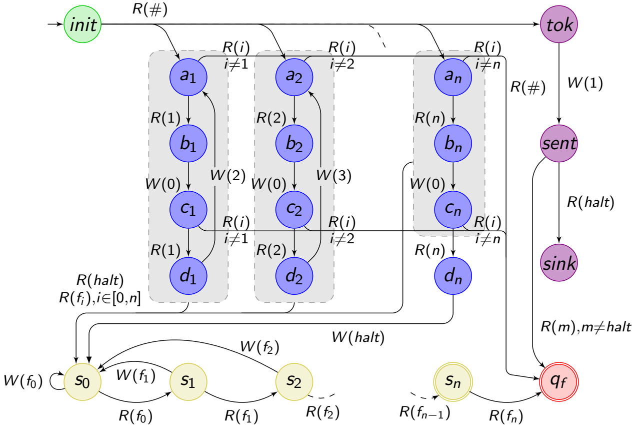
Exponential negative cut-off



Claim: $\exists N > 2^n$ s.t. $\mathbb{P}(\langle init^N, \# \rangle, [\![\Diamond q_f]\!]) < 1$ while $\mathbb{P}(\langle init^{2^n}, \# \rangle, [\![\Diamond q_f]\!]) = 1$.

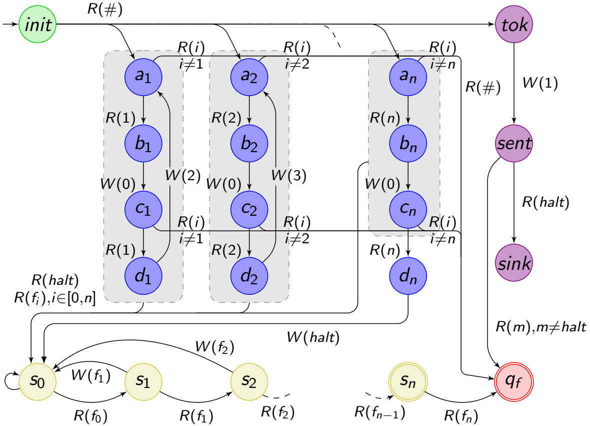
$\Longrightarrow$ **Exponential tight negative cut-off.**

Exponential negative cut-off



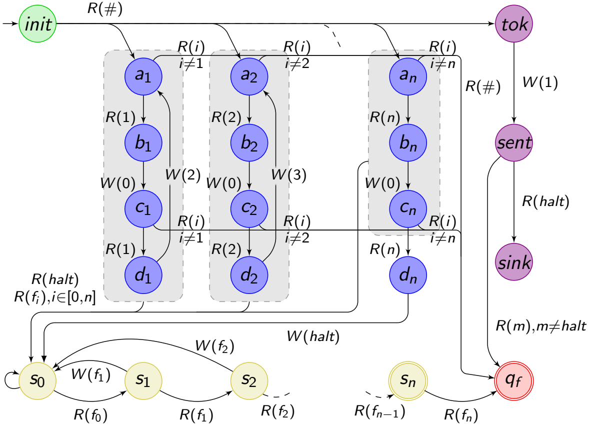
Three phases: initialization, simulation, counting.

Exponential negative cut-off



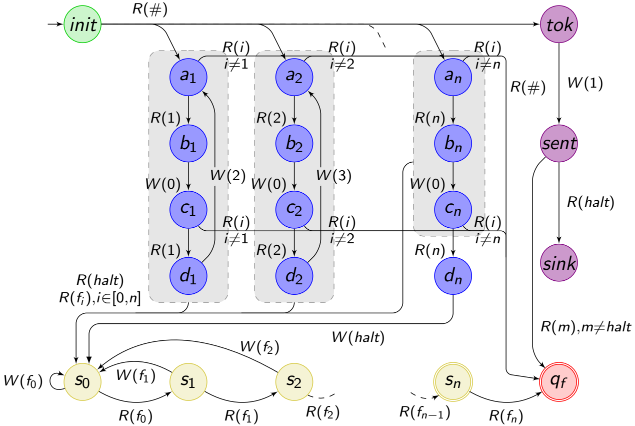
Phase 1: initialization. Processes move to a_i and *tok* until some process in *tok* writes 1 in the register (or until someone reaches q_f by reading # from a_i).

Exponential negative cut-off



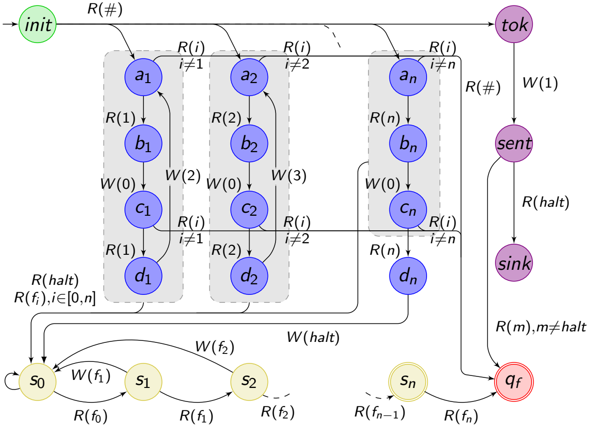
Phase 2: simulation. If all the processes are in *tok*, they will eventually reach q_f. So we assume that there is at least one process in a state a_i.

Exponential negative cut-off



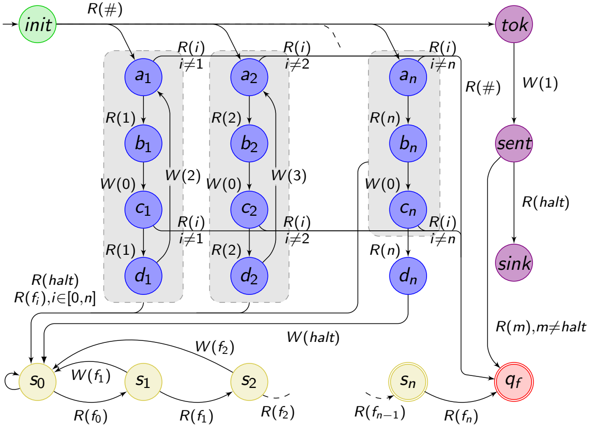
If some a_i is empty, then d_n cannot be reached and we cannot enter the counting phase $\implies$ some process will eventually reach q_f.

Exponential negative cut-off



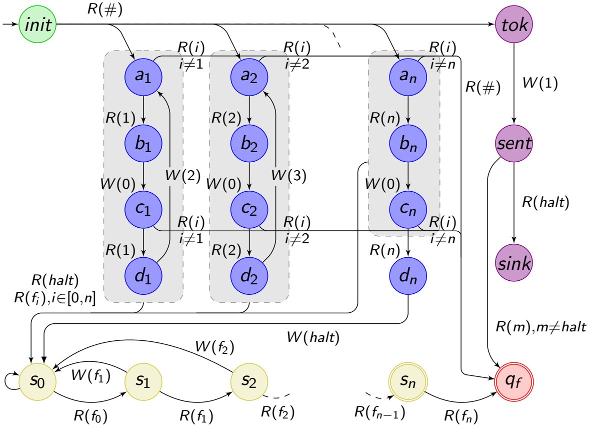
Thus, assume there is at least one process in each state a_i. We can prove that d_i is reachable when at the start of the simulation phase, at least 2^i processes are in *tok* (we need to produce an exponential number of tokens).

Exponential negative cut-off



Reaching s_0 thus requires 2^n processes in *tok*. If we want to avoid reaching q_f, the counting phase must never contain more than n processes (because we have an $(n+1)$ filter). So we assume each a_i has *exactly* one process at the start of the simulation.

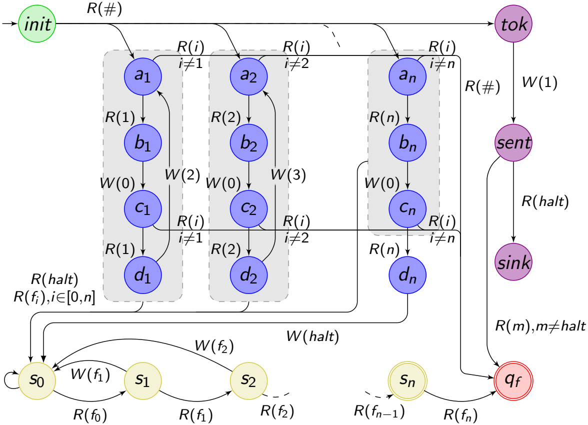
Exponential negative cut-off



To avoid reaching q_f, we need n processes in states a_i and at least 2^n processes in *tok*.

$\Longrightarrow$ **q_f is almost-surely reached in systems with strictly less than $n + 2^n$ processes.**

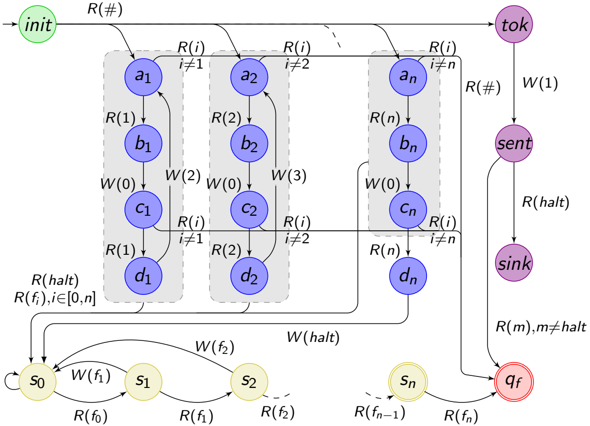
Exponential negative cut-off



It remains to show that for $N \geq n + 2^n$, q_f cannot be reached almost-surely.

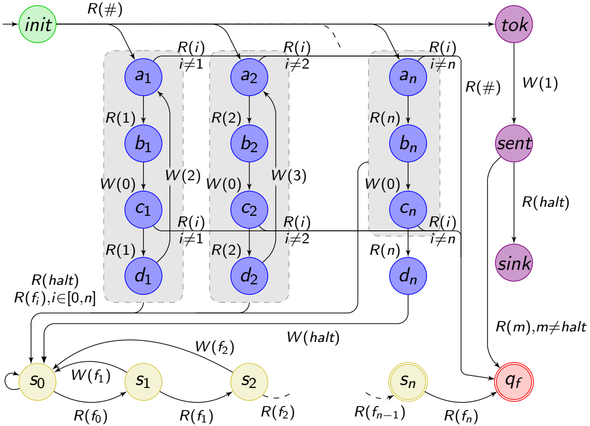
$\Longrightarrow$ **Exhibit a finite execution having no continuation reaching q_f.**

Exponential negative cut-off



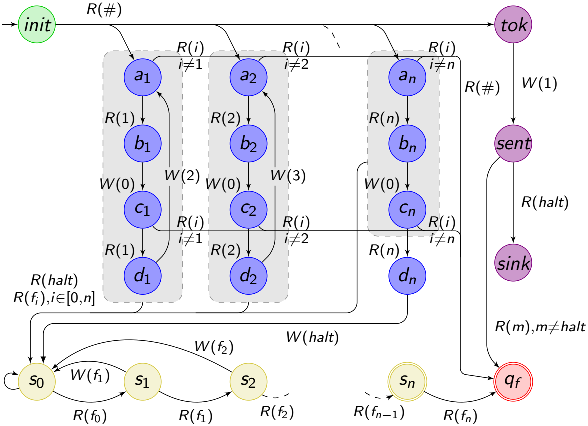
Execution: during initialization, put one process in each a_i and all others in *tok*. One of them writes 1.

Exponential negative cut-off



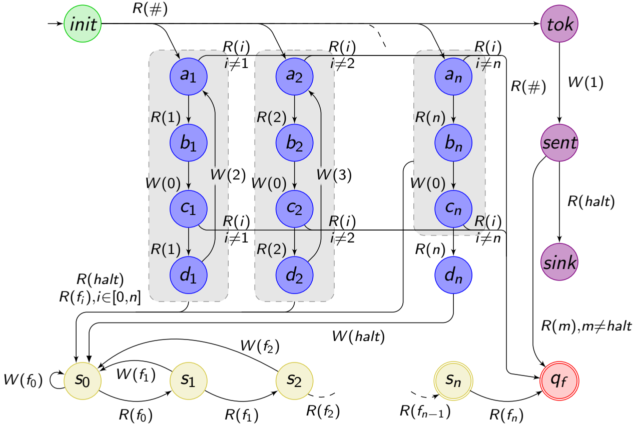
The n processes in states a_i then simulate the incrementations of the counter, consuming tokens at each step, until reaching d_n.

Exponential negative cut-off



All processes in *tok* move to *sent* and the process in d_n writes *halt* and moves to s_0. Other processes in the simulation phase move to s_0 and processes in *sent* move to *sink*.

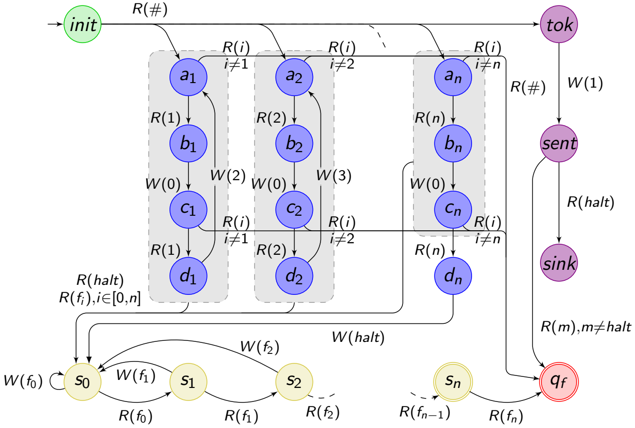
Exponential negative cut-off



We are left with n processes in s_0 and all the others in *sink*. Since we have an $(n+1)$ filter, q_f cannot be reached.

$$\implies \mathbb{P}(\langle \textbf{init}^N, \# \rangle, [\![\Diamond q_f]\!]) < 1 \textbf{ for } N = n + 2^n.$$

Exponential negative cut-off



We have proved a tight negative cut-off of exponential size.

Summary

Our model:

- register protocols,
- non-atomic read/write operations,
- fairness via stochastic scheduler.

Some differences with classical models:

- lack of monotonicity in general,
- complexity (PSPACE-hardness while many problems are polynomial or in NP/coNP),
- cut-offs may be exponential (most models admit polynomial cut-offs).

$\Longrightarrow$ **Slight changes in the setting induce important changes in complexity.**

Future work

Many open questions:

- closing the gaps (complexity, cut-off bounds),
- other objectives (e.g., liveness),
- quantitative questions,
- atomic read/write operations,
- synthesis of local strategies.

Many thanks! Any question?

References I

C. Aiswarya, Benedikt Bollig, and Paul Gastin.
An automata-theoretic approach to the verification of distributed algorithms.
In Luca Aceto and David de Frutos-Escrig, editors, *Proceedings of the 26th International Conference on Concurrency Theory (CONCUR'15)*, volume 42 of *Leibniz International Proceedings in Informatics*, pages 340–353. Leibniz-Zentrum für Informatik, September 2015.

Simon Außerlechner, Swen Jacobs, and Ayrat Khalimov.
Tight cutoffs for guarded protocols with fairness.
In Barbara Jobstmann and K. Rustan M. Leino, editors, *Proceedings of the 17th International Workshop on Verification, Model Checking, and Abstract Interpretation (VMCAI'16)*, volume 9583 of *Lecture Notes in Computer Science*, pages 476–494. Springer-Verlag, January 2016.

Benjamin Aminof, Swen Jacobs, Ayrat Khalimov, and Sasha Rubin.
Parametrized model checking of token-passing systems.
In Kenneth L. McMillan and Xavier Rival, editors, *Proceedings of the 15th International Workshop on Verification, Model Checking, and Abstract Interpretation (VMCAI'14)*, volume 8318 of *Lecture Notes in Computer Science*, pages 262–281. Springer-Verlag, January 2014.

Nathalie Bertrand, Paulin Fournier, and Arnaud Sangnier.
Playing with probabilities in reconfigurable broadcast networks.
In *Proc. of FOSSACS*, LNCS 8412, pages 134–148. Springer, 2014.

Benedikt Bollig, Paul Gastin, and Len Schubert.
Parameterized verification of communicating automata under context bounds.
In Joël Ouaknine, Igor Potapov, and James Worrell, editors, *Proceedings of the 8th Workshop on Reachability Problems in Computational Models (RP'14)*, volume 8762 of *Lecture Notes in Computer Science*, pages 45–57. Springer-Verlag, September 2014.

References II

Patricia Bouyer, Nicolas Markey, Mickael Randour, Arnaud Sangnier, and Daniel Stan.
Reachability in networks of register protocols under stochastic schedulers.
CoRR, abs/1602.05928, 2016.

Edmund M. Clarke, Muralidhar Talupur, Tayssir Touili, and Helmut Veith.
Verification by network decomposition.
In Philippa Gardner and Nobuko Yoshida, editors, *Proceedings of the 15th International Conference on Concurrency Theory (CONCUR'04)*, volume 3170 of *Lecture Notes in Computer Science*, pages 276–291. Springer-Verlag, August-September 2004.

Antoine Durand-Gasselin, Javier Esparza, Pierre Ganty, and Rupak Majumdar.
Model checking parameterized asynchronous shared-memory systems.
In Daniel Kroening and Corina S. Pasareanu, editors, *Proceedings of the 27th International Conference on Computer Aided Verification (CAV'15)*, volume 9206 of *Lecture Notes in Computer Science*, pages 67–84. Springer-Verlag, July 2015.

Stéphane Demri, Marcin Jurdziński, Oded Lachish, and Ranko Lazić.
The covering and boundedness problems for branching vector addition systems.
Journal of Computer and System Sciences, 79(1):23–38, February 2013.

Giorgio Delzanno, Arnaud Sangnier, Riccardo Traverso, and Gianluigi Zavattaro.
On the complexity of parameterized reachability in reconfigurable broadcast networks.
In Deepak D'Souza, Telikepalli Kavitha, and Jaikumar Radhakrishnan, editors, *Proceedings of the 32nd Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'12)*, volume 18 of *Leibniz International Proceedings in Informatics*, pages 289–300. Leibniz-Zentrum für Informatik, December 2012.

References III

Giorgio Delzanno, Arnaud Sangnier, and Gianluigi Zavattaro.
Parameterized verification of ad hoc networks.
In Paul Gastin and François Laroussinie, editors, *Proceedings of the 21st International Conference on Concurrency Theory (CONCUR'10)*, volume 6269 of *Lecture Notes in Computer Science*, pages 313–327. Springer-Verlag, September 2010.

Javier Esparza, Alain Finkel, and Richard Mayr.
On the verification of broadcast protocols.
In *Proceedings of the 14th Annual Symposium on Logic in Computer Science (LICS'99)*, pages 352–359. IEEE Comp. Soc. Press, July 1999.

Javier Esparza, Pierre Ganty, and Rupak Majumdar.
Parameterized verification of asynchronous shared-memory systems.
In Natasha Sharygina and Helmut Veith, editors, *Proceedings of the 25th International Conference on Computer Aided Verification (CAV'13)*, volume 8044 of *Lecture Notes in Computer Science*, pages 124–140. Springer-Verlag, July 2013.

Javier Esparza.
Keeping a crowd safe: On the complexity of parameterized verification (invited talk).
In Ernst W. Mayr and Natacha Portier, editors, *Proceedings of the 31st Symposium on Theoretical Aspects of Computer Science (STACS'14)*, volume 25 of *Leibniz International Proceedings in Informatics*, pages 1–10. Leibniz-Zentrum für Informatik, March 2014.

Steven M. German and A. Prasad Sistla.
Reasoning about systems with many processes.
Journal of the ACM, 39(3):675–735, July 1992.

References IV

Matthew Hague.
Parameterised pushdown systems with non-atomic writes.
In Supratik Chakraborty and Amit Kumar, editors, *Proceedings of the 31st Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'11)*, volume 13 of *Leibniz International Proceedings in Informatics*, pages 457–468. Leibniz-Zentrum für Informatik, December 2011.

Charles Rackoff.
The covering and boundedness problems for vector addition systems.
Theoretical Computer Science, 6:223–231, 1978.

Michael Sipser.
Introduction to the theory of computation.
PWS Publishing Company, 1997.