Reachability in Networks of Register Protocols under Stochastic Schedulers

Patricia Bouyer    Nicolas Markey    Mickael Randour
Arnaud Sangnier    Daniel Stan

(Most slides are courtesy of Mickael Randour)
The talk in one slide

Networks of *arbitrarily many* identical processes:
The talk in one slide

Networks of arbitrarily many identical processes:
- processes = non-deterministic automata,
The talk in one slide

Networks of *arbitrarily many* identical processes:

- processes = non-deterministic automata,
- communication via a *shared register* (read and write),
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.
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?
The talk in one slide

Networks of \textit{arbitrarily many} identical processes:

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

Question:

Is it the case that \textit{almost-surely} one of the processes reaches a final state for a network of $N$ processes?

- Existence of a \textit{cut-off property} (constant answer for large $N$).
The talk in one slide

Networks of \emph{arbitrarily many} identical processes:

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

**Question:**

Is it the case that \emph{almost-surely} one of the processes reaches a final state for a network of $N$ processes?

- Existence of a \emph{cut-off property} (constant answer for large $N$).
- EXPSPACE algorithm based on a \emph{symbolic graph}. 
Networks of register protocols

Almost-sure reachability

Cut-offs

Conclusion

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**.
1. Networks of register protocols
2. Almost-sure reachability
3. Cut-offs: existence and decision algorithm
4. Conclusion
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.
Context: distributed systems

Goal

Study distributed systems composed of many identical components running concurrently.

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

Exploit symmetries of such distributed systems for efficient 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.
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).
Our model in a nutshell

Processes

- *Communication*: **non-atomic** read and write operations on a shared register (see [Hag11, EGM13, DEGM15]).
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].
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 scheduler such that a process reaches the target.
Our model in a nutshell

Scheduler

▷ Here, we want to get rid of this strong assumption.
Our model in a nutshell

Scheduler

▷ Here, we want to get rid of this strong assumption.

∽ Introduction of a fair scheduler.
Our model in a nutshell

Scheduler

▷ Here, we want to get rid of this strong assumption.

〜 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]).
Our model in a nutshell

Scheduler

▶ Here, we want to get rid of this strong assumption.

ılmış 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).

The stochastic scheduler breaks regular patterns (e.g., round-robin) and considers all possible interleaving with probability one in the long run.
Our model in a nutshell

Scheduler

▷ Here, we want to get rid of this strong assumption.

→ 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).

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

→ Important property for our approach.
Related work

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

- stochastic protocols,
- communication via broadcast,
- a “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.
Related work

In [BFS14], Bertrand et al. study networks with
- stochastic protocols,
- communication via broadcast,
- a “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.

⇒ Despite apparent similarities, the models are difficult to compare: different use of probabilities, different communication mechanism, different role of the scheduler.
Our protocols

Definition

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, with a default value \( d_0 \);
- \( q_0 \in Q \) initial location;
- \( T \subseteq Q \times \{ R, W \} \times D \times Q \) set of transitions of the protocol.
Our protocols

Example

Imagine that our network contains a single process.
Our protocols

Example

Imagine that our network contains a single process.
Our protocols

Example

Imagine that our network contains a single process.
Imagine that our network contains a single process.
Imagine that our network contains a single process.
Our protocols

Example

Imagine that our network contains a single process.
Our protocols

Example

Imagine that our network contains a single process.

A single process cannot reach \( q_f \).
Our networks

Sketch

We study **distributed systems**:

- asynchronous composition of $k$ copies of the protocol,
- possibly one copy of a different protocol (leader),
- non-determinism (inside the protocols and choice of process) resolved by a stochastic scheduler (uniform).
Our networks

Sketch

We study **distributed systems**:

- asynchronous composition of \( k \) copies of the protocol,
- possibly one copy of a different protocol (leader),
- non-determinism (inside the protocols and choice of process) resolved by a stochastic scheduler (uniform).

- Markov chain over the set of **configurations**
  \[ \Gamma = Q_l \times \mathbb{N}^{Q_c} \times D \text{ (leader + multiset + data)} \]
- finite if \( k \) is fixed; no creation/deletion of processes.
Our networks

Semantics
Our networks

Semantics

$q_0 \rightarrow 5$
$q_1 \rightarrow 0$
$q_2 \rightarrow 0$
$q_f \rightarrow 0$
Our networks

Semantics
Our networks

Semantics
Our networks
Semantics
Our networks

Semantics
Our networks

Semantics
1. Networks of register protocols

2. Almost-sure reachability

3. Cut-offs: existence and decision algorithm

4. Conclusion
Almost-sure reachability

For $q_f \in Q$:

- $[q_f] = \text{configurations covering } q_f$, i.e., $\gamma$ s.t. $st(\gamma)(q_f) > 0$.
- $[\Diamond q_f] = \text{paths } \gamma_0 \rightarrow^* \gamma_n \text{ s.t. } \exists i \in [0; n], st(\gamma_i)(q_f) > 0$.  
  $\implies \text{Paths covering } q_f$.
- $\mathbb{P}(\gamma, [\Diamond q_f]) = \text{probability to cover } q_f \text{ starting in } \gamma$.

$\leadsto \text{We seek cut-off properties for almost-sure reachability.}$
Cut-off

Definition: cut-off

An integer $k \in \mathbb{N}$ is a cut-off for almost-sure reachability 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 $\n\nexists$ negative.
Cut-off

Definition: cut-off

An integer $k \in \mathbb{N}$ is a cut-off for almost-sure reachability 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.

↔ We will prove that they always exist!
Back to the example

Network for two processes (self-loops omitted).
Back to the example

Network for two processes (self-loops omitted).
Back to the example

Network for two processes (self-loops omitted).

Network of register protocols
Back to the example

Network for two processes (self-loops omitted).

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

From here, non-exhaustive construction.

With 2 processes, $q_f$ reached with probability $\geq 0$ (but $< 1$).

$k = 1$ is a negative cut-off.
Back to the example

Network for two processes (self-loops omitted).

From here, the process in $q_0$ is trapped hence the other one is alone and will never reach $q_f$. 
Back to the example

Network for two processes (self-loops omitted).

From here, the process in $q_0$ is trapped hence the other one is alone and will never reach $q_f$. 
Back to the example

Network for two processes (self-loops omitted).

⇒ From here, the process in \( q_0 \) is trapped hence the other one is alone and will never reach \( q_f \).
Back to the example

Network for two processes (self-loops omitted).

⇒ From here, the process in $q_0$ is trapped hence the other one is alone and will never reach $q_f$. 
Back to the example

Network for two processes (self-loops omitted).

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

From here, non-exhaustive construction.
Back to the example

Network for two processes (self-loops omitted).

\[ \begin{align*}
W(1) \quad & \quad W(2) \\
q_0 & \rightarrow q_1 & R(0) \\
q_1 & \rightarrow q_2 & R(1) \\
q_2 & \rightarrow q_f & R(2)
\end{align*} \]

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

From here, non-exhaustive construction.
Back to the example

Network for two processes (self-loops omitted).

\[q_0 \xrightarrow{W(1)} q_1 \xrightarrow{W(2)} q_2 \xrightarrow{W(2)} q_f\]

\[R(0) \xrightarrow{} q_1 \xrightarrow{} R(1) \xrightarrow{} q_2 \xrightarrow{} R(2) \xrightarrow{} q_f\]

\[=\text{⇒}\quad \text{From here, the process in } q_0 \text{ is trapped hence the other one is alone and will never reach } q_f.\]

\[=\text{⇒}\quad \text{From here, non-exhaustive construction.}\]
Back to the example

Network for two processes (self-loops omitted).

⇒ From here, the process in \( q_0 \) is trapped hence the other one is alone and will never reach \( q_f \).

⇒ From here, non-exhaustive construction.

⇒ With 2 processes, \( q_f \) reached with probability \( > 0 \) (but \( < 1! \))
Back to the example

Network for two processes (self-loops omitted).

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

$$\Rightarrow$$ From here, non-exhaustive construction.

$$\Rightarrow$$ With 2 processes, $q_f$ reached with probability $> 0$ (but $< 1$!)

$$\Rightarrow$$ $k = 1$ is a negative cut-off.
Other examples

Positive cut-off

“Filter” protocol $\mathcal{F}_n$ for $n > 0$. 
Other examples

Positive cut-off

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.
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.

$\implies$ Tight positive cut-off equal to $n$, i.e., linear in the protocol size.
Other examples

Lack of monotonicity for small network sizes

2 is a tight negative cut-off:
Additional processes can create new deadlocks!
Other examples

Lack of monotonicity for small network sizes

⇒ 2 is a tight negative cut-off:

Additional processes can create new deadlocks!

⇒ We need new techniques to detect such behaviors.
Existence of a cut-off

Main result

**Theorem**

For any register protocol $\mathcal{P}$ (possibly with a leader protocol $\mathcal{P}_l$) 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.
Existence of a cut-off

Main result

Theorem

For any register protocol $P$ (possibly with a leader protocol $P_1$) there always exists a cut-off for almost-sure reachability, whose value is at most doubly-exponential in the size of $P$. 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.
Existence of a cut-off

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

$\Rightarrow$ State $q_f$ is reached with probability 1 if and only if the network size is odd.
## Existence of a cut-off

### Partial order over configurations

<table>
<thead>
<tr>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(q_1, \mu_c, d) \preceq (q'_1, \mu'_c, d')$ iff</td>
</tr>
<tr>
<td>- $q_1 = q'_1$, $d = d'$</td>
</tr>
<tr>
<td>- $\mu_c \sqsubseteq \mu'_c$</td>
</tr>
<tr>
<td>- $\mu_c$ and $\mu'_c$ have the same supports</td>
</tr>
</tbody>
</table>
Existence of a cut-off

Partial order over configurations

**Definition**

\[(q_l, \mu_c, d) \preceq (q'_l, \mu'_c, d') \text{ iff}\]

- \(q_l = q'_l, d = d'\)
- \(\mu_c \sqsubseteq \mu'_c\)
- \(\mu_c \text{ and } \mu'_c \text{ have the same supports}\)

**Example**

Initial configurations \(U_0 = \text{upward-closure of } (q_{l,0}, \{q_{c,0}\}, d_0)\).

Objective \(U_f = \{\gamma = (q_l, \mu_c, d) \mid \mu_c(q_f) > 0\}\) is upward-closed.
Existence of a cut-off

Partial order over configurations

**Definition**

\[(q_l, \mu_c, d) \preceq (q'_l, \mu'_c, d') \text{ iff}
\]

- \(q_l = q'_l, \ d = d'
- \mu_c \sqsubseteq \mu'_c
- \mu_c \text{ and } \mu'_c \text{ have the same supports}

**Theorem**

\(\preceq\) is a well quasi-order

**Corollary**

*Upward-closed sets of configurations have finite bases.*
Existence of a cut-off

Monotonicity

---

**Definition**

A protocol is monotonous if for any transition $\gamma_1 \rightarrow \gamma_2$:

- if $\gamma'_1 \succeq \gamma_1$, then there exists $\gamma'_2 \succeq \gamma_2$ s.t. $\gamma'_1 \rightarrow^* \gamma'_2$;
- if $\gamma'_2 \succeq \gamma_2$, then there exists $\gamma'_1 \succeq \gamma_1$ s.t. $\gamma'_1 \rightarrow^* \gamma'_2$. 
Existence of a cut-off

Monotonicity

**Definition**

A protocol is monotonous if for any transition $\gamma_1 \rightarrow \gamma_2$:

- if $\gamma'_1 \succeq \gamma_1$, then there exists $\gamma'_2 \succeq \gamma_2$ s.t. $\gamma'_1 \rightarrow^* \gamma'_2$;
- if $\gamma'_2 \succeq \gamma_2$, then there exists $\gamma'_1 \succeq \gamma_1$ s.t. $\gamma'_1 \rightarrow^* \gamma'_2$.

**Theorem**

*Non-atomic register protocols are monotonous.*

$\leadsto$ **If two protocols are in the same state and one takes a transition, the second one can also perform this transition.**
Existence of a cut-off

Monotonicity

Theorem

*Non-atomic register protocols are monotonous.*

→ This result fails to hold for atomic read/write
Existence of a cut-off

Symbolic graph

**Definition**

Symbolic graph = graph of supports: \( G_P = (V, E) \) with

- \( V = \{ \overline{\gamma} = (q_l, \mu_c, d) \mid (q_l, \mu_c, d) \in \Gamma \} \)
- if \( \gamma \rightarrow \gamma' \), then \( \overline{\gamma} \rightarrow \overline{\gamma}' \).

**Theorem**

Let \( P \) be a non-atomic protocol.

- If \( s \rightarrow^k s' \) in \( G_P \), then there exist configurations \( \gamma \) and \( \gamma' \) s.t. \( s = \overline{\gamma}, s' = \overline{\gamma}', \gamma \rightarrow^* \gamma', \text{and } |\gamma| = |\gamma'| \leq k + |Q_c| \).
- If \( s \rightarrow^* s' \) in \( G_P \), then there is a path from \( s \) to \( s' \) of contributor-size in \( O(|Q_l| \cdot |Q_c|) \).

\[ \implies \text{Existence of a cut-off for } \mathbb{P}(\lnot \diamond q_f) > 0 \text{ is in NP.} \]
Existence of a cut-off

Symbolic graph

\[
\begin{align*}
q_0 \xrightarrow{R(0)} q_1 & \quad W(1) \quad W(2) \\
q_1 \xrightarrow{R(1)} q_2 & \\
q_2 \xrightarrow{R(2)} q_f & \quad W(2)
\end{align*}
\]
Existence of a cut-off

Symbolic graph
Existence of a cut-off

Symbolic graph

⇒ Symbolic graph not correct for almost-sure reachability!
Existence of a cut-off

Proof

Definition

Let $U$ and $U'$ upward-closed sets of configurations. $U$ is **ultimately included** in $U'$ (written $U \subseteq U'$) if there exists $N$ s.t.

$$\forall k > N. \ U \cap \Gamma_k \subseteq U'.$$
Existence of a cut-off

Proof

Definition

Let $U$ and $U'$ upward-closed sets of configurations. $U$ is ultimately included in $U'$ (written $U \subseteq U'$) if there exists $N$ s.t.

$$\forall k > N. \ U \cap \Gamma_k \subseteq U'.$$
Existence of a cut-off

Proof

**Definition**

Let $U$ and $U'$ upward-closed sets of configurations. $U$ is **ultimately included** in $U'$ (written $U \subseteq U'$) if there exists $N$ s.t.

$$\forall k > N. \ U \cap \Gamma_k \subseteq U'.$$
Existence of a cut-off

Proof

**Definition**

Let $U$ and $U'$ upward-closed sets of configurations. $U$ is *ultimately included* in $U'$ (written $U \subseteq U'$) if there exists $N$ s.t.

$$\forall k > N. \ U \cap \Gamma_k \subseteq U'.$$
Existence of a cut-off

Proof

Lemma

\[ \mathbb{P}_n(\diamond U_f) = 1 \iff \text{Post}^*[(\Gamma_n \cap U_0) \setminus U_f] \subseteq \text{Pre}^*(U_f) \]

\[ \iff \text{Post}^*(U_0 \setminus U_f) \subseteq \text{Pre}^*(U_f) \]
Existence of a cut-off

Proof

**Lemma**

\[
P_n(\diamond U_f) = 1 \iff \text{Post}^*[(\Gamma_n \cap U_0) \setminus U_f] \subseteq \text{Pre}^*(U_f)
\]

\[
\iff \text{Post}^*(U_0 \setminus U_f) \subseteq \text{Pre}^*(U_f)
\]

**Proof (existence of a cut-off).**

- if \( \text{Post}^*(U_0 \setminus U_f) \subseteq \text{Pre}^*(U_f) \) then \(|Q_c| \times |\text{Pre}^*(U_f)|\) is a positive cut-off.

- otherwise, there exists \( \gamma \in \text{Post}^*(U_0 \setminus U_f) \) and \( q \in \overline{\gamma} \) s.t. \( \gamma + k \cdot q \notin \text{Pre}^*(U_f) \) for all \( k \).

Then \( \gamma + (k - |\gamma|) \cdot q \in \text{Post}^*(U_0 \setminus U_f) \cap \Gamma_k \setminus \text{Pre}^*(U_f) \); hence \(|\gamma|\) is a negative cut-off.
Extended symbolic graph

Adding a concrete part

**Definition: symbolic graph of index $k$**

Given $\mathcal{P} = (\mathcal{P}_l, \mathcal{P}_c)$ and $k \in \mathbb{N}$, we define $\tilde{\mathcal{P}}^k = (\tilde{\mathcal{P}}^k_l, \mathcal{P}_c)$ with $\tilde{\mathcal{P}}^k_l = (Q'_l, D, q'_0, T'_1)$ defined as

- $Q'_l = Q_l \times \mathbb{N}_k^{Q_c}$,
- $q'_0 = (q_l, 0, \{q'_{c,0} \mapsto k\})$,
- $((q, \mu), A, d, (q', \mu')) \in T'_1$ if
  - $\mu = \mu'$ and $q \xrightarrow{A(d)} q'$;
  - $q = q'$ and $\mu \xrightarrow{A(d)} \mu'$.

Symbolic graph of index $k = $ symbolic graph of $\tilde{\mathcal{P}}^k$.

$\leftarrow$ Transitions impact either 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 $\text{Pre}^*(\lceil q_f \rceil) = \uparrow\{\eta_i \mid 1 \leq i \leq m\}$. We show that the symbolic graph abstraction is complete for $k = K \cdot |Q|$, where $K = \max\{st(\eta_i)(q) \mid q \in Q, 1 \leq i \leq m\}$.

$\implies$ Intuitively, the concrete part must be large enough to capture executions involving minimal elements of $\text{Pre}^*(\lceil q_f \rceil)$.
Symbolic graph

Toward a correct and complete algorithm

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

\[ \implies \text{Intuitively, the concrete part must be large enough to capture executions involving minimal elements of } \text{Pre}^*(\lfloor q_f \rfloor). \]

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 \( \langle (q_l,0, q_c,0), \{q_c,0\}, d_0 \rangle \) 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 \([\text{Rac78, DJLL13}]\), 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.
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 $\implies$ NEXPSpace w.r.t. the protocol $\implies$ EXPSPACE by Savitch’s theorem [Sip97].
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 $\Rightarrow$ NEXPSPACE w.r.t. the protocol $\Rightarrow$ 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 $q_{\text{halt}}$. 
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).

$\Rightarrow$ **Huge gap!**
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).

$\implies$ **Huge gap!**

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

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

\( \implies \) **Exponential tight negative cut-off.**
Exponential negative cut-off

Three phases: initialization, simulation, counting.
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 $q_f$. 
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 $tok$.

$\implies q_f$ 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.

⇒ 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 \( \text{sink} \). Since we have an \((n + 1)\) filter, \( q_f \) cannot be reached.

\[ \implies \mathbb{P}(\langle \text{init}^N, \# \rangle, [\Diamond q_f]) < 1 \text{ for } N = n + 2^n. \]
We have proved a tight negative cut-off of exponential size.
1. Networks of register protocols

2. Almost-sure reachability

3. Cut-offs: existence and decision algorithm

4. Conclusion
Summary

Our model:

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

⇒ **Slight changes in the setting induce important changes in complexity.**
Future work

Many open questions:

- closing the gaps (complexity, cut-off bounds),
Future work

Many open questions:

- closing the gaps (complexity, cut-off bounds),
- other objectives (e.g., liveness),
Future work

Many open questions:
- closing the gaps (complexity, cut-off bounds),
- other objectives (e.g., liveness),
- quantitative questions,
Future work

Many open questions:

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

Simon Außerlechner, Swen Jacobs, and Ayrat Khalimov.
Tight cutoffs for guarded protocols with fairness.

Nathalie Bertrand, Paulin Fournier, and Arnaud Sangnier.
Playing with probabilities in reconfigurable broadcast networks.

Antoine Durand-Gasselin, Javier Esparza, Pierre Ganty, and Rupak Majumdar.
Model checking parameterized asynchronous shared-memory systems.

Stéphane Demri, Marcin Jurdziński, Oded Lachish, and Ranko Lazić.
The covering and boundedness problems for branching vector addition systems.

Giorgio Delzanno, Arnaud Sangnier, Riccardo Traverso, and Gianluigi Zavattaro.
On the complexity of parameterized reachability in reconfigurable broadcast networks.
Javier Esparza, Pierre Ganty, and Rupak Majumdar.
Parameterized verification of asynchronous shared-memory systems.

Steven M. German and A. Prasad Sistla.
Reasoning about systems with many processes.

Matthew Hague.
Parameterised pushdown systems with non-atomic writes.

Charles Rackoff.
The covering and boundedness problems for vector addition systems.

Michael Sipser.
Introduction to the theory of computation.