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)

UNIVERSITÉ LIBRE DE BRUXELLES

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 *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).

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*.

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**.

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.

$\rightsquigarrow$ Exploit symmetries of such distributed systems for efficient 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*.

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).

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]).

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.

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

Our model in a nutshell

Scheduler

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

$\rightsquigarrow$ **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.

$\rightsquigarrow$ **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.

$\rightsquigarrow$ **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.

$\rightsquigarrow$ **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

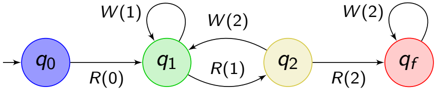
- stochastic protocols,
- communication via broadcast,
- a "helping scheduler".

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.

$\Longrightarrow$ 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 $D = \{0, 1, 2\}$.

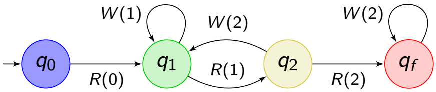
Definition: register protocol

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

- Q finite set of control locations;
- D finite alphabet of data for the shared register, 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

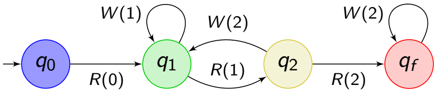
Example



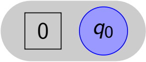
Imagine that our network contains a single process.

Our protocols

Example

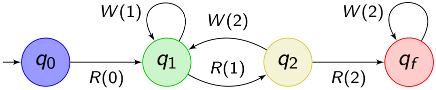


Imagine that our network contains a single process.

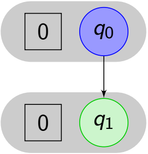


Our protocols

Example

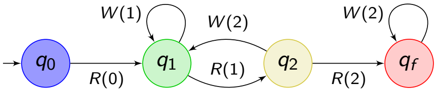


Imagine that our network contains a single process.

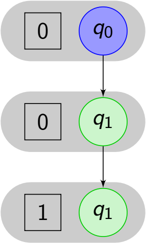


Our protocols

Example

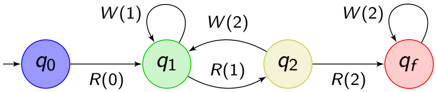


Imagine that our network contains a single process.

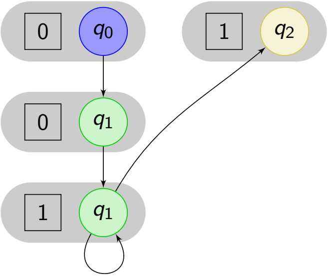


Our protocols

Example

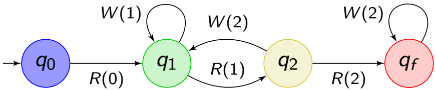


Imagine that our network contains a single process.

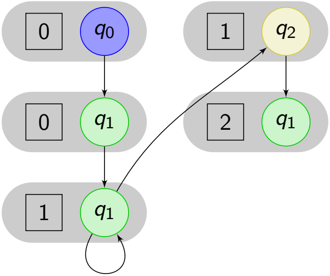


Our protocols

Example

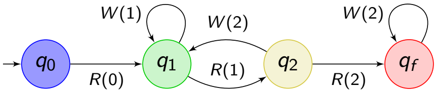


Imagine that our network contains a single process.

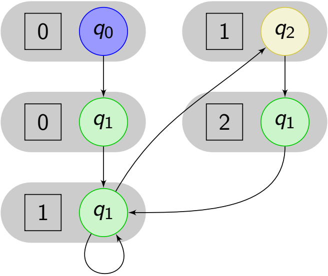


Our protocols

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**:

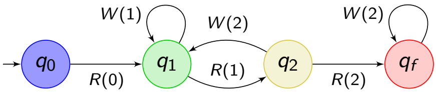
- asynchronous composition of k copies of the protocol,
- possibly one copy of a different protocol (leader),
- non-determinism (inside the protocols and choice of process) resolved by a stochastic scheduler (uniform).

▷ Markov chain over the set of **configurations** $\Gamma = Q_l \times \mathbb{N}^{Q_c} \times D$ (leader + multiset + data).

▷ finite if k is fixed; no creation/deletion of processes.

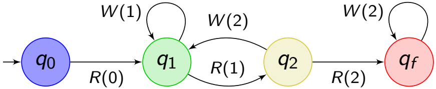
Our networks

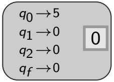
Semantics



Our networks

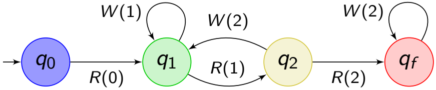
Semantics

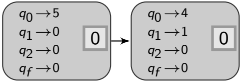




Our networks

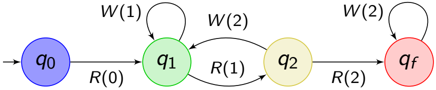
Semantics

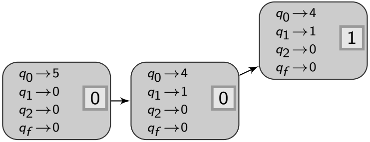




Our networks

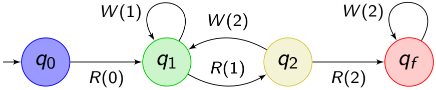
Semantics

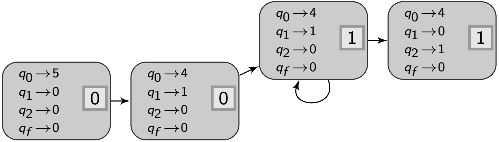




Our networks

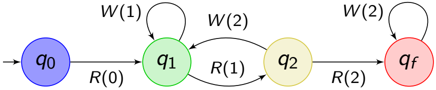
Semantics

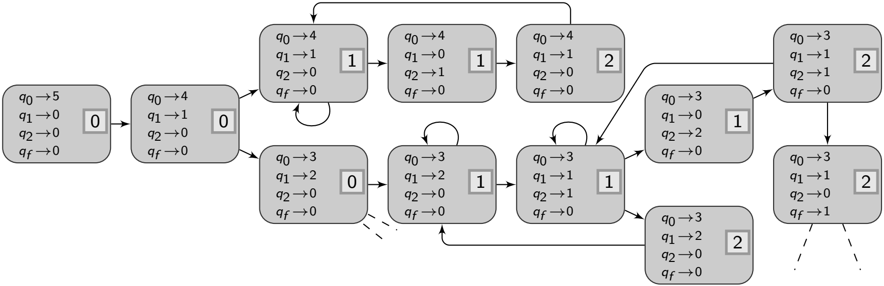




Our networks

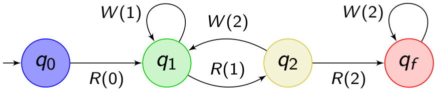
Semantics

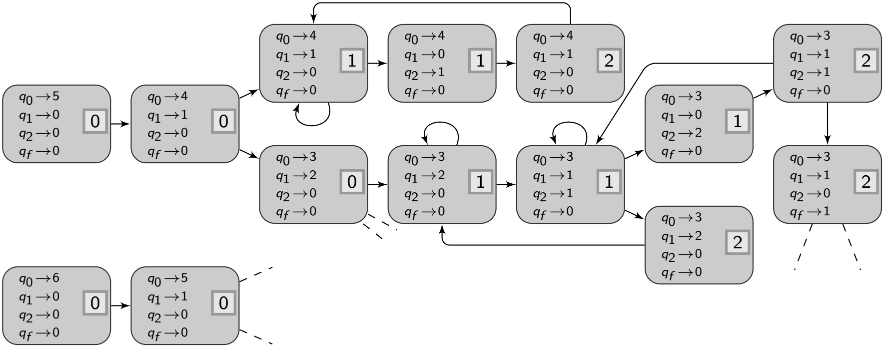




Our networks

Semantics





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 $\gamma_0 \to^* \gamma_n$ s.t. $\exists\, i \in [0; n]$, $st(\gamma_i)(q_f) > 0$.

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

$\rightsquigarrow$ **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 $\not\Rightarrow$ $\exists$ 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.

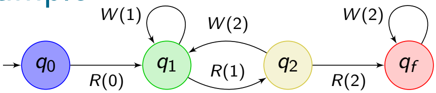
⚠ Cut-offs need not exist from the definition

and

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

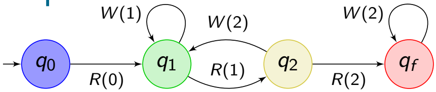
$\hookrightarrow$ **We will prove that they always exist!**

Back to the example

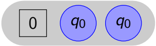


Network for two processes (self-loops omitted).

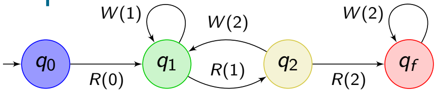
Back to the example



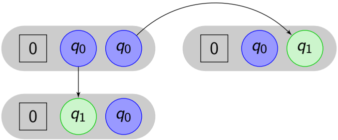
Network for two processes (self-loops omitted).



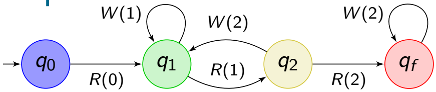
Back to the example



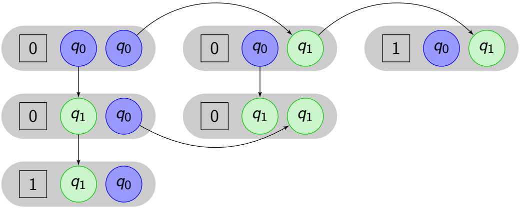
Network for two processes (self-loops omitted).



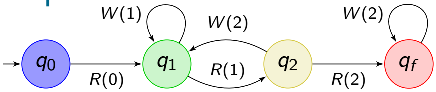
Back to the example



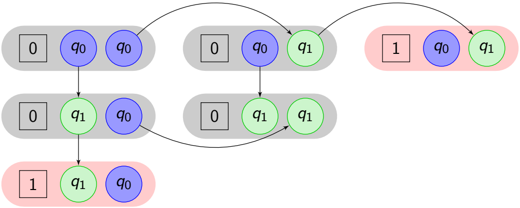
Network for two processes (self-loops omitted).



Back to the example

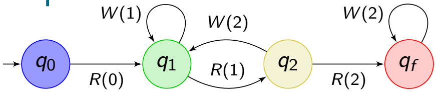


Network for two processes (self-loops omitted).

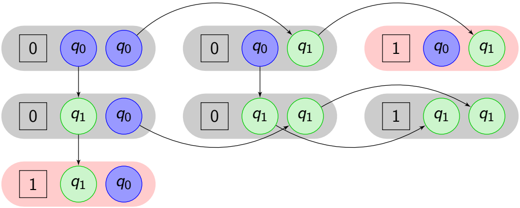


$\Longrightarrow$ **From here, the process in q_0 is trapped hence the other one is alone and will never reach q_f.**

Back to the example

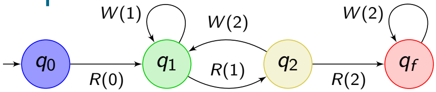


Network for two processes (self-loops omitted).

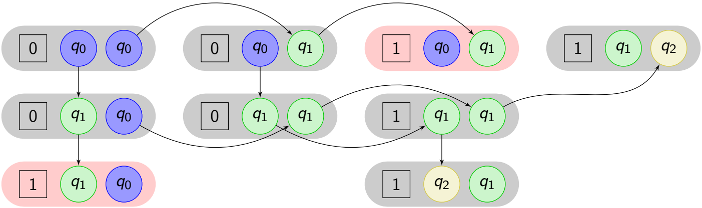


$\Longrightarrow$ **From here, the process in q_0 is trapped hence the other one is alone and will never reach q_f.**

Back to the example

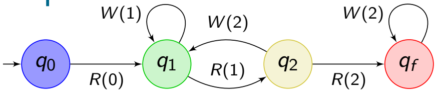


Network for two processes (self-loops omitted).

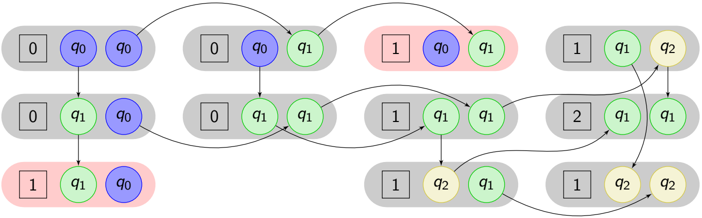


$\Longrightarrow$ **From here, the process in q_0 is trapped hence the other one is alone and will never reach q_f.**

Back to the example

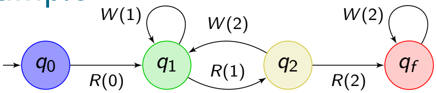


Network for two processes (self-loops omitted).

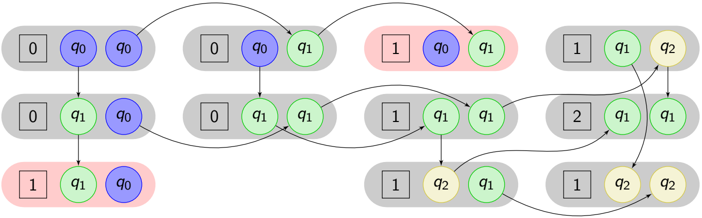


$\Longrightarrow$ **From here, the process in q_0 is trapped hence the other one is alone and will never reach q_f.**

Back to the example



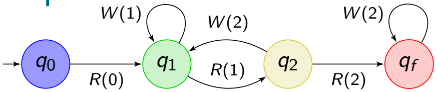
Network for two processes (self-loops omitted).



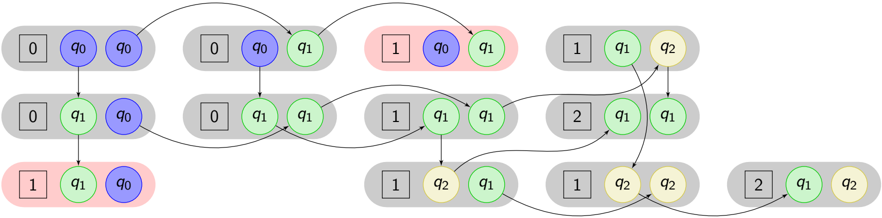
$\Longrightarrow$ **From here, the process in q_0 is trapped hence the other one is alone and will never reach q_f.**

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

Back to the example



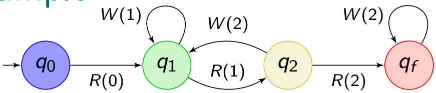
Network for two processes (self-loops omitted).



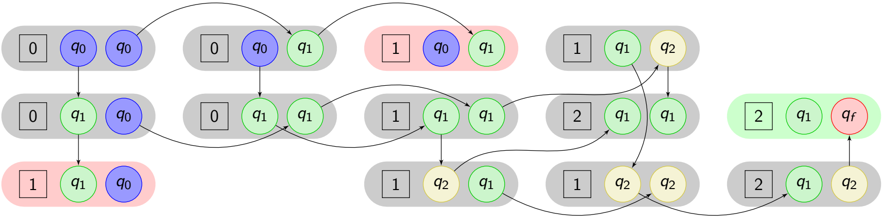
$\Longrightarrow$ **From here, the process in q_0 is trapped hence the other one is alone and will never reach q_f.**

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

Back to the example



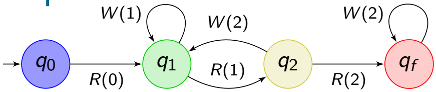
Network for two processes (self-loops omitted).



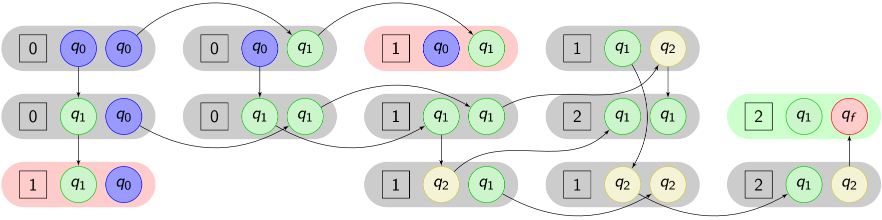
$\Longrightarrow$ **From here, the process in q_0 is trapped hence the other one is alone and will never reach q_f.**

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

Back to the example



Network for two processes (self-loops omitted).

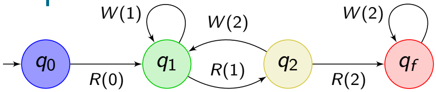


$\Longrightarrow$ **From here, the process in q_0 is trapped hence the other one is alone and will never reach q_f.**

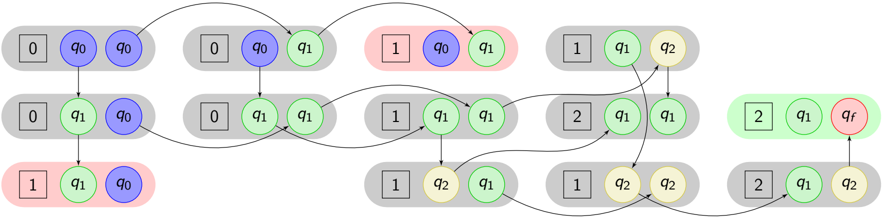
$\Longrightarrow$ **From here, non-exhaustive construction.**

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

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.**

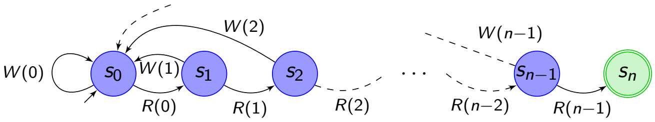
$\Longrightarrow$ **From here, non-exhaustive construction.**

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

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

Other examples

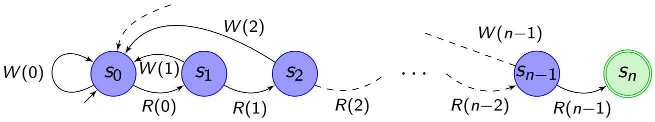
Positive cut-off



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

Other examples

Positive cut-off



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

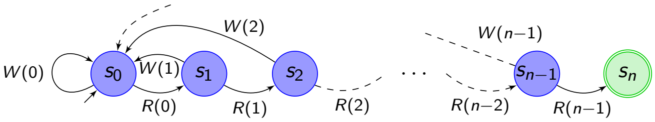
For protocol $\mathcal{F}_n$,

- 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.

Other examples

Positive cut-off



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

For protocol $\mathcal{F}_n$,

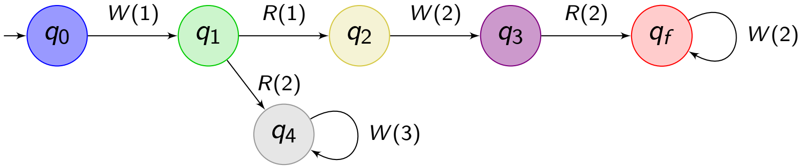
- ▷ networks of size $\geq n$ cover s_n with probability 1,
- ▷ networks of size $< n$ cannot cover s_n.

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

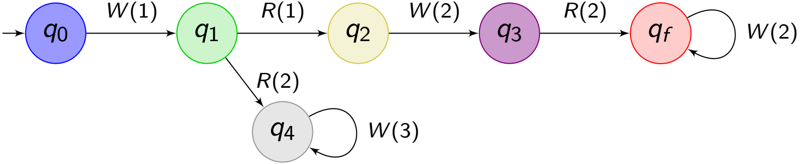


$\Longrightarrow$ 2 **is a tight negative cut-off:**

Additional processes can create new deadlocks!

Other examples

Lack of monotonicity for small network sizes



$\Longrightarrow$ 2 **is a tight negative cut-off:**

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 $\mathcal{P}$ (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

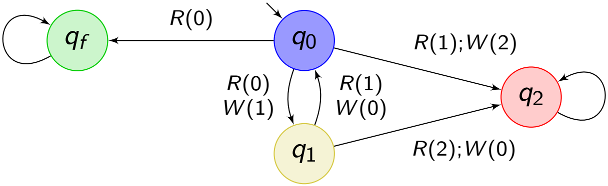
Theorem

For any register protocol $\mathcal{P}$ (possibly with a leader protocol $\mathcal{P}_\mathsf{l}$) **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.

⚠ **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 $\rightsquigarrow$ no cut-off



$\Longrightarrow$ **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

Definition

$(q_\mathsf{l}, \mu_\mathsf{c}, d) \preceq (q'_\mathsf{l}, \mu'_\mathsf{c}, d')$ iff

- $q_\mathsf{l} = q'_\mathsf{l}$, $d = d'$
- $\mu_\mathsf{c} \sqsubseteq \mu'_\mathsf{c}$
- **μ_c and μ'_c have the same supports**

Existence of a cut-off

Partial order over configurations

Definition

$(q_\mathsf{l}, \mu_\mathsf{c}, d) \preceq (q'_\mathsf{l}, \mu'_\mathsf{c}, d')$ iff

- $q_\mathsf{l} = q'_\mathsf{l}$, $d = d'$
- $\mu_\mathsf{c} \sqsubseteq \mu'_\mathsf{c}$
- **μ_c and μ'_c have the same supports**

Example

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

Objective $U_f = \{\gamma = (q_\mathsf{l}, \mu_\mathsf{c}, d) \mid \mu_\mathsf{c}(q_f) > 0\}$ is upward-closed.

Existence of a cut-off

Partial order over configurations

Definition

$(q_\mathsf{l}, \mu_\mathsf{c}, d) \preceq (q'_\mathsf{l}, \mu'_\mathsf{c}, d')$ iff

- $q_\mathsf{l} = q'_\mathsf{l}$, $d = d'$
- $\mu_\mathsf{c} \sqsubseteq \mu'_\mathsf{c}$
- **μ_c and μ'_c 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 $\gamma_1 \to \gamma_2$:

- if $\gamma_1' \succeq \gamma_1$, then there exists $\gamma_2' \succeq \gamma_2$ s.t. $\gamma_1' \to^* \gamma_2'$;
- if $\gamma_2' \succeq \gamma_2$, then there exists $\gamma_1' \succeq \gamma_1$ s.t. $\gamma_1' \to^* \gamma_2'$.

Existence of a cut-off

Monotonicity

Definition

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

- if $\gamma_1' \succeq \gamma_1$, then there exists $\gamma_2' \succeq \gamma_2$ s.t. $\gamma_1' \to^* \gamma_2'$;
- if $\gamma_2' \succeq \gamma_2$, then there exists $\gamma_1' \succeq \gamma_1$ s.t. $\gamma_1' \to^* \gamma_2'$.

Theorem

Non-atomic register potocols are monotonous.

$\rightsquigarrow$ **If two protocols are in the same state and one takes a transition, the second one can also perform this transition.**

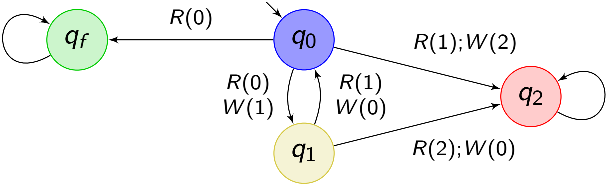
Existence of a cut-off

Monotonicity

Theorem

Non-atomic register potocols are monotonous.

$\Longrightarrow$ **This result fails to hold for atomic read/write**



Existence of a cut-off

Symbolic graph

Definition

Symbolic graph = **graph of supports**: $G_{\mathcal{P}} = (V, E)$ with

- $V = \{\overline{\gamma} = (q_{\mathsf{l}}, \overline{\mu}_{\mathsf{c}}, d) \mid (q_{\mathsf{l}}, \overline{\mu}_{\mathsf{c}}, d) \in \Gamma\}$
- if $\gamma \to \gamma'$, then $\overline{\gamma} \to \overline{\gamma}'$.

Theorem

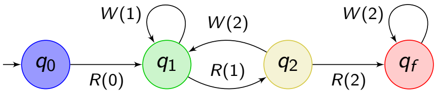
Let $\mathcal{P}$ be a non-atomic protocol.

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

$\Longrightarrow$ **Existence of a cut-off for $\mathbb{P}([\![\Diamond q_f]\!]) > 0$ is in NP.**

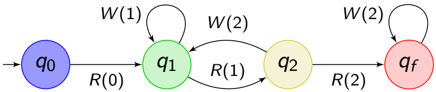
Existence of a cut-off

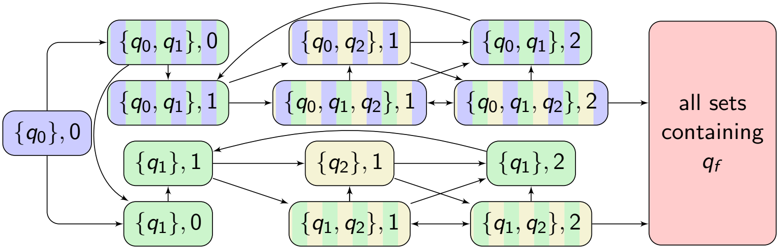
Symbolic graph



Existence of a cut-off

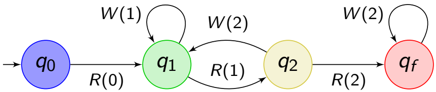
Symbolic graph

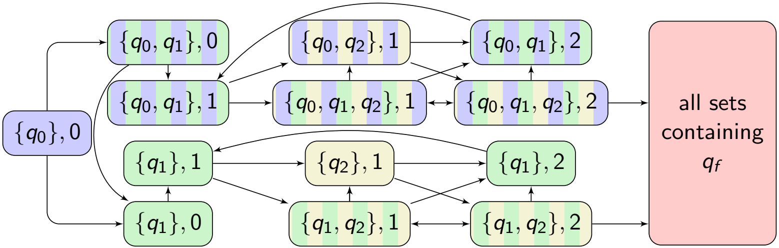




Existence of a cut-off

Symbolic graph





$\Longrightarrow$ **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 $U \sqsubseteq U'$) if there exists N s.t.

$$\forall k > N.\ U \cap \Gamma_k \subseteq U'.$$

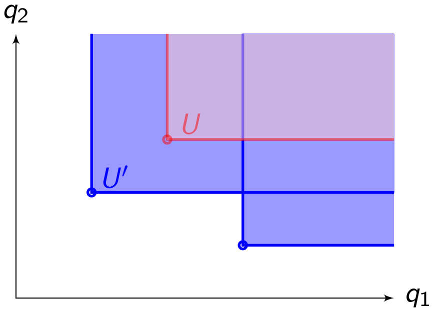
Existence of a cut-off

Proof

Definition

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

$$\forall k > N.\ U \cap \Gamma_k \subseteq U'.$$



$U \sqsubseteq U'$

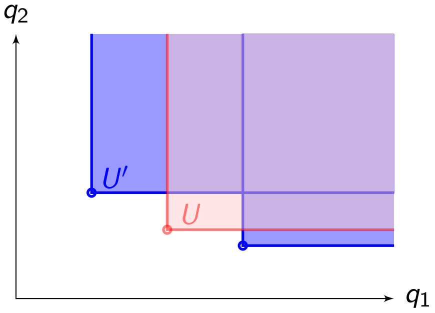
Existence of a cut-off

Proof

Definition

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

$$\forall k > N.\ U \cap \Gamma_k \subseteq U'.$$



$U \sqsubseteq U'$

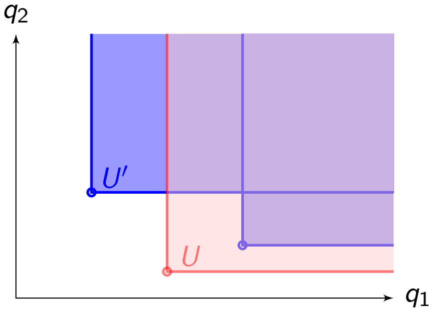
Existence of a cut-off

Proof

Definition

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

$$\forall k > N.\ U \cap \Gamma_k \subseteq U'.$$



$U \not\sqsubseteq U'$

Existence of a cut-off

Proof

Lemma

$$\mathbb{P}_n(\Diamond U_f) = 1 \quad \Leftrightarrow \quad \mathrm{Post}^*[(\Gamma_n \cap U_0) \setminus U_f] \subseteq \mathrm{Pre}^*(U_f)$$
$$\Leftrightarrow \quad \mathrm{Post}^*(U_0 \setminus U_f) \sqsubseteq \mathrm{Pre}^*(U_f)$$

Existence of a cut-off

Proof

Lemma

$$\mathbb{P}_n(\Diamond U_f) = 1 \quad \Leftrightarrow \quad \mathrm{Post}^*[(\Gamma_n \cap U_0) \setminus U_f] \subseteq \mathrm{Pre}^*(U_f)$$
$$\Leftrightarrow \quad \mathrm{Post}^*(U_0 \setminus U_f) \sqsubseteq \mathrm{Pre}^*(U_f)$$

Proof (existence of a cut-off).

- if $\mathrm{Post}^*(U_0 \setminus U_f) \sqsubseteq \mathrm{Pre}^*(U_f)$ then $|Q_c| \times |\mathrm{Pre}^*(U_f)|$ is a positive cut-off.
- otherwise, there exists $\gamma \in \mathrm{Post}^*(U_0 \setminus U_f)$ and $q \in \overline{\gamma}$ s.t. $\gamma + k \cdot q \notin \mathrm{Pre}^*(U_f)$ for all k.
  Then $\gamma + (k - |\gamma|) \cdot q \in \mathrm{Post}^*(U_0 \setminus U_f) \cap \Gamma_k \setminus \mathrm{Pre}^*(U_f)$; hence $|\gamma|$ is a negative cut-off.

□

Extended symbolic graph

Adding a concrete part

Definition: symbolic graph of index k

Given $\mathcal{P} = (\mathcal{P}_\mathsf{l}, \mathcal{P}_\mathsf{c})$ and $k \in \mathbb{N}$, we define $\widetilde{\mathcal{P}}^k = (\widetilde{\mathcal{P}}_\mathsf{l}^k, \mathcal{P}_\mathsf{c})$ with $\widetilde{\mathcal{P}}_\mathsf{l}^k = (Q_\mathsf{l}', D, q_0', T_\mathsf{l}')$ defined as

- $Q_\mathsf{l}' = Q_\mathsf{l} \times \mathbb{N}_k^{Q_\mathsf{c}}$,
- $q_0' = (q_{\mathsf{l},0}, \{q'_{\mathsf{c},0} \mapsto k\})$,
- $((q, \mu), A, d, (q', \mu')) \in T_\mathsf{l}'$ 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 $\widetilde{\mathcal{P}}^k$.

$\hookrightarrow$ 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 $\mathrm{Pre}^*(\llbracket q_f \rrbracket) = \uparrow\{\eta_i \mid 1 \leq i \leq m\}$. 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 $\mathrm{Pre}^*(\llbracket q_f \rrbracket)$.**

Symbolic graph

Toward a correct and complete algorithm

Recall that $\mathrm{Pre}^*([\![q_f]\!]) = \uparrow\{\eta_i \mid 1 \leq i \leq m\}$. 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 $\mathrm{Pre}^*([\![q_f]\!])$.**

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 $\langle (q_{\mathsf{l},0}, q_{\mathsf{c},0}^{K\cdot|Q|}), \{q_{\mathsf{c},0}\}, d_0\rangle$ 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 [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 $\Longrightarrow$ 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}.

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!**

Complexity (2/2)

Lower bounds

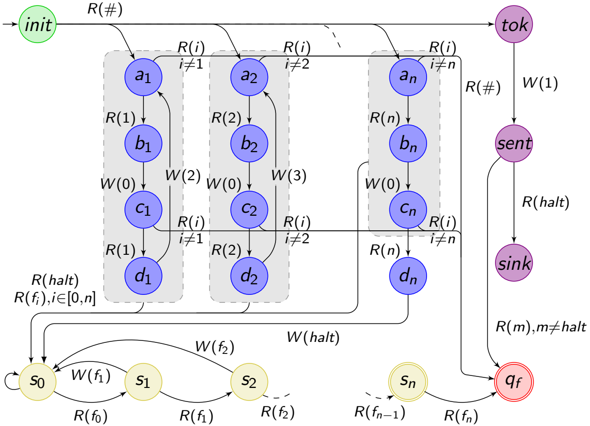
- PSPACE-hardness via linear-bounded Turing machine [Sip97]: we build a protocol for which there is a negative cut-off iff the machine reaches its final state q_{halt}.
- Best lower bound for positive cut-offs so far: linear (cf. "filter" protocol).

$\Longrightarrow$ **Huge gap!**

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

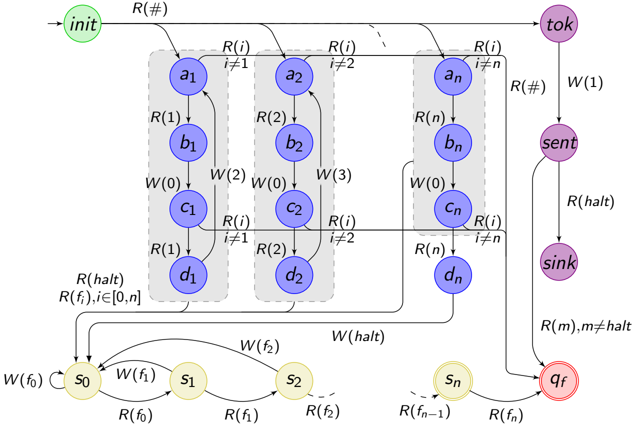
$\Longrightarrow$ **Shares ideas with PSPACE-hardness proof. Let's discuss it now.**

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.

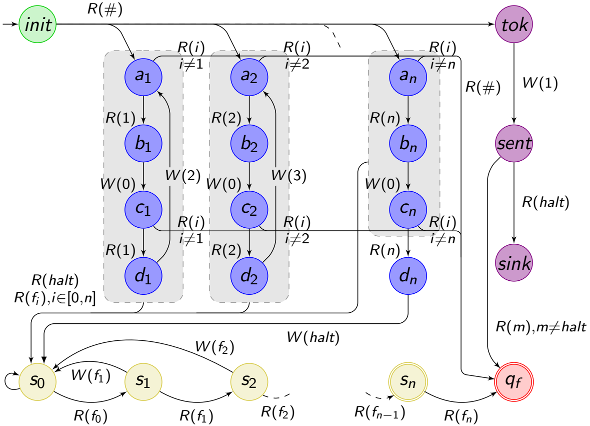
Exponential negative cut-off



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

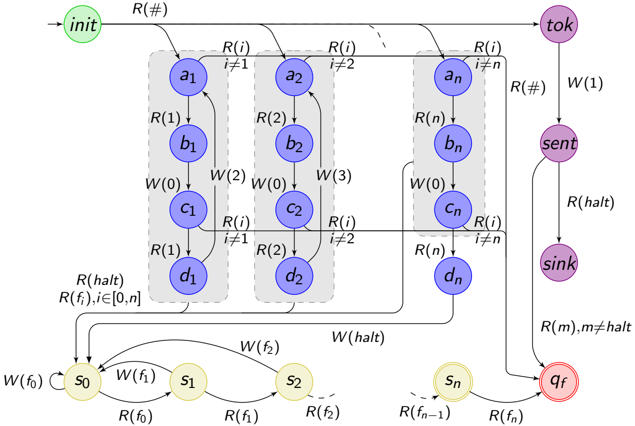
$\Longrightarrow$ **Exponential tight negative cut-off.**

Exponential negative cut-off



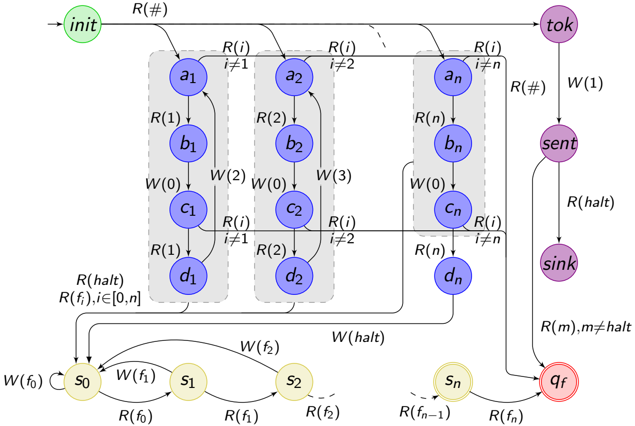
Three phases: initialization, simulation, counting.

Exponential negative cut-off



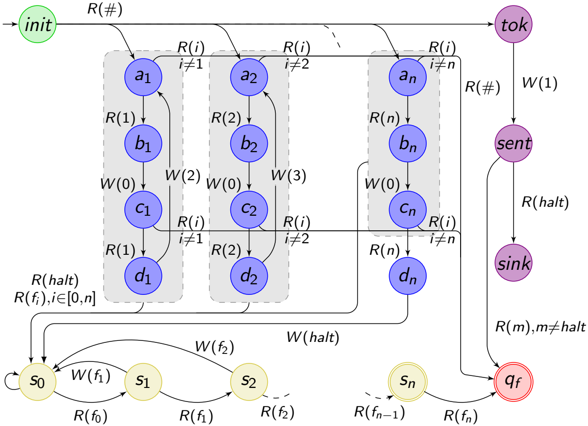
Phase 1: initialization. Processes move to a_i and *tok* until some process in *tok* writes 1 in the register (or until someone reaches q_f by reading # from a_i).

Exponential negative cut-off



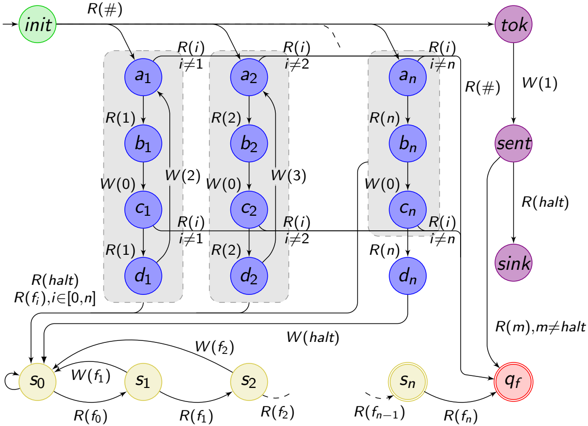
Phase 2: simulation. If all the processes are in *tok*, they will eventually reach q_f. So we assume that there is at least one process in a state a_i.

Exponential negative cut-off



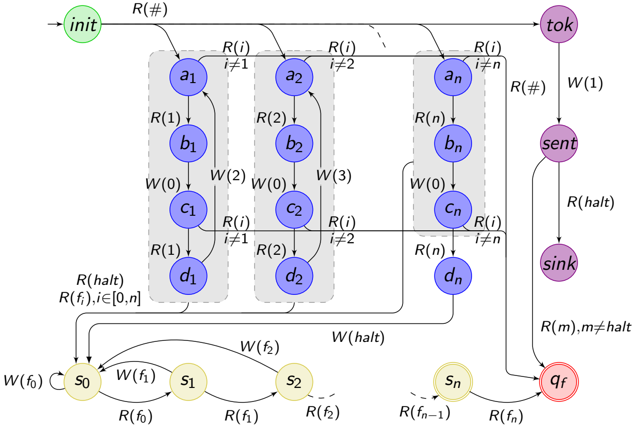
If some a_i is empty, then d_n cannot be reached and we cannot enter the counting phase $\implies$ some process will eventually reach q_f.

Exponential negative cut-off



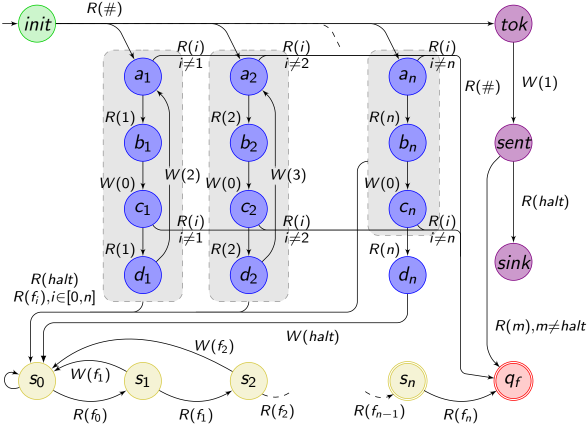
Thus, assume there is at least one process in each state a_i. We can prove that d_i is reachable when at the start of the simulation phase, at least 2^i processes are in *tok* (we need to produce an exponential number of tokens).

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.

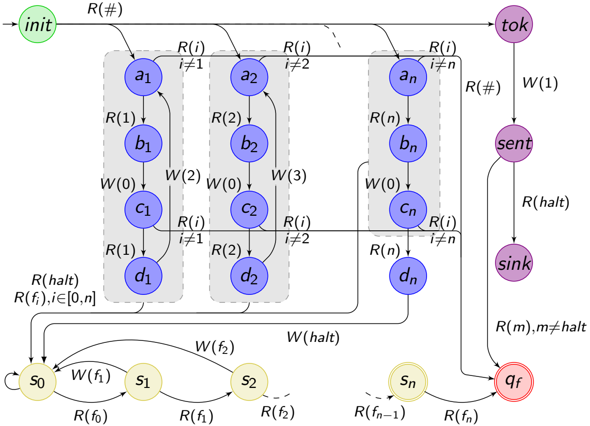
Exponential negative cut-off



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.**

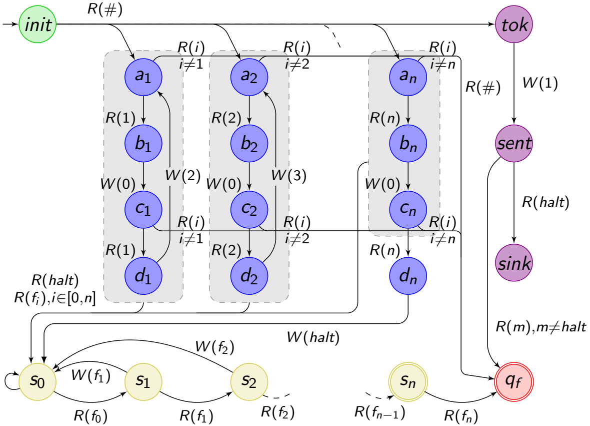
Exponential negative cut-off



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

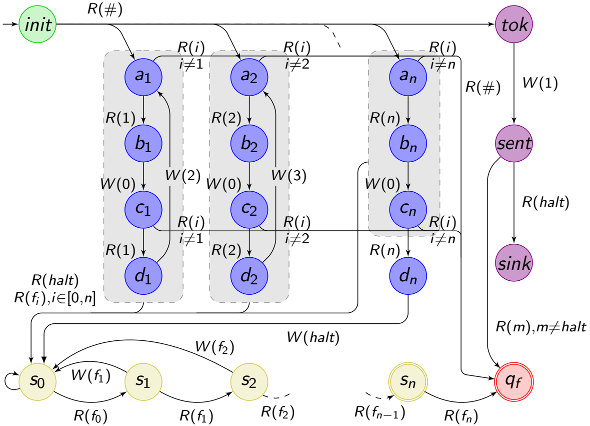
$\Longrightarrow$ **Exhibit a finite execution having no continuation reaching q_f.**

Exponential negative cut-off



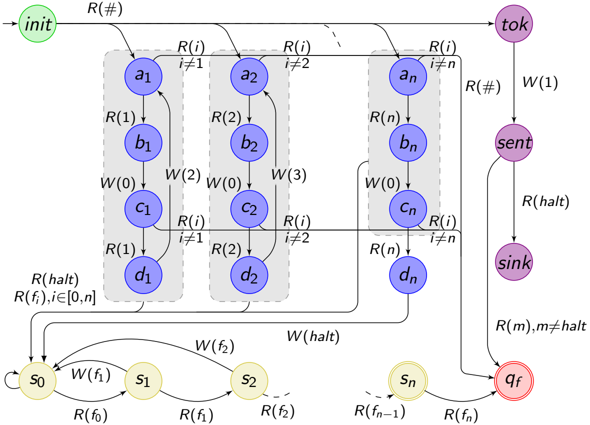
Execution: during initialization, put one process in each a_i and all others in *tok*. One of them writes 1.

Exponential negative cut-off



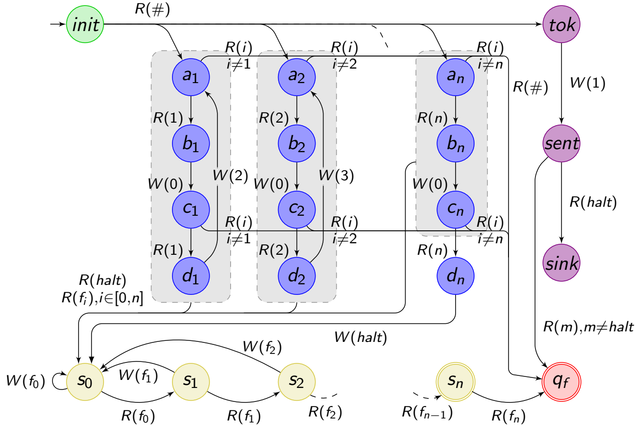
The n processes in states a_i then simulate the incrementations of the counter, consuming tokens at each step, until reaching d_n.

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*.

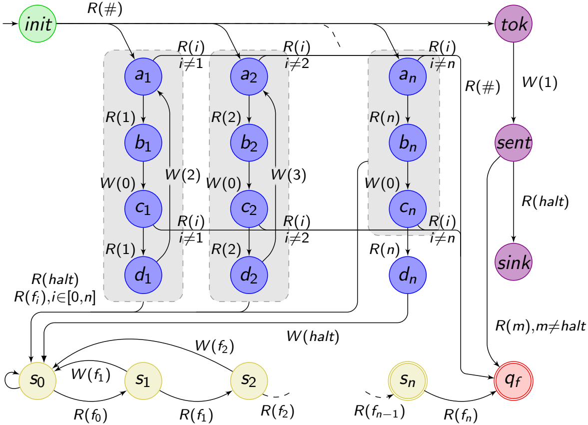
Exponential negative cut-off



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.

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

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.

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),

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.
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.

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.

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 II

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.

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

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.