Fair Termination for Parameterized Probabilistic Concurrent Systems

Ondřej Lengál[1] Anthony W. Lin[2]
Rupak Majumdar[3] Philipp Rümmer[4]

[1]Brno University of Technology, Czech Republic
[2]Department of Computer Science, University of Oxford, UK
[3]MPI-SWS Kaiserslautern, Germany
[4]Uppsala University, Sweden

26 April 2017 (TACAS'17)

Outline

- Parameterized probabilistic concurrent systems

Outline

- Parameterized probabilistic concurrent systems
- Liveness

Outline

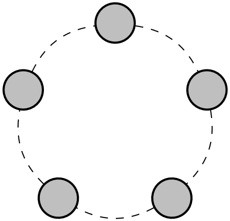
- Parameterized probabilistic concurrent systems
- Liveness
- Fairness

Outline

- Parameterized probabilistic concurrent systems
- Liveness
- Fairness
- Regular model checking

Motivating Example

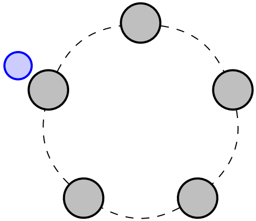
Herman's protocol (merging version)



Motivating Example

Herman's protocol (merging version)

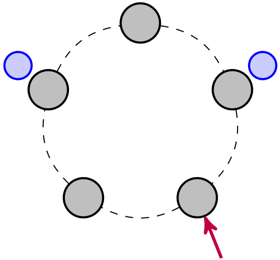
- ring topology, leader election



Motivating Example

Herman's protocol (merging version)

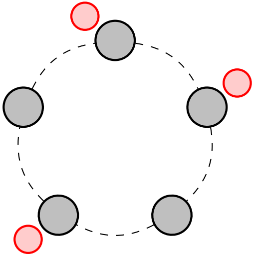
- ring topology, leader election
- scheduler selects processes



Motivating Example

Herman's protocol (merging version)

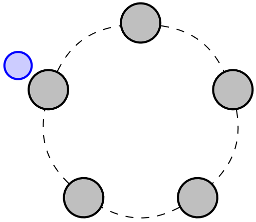
- ring topology, leader election
- scheduler selects processes
- **unstable** configuration:
  - >1 tokens



Motivating Example

Herman's protocol (merging version)

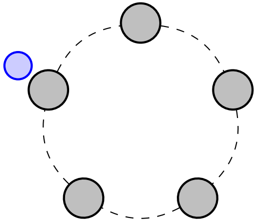
- ring topology, leader election
- scheduler selects processes
- **unstable** configuration:
  - >1 tokens
- **stable** configuration:
  - 1 token (**leader**)



Motivating Example

Herman's protocol (merging version)

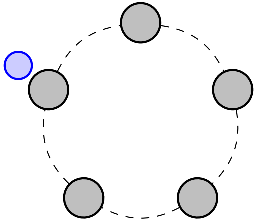
- ring topology, leader election
- scheduler selects processes
- **unstable** configuration:
  - >1 tokens
- **stable** configuration:
  - 1 token (**leader**)
- **goal**: $\models \Diamond$**leader** is elected



Motivating Example

Herman's protocol (merging version)

- ring topology, leader election
- scheduler selects processes
- **unstable** configuration:
  - $>$ 1 tokens
- **stable** configuration:
  - 1 token (**leader**)
- **goal**: $\models \Diamond$**leader** is elected

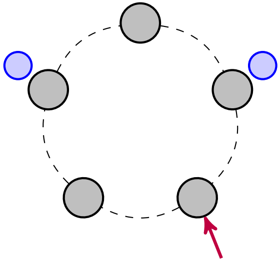


Herman's algorithm:

Motivating Example

Herman's protocol (merging version)

- ring topology, leader election
- scheduler selects processes
- **unstable** configuration:
  - $>$ 1 tokens
- **stable** configuration:
  - 1 token (**leader**)
- **goal**: $\models \Diamond$**leader** is elected



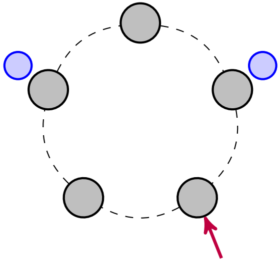
Herman's algorithm:

- when **selected**:

Motivating Example

Herman's protocol (merging version)

- ring topology, leader election
- scheduler selects processes
- **unstable** configuration:
  - > 1 tokens
- **stable** configuration:
  - 1 token (**leader**)
- **goal**: $\models \Diamond$**leader** is elected



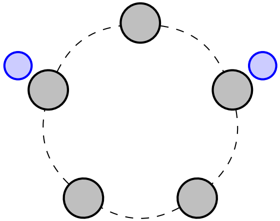
Herman's algorithm:

- when **selected**:
  - if no **token**:

Motivating Example

Herman's protocol (merging version)

- ring topology, leader election
- scheduler selects processes
- **unstable** configuration:
  - >1 tokens
- **stable** configuration:
  - 1 token (**leader**)
- **goal**: $\models \Diamond$**leader** is elected



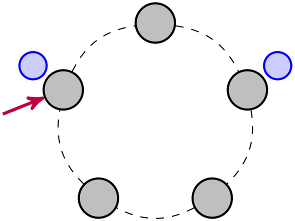
Herman's algorithm:

- when **selected**:
  - if no **token**: return

Motivating Example

Herman's protocol (merging version)

- ring topology, leader election
- scheduler selects processes
- **unstable** configuration:
  - $>$ 1 tokens
- **stable** configuration:
  - 1 token (**leader**)
- **goal**: $\models \Diamond$**leader** is elected



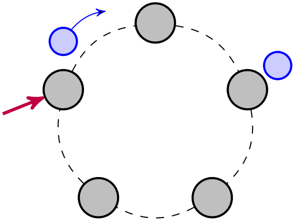
Herman's algorithm:

- when **selected**:
  - if no **token**: return
  - if has **token**: flip a coin

Motivating Example

Herman's protocol (merging version)

- ring topology, leader election
- scheduler selects processes
- **unstable** configuration:
  - > 1 tokens
- **stable** configuration:
  - 1 token (**leader**)
- **goal**: $\models \Diamond$**leader** is elected



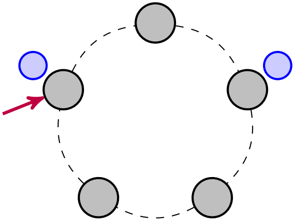
Herman's algorithm:

- when **selected**:
  - if no **token**: return
  - if has **token**: flip a coin
    - **heads**: pass the token clockwise

Motivating Example

Herman's protocol (merging version)

- ring topology, leader election
- scheduler selects processes
- **unstable** configuration:
  - > 1 tokens
- **stable** configuration:
  - 1 token (**leader**)
- **goal**: $\models \Diamond$**leader** is elected



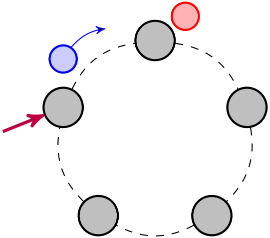
Herman's algorithm:

- when **selected**:
  - if no **token**: return
  - if has **token**: flip a coin
    - **heads**: pass the token clockwise
    - **tails**: keep the token

Motivating Example

Herman's protocol (merging version)

- ring topology, leader election
- scheduler selects processes
- **unstable** configuration:
  - >1 tokens
- **stable** configuration:
  - 1 token (**leader**)
- **goal**: $\models \Diamond$**leader** is elected



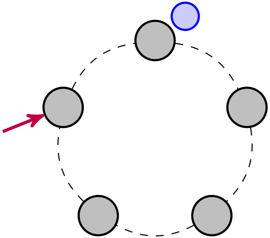
Herman's algorithm:

- when **selected**:
  - if no **token**: return
  - if has **token**: flip a coin
    - **heads**: pass the token clockwise
    - **tails**: keep the token
- if a process with a token gets another one:

Motivating Example

Herman's protocol (merging version)

- ring topology, leader election
- scheduler selects processes
- **unstable** configuration:
  - $>$ 1 tokens
- **stable** configuration:
  - 1 token (**leader**)
- **goal**: $\models \Diamond$**leader** is elected



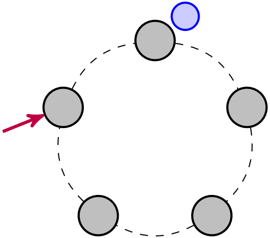
Herman's algorithm:

- when **selected**:
  - if no **token**: return
  - if has **token**: flip a coin
    - **heads**: pass the token clockwise
    - **tails**: keep the token
- if a process with a token gets another one: **merge** them

Motivating Example

Herman's protocol (merging version)

- ring topology, leader election
- scheduler selects processes
- **unstable** configuration:
  - $>$ 1 tokens
- **stable** configuration:
  - 1 token (**leader**)
- **goal**: $\models \Diamond$**leader** is elected



Herman's algorithm:

- when **selected**:
  - if no **token**: return
  - if has **token**: flip a coin
    - **heads**: pass the token clockwise
    - **tails**: keep the token
- if a process with a token gets another one: **merge** them

$$\Pr(\models \Diamond \textbf{leader} \text{ is elected}) = 1$$

Motivating Example

Herman's protocol (merging version)

Motivating Example

Herman's protocol (merging version)

$\Pr(\models \Diamond$**leader** is elected$) = 1$

Motivating Example

Herman's protocol (merging version)

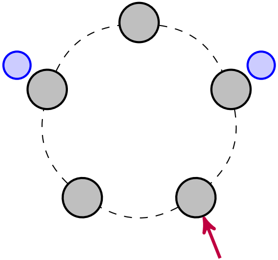
$\Pr(\models \Diamond$**leader** is elected$) = 1$

- really?

Motivating Example

Herman's protocol (merging version)

$\mathrm{Pr}(\models \Diamond$**leader** is elected$) = 1$



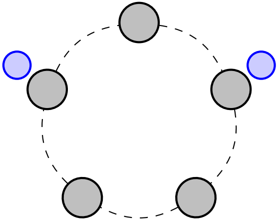
- really?

Motivating Example

Herman's protocol (merging version)

$\Pr(\models \Diamond$**leader** is elected$) = 1$

- really?

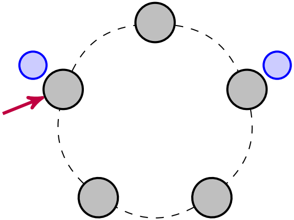


Motivating Example

Herman's protocol (merging version)

$\Pr(\models \Diamond$**leader** is elected$) = 1$

- really?

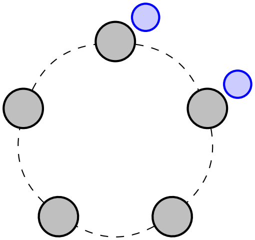


Motivating Example

Herman's protocol (merging version)

$\Pr(\models \Diamond$**leader** is elected$) = 1$

- really?

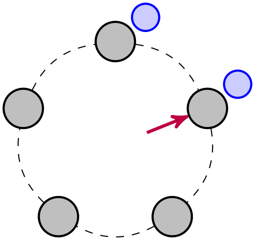


Motivating Example

Herman's protocol (merging version)

$\Pr(\models \Diamond$**leader** is elected$) = 1$

- really?

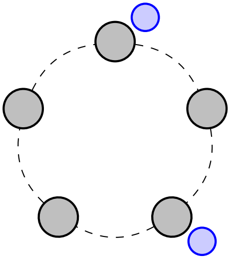


Motivating Example

Herman's protocol (merging version)

$\Pr(\models \Diamond$**leader** is elected$) = 1$

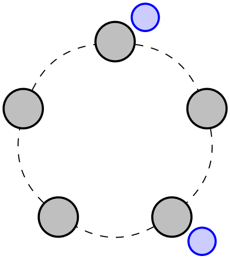
- really?



Motivating Example

Herman's protocol (merging version)

$\Pr(\models \Diamond$**leader** is elected$) = 1$

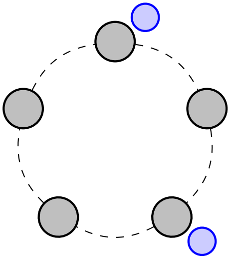


- really?
- **Fairness** needed!

Motivating Example

Herman's protocol (merging version)

$\Pr(\models \Diamond$**leader** is elected$) = 1$

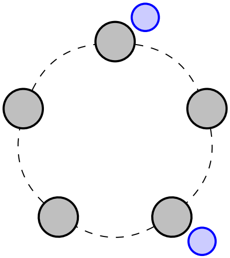


- really?
- **Fairness** needed!
- But **which** fairness?

Motivating Example

Herman's protocol (merging version)

$\Pr(\models \Diamond$**leader** is elected$) = 1$



- really?
- **Fairness** needed!
- But **which** fairness?
- We use **finitary** fairness

Setting

- Liveness of Fair Parameterized Probabilistic Concurrent Systems

Setting

- Liveness of Fair Parameterized Probabilistic Concurrent Systems
  - Parameterized Concurrent Systems: N finite-state processes

Setting

- Liveness of Fair Parameterized Probabilistic Concurrent Systems
  - Parameterized Concurrent Systems: N finite-state processes
  - Probabilistic: each process can flip a coin

Setting

- Liveness of Fair Parameterized Probabilistic Concurrent Systems
  - Parameterized Concurrent Systems: N finite-state processes
  - Probabilistic: each process can flip a coin
  - Fair: each process will have the opportunity to move

Setting

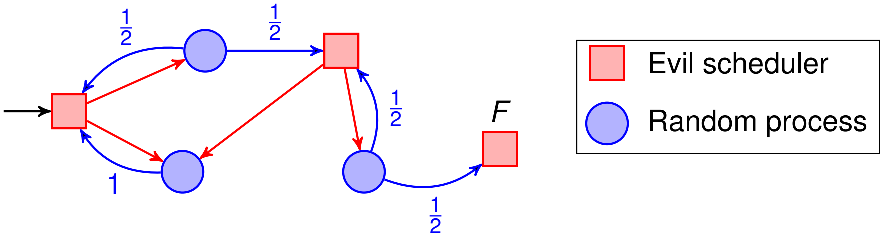
- Liveness of Fair Parameterized Probabilistic Concurrent Systems
  - Parameterized Concurrent Systems: *N* finite-state processes
  - Probabilistic: each process can flip a coin
  - Fair: each process will have the opportunity to move
  - Liveness: a **good** configuration is always reachable with $\Pr = 1$

Setting

- Liveness of Fair Parameterized Probabilistic Concurrent Systems
  - Parameterized Concurrent Systems: N finite-state processes
  - Probabilistic: each process can flip a coin
  - Fair: each process will have the opportunity to move
  - Liveness: a **good** configuration is always reachable with $\Pr = 1$
- Examples: Herman's protocol, Israeli-Jalfon protocol, population protocols, ...

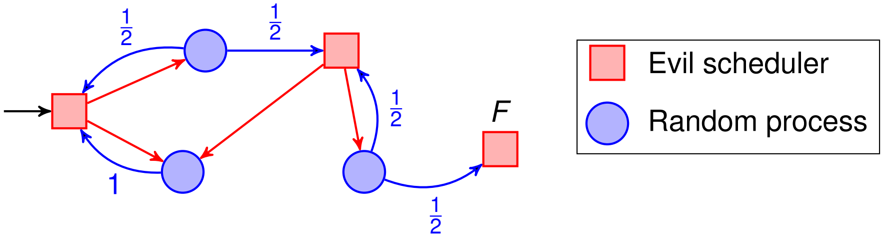
Setting

- Liveness of Fair Parameterized Probabilistic Concurrent Systems
  - Parameterized Concurrent Systems: N finite-state processes
  - Probabilistic: each process can flip a coin
  - Fair: each process will have the opportunity to move
  - Liveness: a **good** configuration is always reachable with $\Pr = 1$
- Examples: Herman's protocol, Israeli-Jalfon protocol, population protocols, ...
- An infinite-state Markov Decision Process (MDP)



Setting

- Liveness of Fair Parameterized Probabilistic Concurrent Systems
  - Parameterized Concurrent Systems: N finite-state processes
  - Probabilistic: each process can flip a coin
  - Fair: each process will have the opportunity to move
  - Liveness: a **good** configuration is always reachable with $\Pr = 1$
- Examples: Herman's protocol, Israeli-Jalfon protocol, population protocols, ...
- An infinite-state Markov Decision Process (MDP)



- $\Pr(s_0 \models \Diamond F) \stackrel{?}{=} 1$

Almost-Sure Liveness

Weakly-finite MDPs:

- for a fixed initial configuration, the set of reachable states is finite

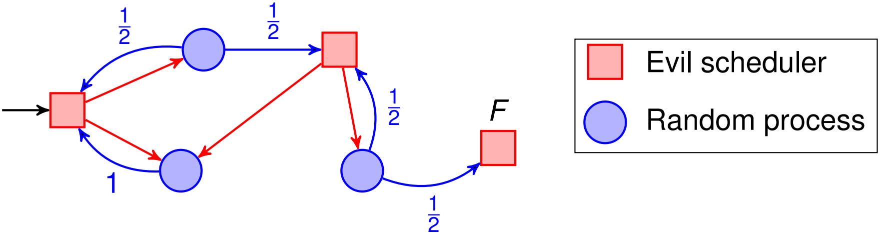
Almost-Sure Liveness

Weakly-finite MDPs:

- for a fixed initial configuration, the set of reachable states is finite

Almost-sure liveness in weakly-finite MDPs:

- only distinguish $= 0$ and > 0 transitions



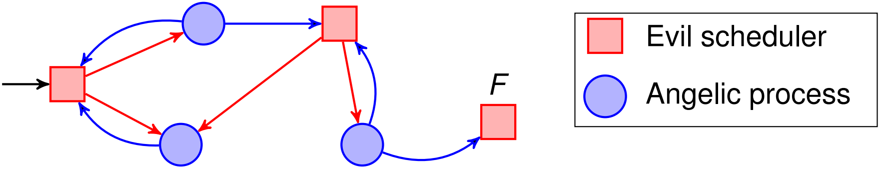
Almost-Sure Liveness

Weakly-finite MDPs:

- for a fixed initial configuration, the set of reachable states is finite

Almost-sure liveness in weakly-finite MDPs:

- only distinguish $= 0$ and > 0 transitions



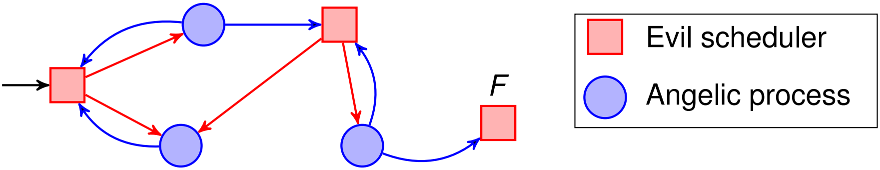
Almost-Sure Liveness

Weakly-finite MDPs:

- for a fixed initial configuration, the set of reachable states is finite

Almost-sure liveness in weakly-finite MDPs:

- only distinguish $= 0$ and > 0 transitions



Lemma

$\Pr(s_0 \models \Diamond F) = 1$ *iff Proc. has winning strategy from all* $s \in Reach(s_0)$.

Symbolic Framework: Regular Model Checking

Regular Model Checking for liveness in param. prob. conc. systems under all schedulers

Symbolic Framework: Regular Model Checking

Regular Model Checking for liveness in param. prob. conc. systems under all schedulers

- **Regular Model Checking**: Uppsala & Paris
  - Bouajjani, Jonsson, Nilsson, and Touili [CAV'00]

Symbolic Framework: Regular Model Checking

Regular Model Checking for liveness in param. prob. conc. systems under all schedulers

- **Regular Model Checking**: Uppsala & Paris
  - Bouajjani, Jonsson, Nilsson, and Touili [CAV'00]
  - usually safety of deterministic systems

Symbolic Framework: Regular Model Checking

Regular Model Checking for liveness in param. prob. conc. systems under all schedulers

- **Regular Model Checking**: Uppsala & Paris
  - Bouajjani, Jonsson, Nilsson, and Touili [CAV'00]
  - usually safety of deterministic systems
- liveness in parameterized probabilistic concurrent systems:
  - extension of Lin & Rümmer [CAV'16]

Symbolic Framework: Regular Model Checking

Regular Model Checking for liveness in param. prob. conc. systems under all schedulers

- **Regular Model Checking**: Uppsala & Paris
  - Bouajjani, Jonsson, Nilsson, and Touili [CAV'00]
  - usually safety of deterministic systems
- liveness in parameterized probabilistic concurrent systems:
  - extension of Lin & Rümmer [CAV'16]
- **this talk**: embedding of **fairness** into the system

Symbolic Framework: Regular Model Checking

Regular Model Checking

Symbolic Framework: Regular Model Checking

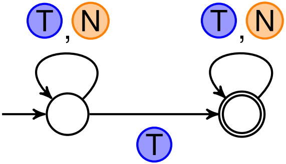
Regular Model Checking

- A configuration: a word over Σ: T N T N N

Symbolic Framework: Regular Model Checking

Regular Model Checking

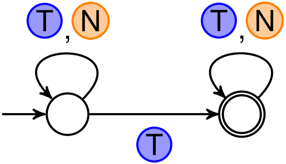
- A configuration: a word over Σ: T N T N N
- A set of configurations: a **finite automaton** *A* over Σ



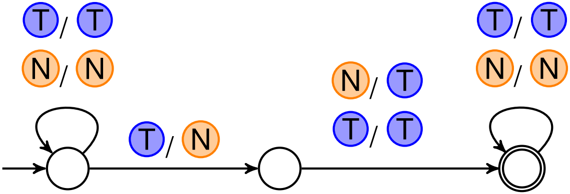
Symbolic Framework: Regular Model Checking

Regular Model Checking

- A configuration: a word over Σ: T N T N N
- A set of configurations: a **finite automaton** A over Σ



- Transition relation: a (length-preserving) **transducer** τ



Symbolic Framework: Regular Model Checking

Regular Model Checking for 2-player reachability games:

Symbolic Framework: Regular Model Checking

Regular Model Checking for 2-player reachability games:

- Liveness:
  - *Start*, *Good*, τ_1, and τ_2 given

Regular Model Checking for 2-player reachability games:

- Liveness:
  - *Start*, *Good*, τ_1, and τ_2 given
  - **Task:** find

Symbolic Framework: Regular Model Checking

Regular Model Checking for 2-player reachability games:

- Liveness:
  - *Start*, *Good*, τ_1, and τ_2 given
  - **Task:** find
    - FA *Inv* over-approximating reachable states

Symbolic Framework: Regular Model Checking

Regular Model Checking for 2-player reachability games:

- Liveness:
  - *Start*, *Good*, τ_1, and τ_2 given
  - **Task:** find
    - FA *Inv* over-approximating reachable states, and
    - transducer $P_<$ encoding **progress** for Process

Symbolic Framework: Regular Model Checking

Regular Model Checking for 2-player reachability games:

- Liveness:
  - *Start*, *Good*, τ_1, and τ_2 given
  - **Task:** find
    - FA *Inv* over-approximating reachable states, and
    - transducer $P_<$ encoding **progress** for Process

    } **Advice bits**

Symbolic Framework: Regular Model Checking

Regular Model Checking for 2-player reachability games:

- Liveness:
- *Start*, *Good*, τ_1, and τ_2 given
- **Advice bits**: local conditions on FA *Inv* and transducer $P_<$ over Σ

Symbolic Framework: Regular Model Checking

Regular Model Checking for 2-player reachability games:

- Liveness:
- *Start*, *Good*, τ_1, and τ_2 given
- **Advice bits**: local conditions on FA *Inv* and transducer $P_<$ over Σ
  1. $\mathit{Start} \subseteq \mathit{Inv}$
  2. $\tau_\cup(\mathit{Inv}) \subseteq \mathit{Inv}$

Symbolic Framework: Regular Model Checking

Regular Model Checking for 2-player reachability games:

- Liveness:
- *Start*, *Good*, τ_1, and τ_2 given
- **Advice bits**: local conditions on FA *Inv* and transducer $P_<$ over Σ
  1. $\mathit{Start} \subseteq \mathit{Inv}$
  2. $\tau_\cup(\mathit{Inv}) \subseteq \mathit{Inv}$
  3. $P_<$ is a strict preorder (i.e., irreflexive, transitive)

Symbolic Framework: Regular Model Checking

Regular Model Checking for 2-player reachability games:

- Liveness:
- *Start*, *Good*, τ_1, and τ_2 given
- **Advice bits**: local conditions on FA *Inv* and transducer $P_<$ over Σ
  1. $Start \subseteq Inv$
  2. $\tau_\cup(Inv) \subseteq Inv$
  3. $P_<$ is a strict preorder (i.e., irreflexive, transitive)
  4. For any evil transition from $Inv \setminus Good$ to s_e, there is an angelic transition from s_e that
     - goes to *Inv* and
     - progresses w.r.t. $P_<$

$$\forall x \in Inv \setminus Good, \quad \forall y \in \Sigma^* \setminus Good :$$
$$(x \rightarrow_{\tau_1} y) \Rightarrow (\exists z \in Inv : (y \rightarrow_{\tau_2} z \wedge z <_P x))$$

Finitary Fairness — [Alur & Henzinger'98]

k-Fairness

Finitary Fairness — [Alur & Henzinger'98]

k-Fairness

- *intuition*: binds the scope of $\Box$ and $\Diamond$ operators to k steps.

Finitary Fairness — [Alur & Henzinger'98]

k-Fairness

- *intuition*: binds the scope of $\Box$ and $\Diamond$ operators to k steps.
- **weak** (justice): $\Diamond\Box A \Rightarrow \Box\Diamond B$

Finitary Fairness — [Alur & Henzinger'98]

k-Fairness

- *intuition*: binds the scope of $\Box$ and $\Diamond$ operators to k steps.
- **weak** (justice): $\Diamond\Box A \Rightarrow \Box\Diamond B$

> No (sub-)path of length k satisfies $\Box(A \land \neg B)$.

Finitary Fairness — [Alur & Henzinger'98]

k-Fairness

- *intuition*: binds the scope of $\Box$ and $\Diamond$ operators to k steps.
- **weak** (justice): $\Diamond\Box A \Rightarrow \Box\Diamond B$

No (sub-)path of length k satisfies $\Box(A \wedge \neg B)$.

  - A cannot hold for k consecutive steps without B holding.

Finitary Fairness — [Alur & Henzinger'98]

k-Fairness

- *intuition*: binds the scope of $\Box$ and $\Diamond$ operators to k steps.
- **weak** (justice): $\Diamond\Box A \Rightarrow \Box\Diamond B$

No (sub-)path of length k satisfies $\Box(A \wedge \neg B)$.

  - A cannot hold for k consecutive steps without B holding.

- **strong** (compassion): $\Box\Diamond A \Rightarrow \Box\Diamond B$

Finitary Fairness — [Alur & Henzinger'98]

k-Fairness

- *intuition*: binds the scope of $\Box$ and $\Diamond$ operators to k steps.
- **weak** (justice): $\Diamond\Box A \Rightarrow \Box\Diamond B$

No (sub-)path of length k satisfies $\Box(A \wedge \neg B)$.

  - A cannot hold for k consecutive steps without B holding.
- **strong** (compassion): $\Box\Diamond A \Rightarrow \Box\Diamond B$

No path satisfies $\psi_k \wedge \Box\neg B$.

$$\psi_0 = \mathit{true}$$
$$\psi_i = \Diamond(A \wedge \psi_{i-1})$$

Finitary Fairness — [Alur & Henzinger'98]

k-Fairness

- *intuition*: binds the scope of $\Box$ and $\Diamond$ operators to k steps.
- **weak** (justice): $\Diamond\Box A \Rightarrow \Box\Diamond B$

No (sub-)path of length k satisfies $\Box(A \wedge \neg B)$.

  - A cannot hold for k consecutive steps without B holding.

- **strong** (compassion): $\Box\Diamond A \Rightarrow \Box\Diamond B$

No path satisfies $\psi_k \wedge \Box\neg B$.

$$\psi_0 = \mathit{true}$$
$$\psi_i = \Diamond(A \wedge \psi_{i-1})$$

  - A cannot hold k times without B holding at some point.

Finitary Fairness — [Alur & Henzinger'98]

k-Fairness

- *intuition*: binds the scope of $\Box$ and $\Diamond$ operators to k steps.
- **weak** (justice): $\Diamond\Box A \Rightarrow \Box\Diamond B$

No (sub-)path of length k satisfies $\Box(A \wedge \neg B)$.

  - A cannot hold for k consecutive steps without B holding.

- **strong** (compassion): $\Box\Diamond A \Rightarrow \Box\Diamond B$

No path satisfies $\psi_k \wedge \Box\neg B$.

$$\psi_0 = \mathit{true}$$
$$\psi_i = \Diamond(A \wedge \psi_{i-1})$$

  - A cannot hold k times without B holding at some point.

Finitary fairness: if k-fair for some k

Encoding Finitary Fairness into RMC

Encoding Finitary Fairness into RMC:

Encoding Finitary Fairness into RMC

Encoding Finitary Fairness into RMC:

- Fix some k

Encoding Finitary Fairness into RMC

Encoding Finitary Fairness into RMC:

- Fix some k
- Example for process selection (weak fairness)
  - every process is selected at least once in k steps

Encoding Finitary Fairness into RMC

Encoding Finitary Fairness into RMC:

- Fix some k
- Example for process selection (weak fairness)
  - every process is selected at least once in k steps
- Append a counter to encoding of every process, initialized to maximum
  - the maximum value is bounded

Encoding Finitary Fairness into RMC:

- Fix some k
- Example for process selection (weak fairness)
  - every process is selected at least once in k steps
- Append a counter to encoding of every process, initialized to maximum
  - the maximum value is bounded
- When a process is selected, reset its counter to max. value

Encoding Finitary Fairness into RMC

Encoding Finitary Fairness into RMC:

- Fix some k
- Example for process selection (weak fairness)
  - every process is selected at least once in k steps
- Append a counter to encoding of every process, initialized to maximum
  - the maximum value is bounded
- When a process is selected, reset its counter to max. value
- When a process is not selected, decrement its counter

Encoding Finitary Fairness into RMC

Encoding Finitary Fairness into RMC:

- Fix some k
- Example for process selection (weak fairness)
  - every process is selected at least once in k steps
- Append a counter to encoding of every process, initialized to maximum
  - the maximum value is bounded
- When a process is selected, reset its counter to max. value
- When a process is not selected, decrement its counter
- *Good* configurations are also those where some counter $= 0$

Encoding Finitary Fairness into RMC

Encoding Finitary Fairness into RMC:

- Fix some k
- Example for process selection (weak fairness)
  - every process is selected at least once in k steps
- Append a counter to encoding of every process, initialized to maximum
  - the maximum value is bounded
- When a process is selected, reset its counter to max. value
- When a process is not selected, decrement its counter
- *Good* configurations are also those where some counter $= 0$
- Generalized to arbitrary weak and strong fairness

Encoding Finitary Fairness into RMC

Example: Herman's protocol:

- w/o fairness: N | T | T | N

Encoding Finitary Fairness into RMC

Example: Herman's protocol:

- w/o fairness: N | T | T | N
- w/ fairness: N 1 1 0 | T 1 1 1 | T 1 1 0 | N 1 0 0

Encoding Finitary Fairness into RMC

Example: Herman's protocol:

- w/o fairness: N | T | T | N
- w/ fairness: N 1 1 0 | T 1 1 1 | T 1 1 0 | N 1 0 0
- scheduler picks a process

N 1 1 0 | T 1 1 1 | T 1 1 0 | N 1 0 0

Encoding Finitary Fairness into RMC

Example: Herman's protocol:

- w/o fairness: N | T | T | N
- w/ fairness: N 1 1 0 | T 1 1 1 | T 1 1 0 | N 1 0 0
- scheduler picks a process

  N 1 1 0 | T 1 1 1 | **T 1 1 0** | N 1 0 0
- process player decrements/resets counters

  N 1 0 0 | T 1 1 0 | **T 1 1 1** | N 0 0 0

Encoding Finitary Fairness into RMC

Theorem

Let S be a regular representation of an MDP with finitary fairness constraints C. The presented transformation yields a regular representation of an MDP S_F (without fairness constraints) such that (if C are realizable)

$$\Pr(\mathit{Start} \models \Diamond \mathit{Good}) = 1 \qquad \textit{iff} \qquad \Pr(\mathit{Start}_F \models \Diamond \mathit{Good}_F) = 1$$

Case Studies: Population Protocols

Moran process

Moran process

- a model of genetic drift

Case Studies: Population Protocols

Moran process

- a model of genetic drift
- linear array

Case Studies: Population Protocols

Moran process

- a model of genetic drift
- linear array
- alleles (A) or (B)

Case Studies: Population Protocols

Moran process

- a model of genetic drift
- linear array
- alleles (A) or (B)
- rules:

Case Studies: Population Protocols

Moran process

- a model of genetic drift
- linear array
- alleles Ⓐ or Ⓑ
- rules:
  - $\ldots$ Ⓐ $\ldots \rightsquigarrow \ldots$ Ⓐ $\ldots$

Moran process

- a model of genetic drift
- linear array
- alleles (A) or (B)
- rules:
  - $\ldots$ (A) $\ldots \rightsquigarrow \ldots$ (A) $\ldots$
  - $\ldots$ (A) (B) $\ldots \rightsquigarrow \ldots$ (A) (A) $\ldots$

Case Studies: Population Protocols

Moran process

- a model of genetic drift
- linear array
- alleles Ⓐ or Ⓑ
- rules:
  - . . . Ⓐ . . . ⇝ . . . Ⓐ . . .
  - . . . Ⓐ Ⓑ . . . ⇝ . . . Ⓐ Ⓐ . . .
  - . . . Ⓑ Ⓐ . . . ⇝ . . . Ⓐ Ⓐ . . .

Moran process

- a model of genetic drift
- linear array
- alleles (A) or (B)
- rules:
  - $\ldots$ (A) $\ldots \rightsquigarrow \ldots$ (A) $\ldots$
  - $\ldots$ (A) (B) $\ldots \rightsquigarrow \ldots$ (A) (A) $\ldots$
  - $\ldots$ (B) (A) $\ldots \rightsquigarrow \ldots$ (A) (A) $\ldots$
  - (similar for (B))

Case Studies: Population Protocols

Moran process

- a model of genetic drift
- linear array
- alleles (A) or (B)
- rules:
  - ... [(A)] ... ⇝ ... [(A)] ...
  - ... [(A)] (B) ... ⇝ ... [(A)] (A) ...
  - ... (B) [(A)] ... ⇝ ... (A) [(A)] ...
  - (similar for [(B)])
- goal: $(A)^*$ or $(B)^*$

Case Studies: Population Protocols

Moran process

- a model of genetic drift
- linear array
- alleles Ⓐ or Ⓑ
- rules:
  - $\ldots$ Ⓐ $\ldots \rightsquigarrow \ldots$ Ⓐ $\ldots$
  - $\ldots$ Ⓐ Ⓑ $\ldots \rightsquigarrow \ldots$ Ⓐ Ⓐ $\ldots$
  - $\ldots$ Ⓑ Ⓐ $\ldots \rightsquigarrow \ldots$ Ⓐ Ⓐ $\ldots$
  - (similar for Ⓑ)
- goal: Ⓐ* or Ⓑ*
- **Cell cycle switch** — similar, but has an intermediate state

Case Studies: Population Protocols

Clustering

Case Studies: Population Protocols

Clustering

- linear array

Clustering

- linear array
- alleles A or B

Case Studies: Population Protocols

Clustering

- linear array
- alleles (A) or (B)
- rules:

Case Studies: Population Protocols

Clustering

- linear array
- alleles Ⓐ or Ⓑ
- rules:
  - $\ldots$ Ⓐ Ⓑ $\ldots \rightsquigarrow \ldots$ Ⓑ Ⓐ $\ldots$

Clustering

- linear array
- alleles (A) or (B)
- rules:
  - ... (A) (B) ... ⇝ ... (B) (A) ...
  - ... (B) (A) ... ⇝ ... (A) (B) ...

Clustering

- linear array
- alleles (A) or (B)
- rules:
  - … (A) (B) … $\leadsto$ … (B) (A) …
  - … (B) (A) … $\leadsto$ … (A) (B) …
  - (similar for (B))

Clustering

- linear array
- alleles (A) or (B)
- rules:
  - … (A)(B) … ⇝ … (B)(A) …
  - … (B)(A) … ⇝ … (A)(B) …
  - (similar for (B))
- goal: $(A)^*(B)^*$ or $(B)^*(A)^*$

Coin game

- a population of **agents**
- every agent has one currency: Dollars or Euros
- in each step, an agent either:

Case Studies: Population Protocols

Coin game

- a population of **agents**
- every agent has one currency: Dollars or Euros
- in each step, an agent either:
  - keeps it currency or
  - randomly selects k neighbours and changes currency to the majority

Coin game

- a population of **agents**
- every agent has one currency: Dollars or Euros
- in each step, an agent either:
  - keeps it currency or
  - randomly selects k neighbours and changes currency to the majority
- goal: D^* or E^*

Evaluation

- Encoding implemented in FAIRYTAIL

Evaluation

- Encoding implemented in FAIRYTAIL
- **Input**:
  - FAs for *Start*, *Good*
  - transducers for τ_1, and τ_2

Evaluation

- Encoding implemented in FAIRYTAIL
- **Input**:
  - FAs for *Start*, *Good*
  - transducers for τ_1, and τ_2
- **Output**:
  - FAs for Start^F, Good^F
  - transducers for τ_1^F, and τ_2^F

Evaluation

- Encoding implemented in FAIRYTAIL
- **Input**:
  - FAs for *Start*, *Good*
  - transducers for τ_1, and τ_2
- **Output**:
  - FAs for Start^F, Good^F
  - transducers for τ_1^F, and τ_2^F
- SLRP [Lin & Rümmer, CAV'16] used to find **advice bits**

Evaluation

- Encoding implemented in FAIRYTAIL
- **Input**:
  - FAs for *Start*, *Good*
  - transducers for τ_1, and τ_2
- **Output**:
  - FAs for $Start^F$, $Good^F$
  - transducers for τ_1^F, and τ_2^F
- SLRP [Lin & Rümmer, CAV'16] used to find **advice bits**
  - SYNTHESISE: use a SAT solver (Sat4j) to obtain a candidate

Evaluation

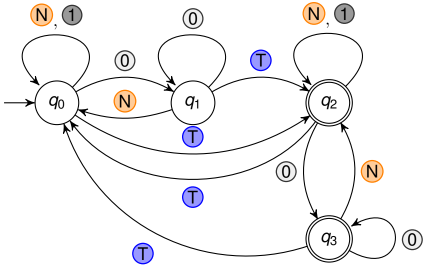
- Encoding implemented in FAIRYTAIL
- **Input**:
  - FAs for *Start*, *Good*
  - transducers for τ_1, and τ_2
- **Output**:
  - FAs for $Start^F$, $Good^F$
  - transducers for τ_1^F, and τ_2^F
- SLRP [Lin & Rümmer, CAV'16] used to find **advice bits**
  - SYNTHESISE: use a SAT solver (Sat4j) to obtain a candidate
  - VERIFY: check the candidate is OK/refine SAT formula

Evaluation

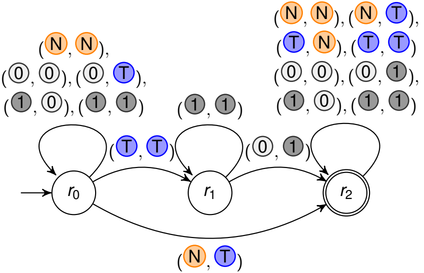
Table: Results of experiments (timeout = 10 hours).

Case study	Time
Herman's protocol (merge, line)	3.64 s
Herman's protocol (annih., line)	4.33 s
Herman's protocol (merge, ring)	4.31 s
Herman's protocol (annih., ring)	4.61 s
Moran process (2 types, line)	2 m 48 s
Moran process (3 types, line)	56 m 14 s
Cell cycle switch (1 types, line)	43.94 s
Cell cycle switch (2 types, line)	9 h 46 m
Clustering (2 types, line)	10 m 30 s
Clustering (3 types, line)	T/O
Coin game ($k = 3$, clique)	1 m 0 s

Solution to Herman's protocol (merge, ring)



Inv



$P_<$

Conclusion

- A nice **symbolic framework** for reasoning about parameterized probabilistic concurrent systems.

Conclusion

- A nice **symbolic framework** for reasoning about parameterized probabilistic concurrent systems.
- In this talk extended with **finitary fairness**.
  - a natural notion of fairness in such systems

Conclusion

- A nice **symbolic framework** for reasoning about parameterized probabilistic concurrent systems.
- In this talk extended with **finitary fairness**.
  - a natural notion of fairness in such systems

Future work:

- many optimizations possible

Conclusion

- A nice **symbolic framework** for reasoning about parameterized probabilistic concurrent systems.
- In this talk extended with **finitary fairness**.
  - a natural notion of fairness in such systems

Future work:

- many optimizations possible
- more general systems (e.g., grid topology)

Conclusion

- A nice **symbolic framework** for reasoning about parameterized probabilistic concurrent systems.
- In this talk extended with **finitary fairness**.
  - a natural notion of fairness in such systems

Future work:

- many optimizations possible
- more general systems (e.g., grid topology)
- more general fairness