Verifying qualitative liveness properties of replicated systems with stochastic scheduling

Martin Helfrich
Joint work with Michael Blondin, Javier Esparza, Antonín Kučera, and Philipp J. Meyer
**Replicated systems**: formal model to describe “swarms” of identical finite-state agents.
Replicated Systems

- **Replicated systems**: formal model to describe “swarms” of identical finite-state agents

- set of states and set of *multiset rewriting transitions* $M \rightarrow M'$ where $|M| = |M'|$ (conservative VASs)
Replicated Systems

- **Replicated systems**: formal model to describe “swarms” of identical finite-state agents
- set of states and set of **multiset rewriting transitions**\( M \rightarrow M' \) where \(|M| = |M'|\) (conservative VASs)
- unknown nr. of agents & infinite sets of initial configurations
Replicated Systems

- **Replicated systems**: formal model to describe “swarms” of identical finite-state agents
  - set of states and set of *multiset rewriting transitions* $M \rightarrow M'$ where $|M| = |M'|$ (conservative VASs)
  - unknown nr. of agents & infinite sets of initial configurations
- **Stochastic scheduling**: agents to move next are chosen stochastically (every enabled transition has nonzero probability).
Replicated Systems

- **Replicated systems**: formal model to describe "swarms" of identical finite-state agents

- set of states and set of multiset rewriting transitions $M \rightarrow M'$ where $|M| = |M'|$ (conservative VASs)

- unknown nr. of agents & infinite sets of initial configurations

- **Stochastic scheduling**: agents to move next are chosen stochastically (every enabled transition has nonzero probability).

- Can model (abstractions of) multithreaded programs, population protocols and other distributed consensus algorithms
Qualitative model checking:

- LTL with Presburger formulas as atomic propositions encoding sets of configurations
- Problem: decide if the runs satisfying the property have probability 1
- Unsurprisingly: not even semi-decidable
Qualitative model checking:

- LTL with **Presburger formulas** as atomic propositions encoding sets of configurations
- Problem: decide if the runs satisfying the property have probability 1
- Unsurprisingly: not even semi-decidable

**Limit to fragment: stable termination**

$$\text{Pre} \rightarrow F \left( \bigvee_{i=1}^{k} G \text{ Post}_i \right)$$

“Eventually some postcondition $i$ holds forever.”
Qualitative model checking:

- LTL with Presburger formulas as atomic propositions encoding sets of configurations
- Problem: decide if the runs satisfying the property have probability 1
- Unsurprisingly: not even semi-decidable

Limit to fragment: stable termination

\[ \text{Pre} \rightarrow F \left( \bigvee_{i=1}^k \text{G Post}_i \right) \]

“Eventually some postcondition \( i \) holds forever.”

Typical properties: Leader election, consensus
At least as many blue birds as red birds?
Example: majority voting protocol

At least as many blue birds as red birds?

Protocol:

- 4 states: blue/red, large/small
- Two large birds of different colors become small and blue
- Large birds convert small birds to their color
- Small blue birds convert small red birds
Example: majority voting protocol

At least as many **blue birds** as **red birds**?

**Protocol:**

- 4 states: blue/red, large/small
- Two large birds of different colors become small and blue
- Large birds convert small birds to their color
- Small blue birds convert small red birds
Example: majority voting protocol

At least as many blue birds as red birds?

Protocol:

- 4 states: blue/red, large/small
- Two large birds of different colors become small and blue
- Large birds convert small birds to their color
- Small blue birds convert small red birds
Example: majority voting protocol

At least as many blue birds as red birds?

Protocol:

- 4 states: blue/red, large/small
- Two large birds of different colors become small and blue
- Large birds convert small birds to their color
- Small blue birds convert small red birds
At least as many blue birds as red birds?

Protocol:
- 4 states: blue/red, large/small
- Two large birds of different colors become small and blue
- Large birds convert small birds to their color
- Small blue birds convert small red birds
Example: majority voting protocol

At least as many **blue birds** as **red birds**?

**Protocol:**

- 4 states: blue/red, large/small
- **Two large birds of different colors become small and blue**
- **Large birds convert small birds to their color**
- **Small blue birds convert small red birds**
Example: majority voting protocol

At least as many blue birds as red birds?

Protocol:

- 4 states: blue/red, large/small
- Two large birds of different colors become small and blue
- Large birds convert small birds to their color
- Small blue birds convert small red birds
At least as many blue birds as red birds?

Protocol:

- 4 states: blue/red, large/small
- Two large birds of different colors become small and blue
- Large birds convert small birds to their color
- Small blue birds convert small red birds
At least as many blue birds as red birds?

Protocol:
- 4 states: blue/red, large/small
- Two large birds of different colors become small and blue
- Large birds convert small birds to their color
- Small blue birds convert small red birds
At least as many blue birds as red birds?

Protocol:

- 4 states: blue/red, large/small
- Two large birds of different colors become small and blue
- Large birds convert small birds to their color
- Small blue birds convert small red birds
Example: majority voting protocol

At least as many blue birds as red birds?

Protocol:

- 4 states: blue/red, large/small
- Two large birds of different colors become small and blue
- Large birds convert small birds to their color
- Small blue birds convert small red birds

Stable termination properties:

\[
\begin{align*}
\left( \begin{array}{c}
\geq \\
\end{array} \right) & \implies FG \left( \begin{array}{c}
+ \\
= 0 \\
\end{array} \right) \\
\left( \begin{array}{c}
< \\
\end{array} \right) & \implies FG \left( \begin{array}{c}
+ \\
= 0 \\
\end{array} \right)
\end{align*}
\]

“Birds converge to color of majority.”
Replicated Systems: formal model

- **States:** finite set $Q$
- **Transitions:**
  $$ T \subseteq \bigcup_{k \geq 2} Q^{(k)} \times Q^{(k)} $$
Replicated Systems: formal model

- **States:**
  - finite set $Q$

- **Transitions:**
  - $T \subseteq \bigcup_{k \geq 2} Q^{(k)} \times Q^{(k)}$
Replicated Systems: formal model

- **Configurations**: \( Q \rightarrow \mathbb{N} \)

- Transitions induce step relation \( C \rightarrow C' \) between configurations
• Configurations: $Q \rightarrow \mathbb{N}$

• Transitions induce step relation $C \rightarrow C'$ between configurations
Our Results

We answer two questions:

1. **Theory:** How to verify stable termination?
   → sound & complete procedure producing structural proofs
Our Results

We answer two questions:

1. **Theory:** How to verify stable termination?
   → sound & complete procedure producing structural proofs

2. **Practice:** How to automatically verify stable termination?
   → semi-decision algorithm
Most stable termination proofs are structured in **stages**: milestones trapping the system in increasingly smaller sets of configurations, until it gets trapped in some Post; 

\[\downarrow\]

**Stage Graphs**
Preliminaries: inductive set

A (possibly infinite) set of configurations $S$ is inductive iff it closed under reachability:
A (possibly infinite) set of configurations $S$ is \textit{inductive} iff it closed under reachability:
A (possibly infinite) set of configurations $S$ is **inductive** iff it closed under reachability:

\[ S_2: \text{ inductive} \]
A (possibly infinite) set of configurations $S$ is **inductive** iff it closed under reachability:

$S_3$: inductive
A (possibly infinite) set of configurations $S$ is **inductive** iff it closed under reachability:

$S_4$: **not** inductive
Let $S, S'$ be sets of configurations

- $S \leadsto S'$: runs starting at $S$ visit $S'$ with probability 1
Preliminaries: certificate

Let $S, S'$ be sets of configurations

- $S \rightsquigarrow S'$: runs starting at $S$ visit $S'$ with probability 1
Let $S, S'$ be sets of configurations

- $S \rightsquigarrow S'$: runs starting at $S$ visit $S'$ with probability 1
Preliminaries: certificate

Let $S, S'$ be sets of configurations

- $S \leadsto S'$: runs starting at $S$ visit $S'$ with probability 1
- Certificate for $S \leadsto S'$: mapping $f : S \rightarrow \mathbb{N}$ such that for every $C \in S \setminus S'$ there exists $C \rightarrow C'$ such that $f(C) > f(C')$. 

![Diagram showing $S_2 \leadsto S_3$]
Preliminaries: certificate

Let $S, S'$ be sets of configurations

- $S \leadsto S'$: runs starting at $S$ visit $S'$ with probability 1
- **Certificate for $S \leadsto S'$**: mapping $f : S \rightarrow \mathbb{N}$ such that for every $C \in S \setminus S'$ there exists $C \xrightarrow{*} C'$ such that $f(C) > f(C')$. 

![Diagram](image_url)
A **stage graph** for a given property $\text{Pre} \rightarrow F\left(\bigvee_{i=1}^{k} G \text{Post}_i\right)$ is a finite DAG satisfying:

1. The nodes of the DAG, called stages, are inductive sets of configurations.
2. Every configuration of $\text{Pre}$ belongs to some stage.
3. For every non-terminal stage $S$ with children $S_1, \ldots, S_n$, there is a certificate for $S \leadsto S_1 \cup \cdots \cup S_n$.
4. Every terminal stage is contained on some $\text{Post}_i$. 


Stage Graphs

A stage graph for a given property $\text{Pre} \rightarrow F \left( \bigvee_{i=1}^{k} G \text{Post}_i \right)$ is a finite DAG satisfying:

1. The nodes of the DAG, called stages, are inductive sets of configurations.
A stage graph for a given property $\text{Pre} \rightarrow F \left( \bigvee_{i=1}^{k} G \text{Post}_i \right)$ is a finite DAG satisfying:

1. The nodes of the DAG, called stages, are inductive sets of configurations
2. Every configuration of $\text{Pre}$ belongs to some stage
Stage Graphs

A stage graph for a given property $\text{Pre} \rightarrow F \left( \bigvee_{i=1}^{k} G \text{Post}_i \right)$ is a finite DAG satisfying:

1. The nodes of the DAG, called stages, are inductive sets of configurations
2. Every configuration of $\text{Pre}$ belongs to some stage
3. For every non-terminal stage $S$ with children $S_1, \ldots, S_n$ there is a certificate for $S \leadsto S_1 \cup \cdots \cup S_n$
A **stage graph** for a given property $\text{Pre} \rightarrow F \left( \bigvee_{i=1}^{k} G \text{Post}_i \right)$ is a finite DAG satisfying:

1. The nodes of the DAG, called **stages**, are inductive sets of configurations
2. Every configuration of $\text{Pre}$ belongs to some stage
3. For every non-terminal stage $S$ with children $S_1, \ldots, S_n$ there is a certificate for $S \rightsquigarrow S_1 \cup \cdots \cup S_n$
4. Every terminal stage is contained on some $\text{Post}_i$
Stage Graphs

\[ G \]

\[ S_1 \]

\[ S_2 \]

\[ S_3 \]
Stages $S$ of $G$ are inductive sets
Stage Graphs

\[ \mathcal{G} \]

\[ S_1 \]

\[ S_2 \]

\[ S_3 \]

\[ \text{Pre} \subseteq \bigcup_{S \in \mathcal{G}} S \]
Certificates for non-terminal stages $S \leadsto S_1 \cup \ldots \cup S_k$
Stage Graphs

Certificates for non-terminal stages \( S \sim S_1 \cup \ldots \cup S_k \)

children of \( S \)
Stage Graphs

$G$

$S_1$

$S_2$

$S_3$

Terminal stages $S \subseteq Post$
Stage Graph Example: majority voting protocol

$t_1$:  
$t_2$:  
$t_3$:  
$t_4$:  

Cert: $S_1 \land S_2 \land S_3$
Stage Graph Example: majority voting protocol

Stage graph for property \((\geq) \implies \text{FG} (\geq + < 0)\)

\[
\begin{align*}
\text{Stage Graph:} & \quad t_1: \quad \text{blue, red} \quad \implies \quad \text{red, blue} \quad t_3: \quad \text{red, blue} \quad \implies \quad \text{red, red} \\
& \quad t_2: \quad \text{blue, red} \quad \implies \quad \text{blue, blue} \quad t_4: \quad \text{blue, red} \quad \implies \quad \text{blue, blue}
\end{align*}
\]
Stage Graph Example: majority voting protocol

Stage graph for property $(\text{blue} \geq \text{red}) \iff \text{FG} (\text{red} + \text{red} = 0)$

$S_1: \text{Reach} (\text{blue} \geq \text{red})$

Cert: $\text{blue} \geq \text{red}$

$S_2: \text{Reach} (\text{blue} \geq \text{red}) \land \text{red} = 0$

Cert: $\text{blue} \geq \text{red}$

$S_3: \text{Reach} (\text{blue} \geq \text{red}) \land \text{red} + \text{red} = 0$
### Soundness

If there is a stage graph for a property, then it holds.

### Completeness

If a property holds, then there is a stage graph proving it.
<table>
<thead>
<tr>
<th>Theory</th>
<th>CAV 2020</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Soundness</strong></td>
<td></td>
</tr>
<tr>
<td>If there is a stage graph for a property, then it holds.</td>
<td></td>
</tr>
<tr>
<td><strong>Completeness</strong></td>
<td></td>
</tr>
<tr>
<td>If a property holds, then there is a stage graph proving it.</td>
<td></td>
</tr>
<tr>
<td><strong>Soundness</strong></td>
<td>CAV 2020</td>
</tr>
<tr>
<td>---------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>If there is a stage graph for a property, then it holds.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Completeness</strong></th>
<th>CAV 2020</th>
</tr>
</thead>
<tbody>
<tr>
<td>If a property holds, then there is a stage graph proving it.</td>
<td></td>
</tr>
</tbody>
</table>

What about decidability?
→ unknown (stages can be arbitrarily complicated!)
Stage Graphs: theory

<table>
<thead>
<tr>
<th>Soundness</th>
<th>CAV 2020</th>
</tr>
</thead>
<tbody>
<tr>
<td>If there is a stage graph for a property, then it holds.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Completeness</th>
<th>CAV 2020</th>
</tr>
</thead>
<tbody>
<tr>
<td>If a property holds, then there is a stage graph proving it.</td>
<td></td>
</tr>
</tbody>
</table>

A **Presburger stage graph** is a stage graph where

- nodes are **Presburger** sets,

and

\[ C \in S \iff \phi(C) \]
Stage Graphs: theory

<table>
<thead>
<tr>
<th>Soundness</th>
<th>CAV 2020</th>
</tr>
</thead>
<tbody>
<tr>
<td>If there is a stage graph for a property, then it holds.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Completeness</th>
<th>CAV 2020</th>
</tr>
</thead>
<tbody>
<tr>
<td>If a property holds, then there is a stage graph proving it.</td>
<td></td>
</tr>
</tbody>
</table>

A Presburger stage graph is a stage graph where

- nodes are Presburger sets, and
- certificates are Presburger certificates.

\[
C \in \mathcal{S} \iff \phi(C)
\]

\[
f(C) = a \iff \phi(C, a)
\]
Stage Graphs: theory

<table>
<thead>
<tr>
<th>Soundness</th>
<th>CAV 2020</th>
</tr>
</thead>
<tbody>
<tr>
<td>If there is a <strong>Presburger</strong> stage graph for a property, then it holds.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Completeness</th>
<th>CAV 2020</th>
</tr>
</thead>
<tbody>
<tr>
<td>If a property holds, then there is a <strong>Presburger</strong> stage graph proving it.</td>
<td></td>
</tr>
</tbody>
</table>

A **Presburger stage graph** is a stage graph where

- nodes are **Presburger** sets,
- certificates are **Presburger** certificates.

\[
C \in \mathcal{S} \iff \phi(C)
\]

\[
f(C) = a \iff \phi(C, a)
\]
A Presburger stage graph is a stage graph where

- nodes are Presburger sets, and
- certificates are Presburger certificates.

\[
C \in S \iff \phi(C)
\]

\[
f(C) = a \iff \phi(C, a)
\]
## Stage Graphs: theory

<table>
<thead>
<tr>
<th>Soundness</th>
<th>CAV 2020</th>
</tr>
</thead>
<tbody>
<tr>
<td>If there is a Presburger stage graph for a property, then it holds.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Completeness</th>
<th>CAV 2020</th>
</tr>
</thead>
<tbody>
<tr>
<td>If a property holds, then there is a Presburger stage graph proving it.</td>
<td></td>
</tr>
</tbody>
</table>

Presburger stage graph can be **independently checked**!

→ everything reduces to checking Presburger formulas
<table>
<thead>
<tr>
<th><strong>Soundness</strong></th>
<th>CAV 2020</th>
</tr>
</thead>
<tbody>
<tr>
<td>If there is a Presburger stage graph for a property, then it holds.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Completeness</strong></th>
<th>CAV 2020</th>
</tr>
</thead>
<tbody>
<tr>
<td>If a property holds, then there is a Presburger stage graph proving it.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Decidability</strong></th>
<th>CAV 2020</th>
</tr>
</thead>
<tbody>
<tr>
<td>It is decidable if a system satisfies a given stable termination property.</td>
<td></td>
</tr>
</tbody>
</table>
**Stage Graphs: theory**

<table>
<thead>
<tr>
<th>Soundness</th>
<th>CAV 2020</th>
</tr>
</thead>
<tbody>
<tr>
<td>If there is a Presburger stage graph for a property, then it holds.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Completeness</th>
<th>CAV 2020</th>
</tr>
</thead>
<tbody>
<tr>
<td>If a property holds, then there is a Presburger stage graph proving it.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Decidability</th>
<th>CAV 2020</th>
</tr>
</thead>
<tbody>
<tr>
<td>It is decidable if a system satisfies a given stable termination property.</td>
<td></td>
</tr>
</tbody>
</table>

**(Alternative) proof.**

Two semi-decision algorithms:

- **For non-correctness:** enumerate all configurations and check property (finite-state model checking)
- **For correctness:** enumerate all Presburger stage graphs and check if they prove the property
### Soundness  
If there is a Presburger stage graph for a property, then it holds.  

### Completeness  
If a property holds, then there is a Presburger stage graph proving it.  

### Decidability  
It is decidable if a system satisfies a given stable termination property.  

**Problem**: stage graphs might be huge (non-elementary)  
→ How can stage graphs help with automatic verification?
Ideas:

• Most systems have small stage graphs
Stage Graphs: practice

Idea:\n• Most systems have small stage graphs
• Most systems "make progress" by "killing" transitions

Definition:
A transition is **dead** if it can never be enabled again.
Stage Graphs: practice

Ideas:
- Most systems have small stage graphs
- Most systems "make progress" by "killing" transitions
  → search for stages with more and more dead transitions

Definition
A transition is **dead** if it can never be enabled again.
Ideas:
- Most systems have small stage graphs
- Most systems "make progress" by "killing" transitions
  → search for stages with more and more dead transitions

Definition
A transition is **dead** if it can never be enabled again.

Algorithm:
SMT based semi-algorithm to automatically **construct** Presburger stage graphs
Stage Graphs: practice

Ideas:
- Most systems have small stage graphs
- Most systems "make progress" by "killing" transitions → search for stages with more and more dead transitions

Definition
A transition is dead if it can never be enabled again.

Algorithm:
SMT based semi-algorithm to automatically construct Presburger stage graphs
- reachability is TOWER-hard & not Presburger → overapproximate
Stage Graphs: practice

**Ideas:**
- Most systems have small stage graphs
- Most systems "make progress" by "killing" transitions → search for stages with more and more dead transitions

**Definition**
A transition is **dead** if it can never be enabled again.

**Algorithm:**
SMT based semi-algorithm to automatically **construct** Presburger stage graphs
- reachability is TOWER-hard & not Presburger → **overapproximate**
- use heuristics to find eventually dead transitions → find **linear ranking functions**
Automatically verifying population protocols:

| Population Protocol          | Predicate              | $|Q|$ | $|T|$ | Time   |
|------------------------------|------------------------|------|------|--------|
| Broadcast [31,22]            | $x_1 \lor \ldots \lor x_n$ | 2    | 1    | < 1s   |
| Majority (Example 1)[22]     | $x \geq y$             | 4    | 4    | < 1s   |
| Majority [23, Ex. 3]         | $x \geq y$             | 5    | 6    | < 1s   |
| Majority [5] (m=21,d=20)     | $x \geq y$             | 62   | 1953 | 3301s  |
| Flock-of-birds [28,22]       | $x \geq 80$            | 81   | 3240 | 1217s  |
| Flock-of-birds [20, Sect. 3] | $x \geq 120$           | 9    | 21   | 2551s  |
| F.o.B. [31,22, threshold-n]  | $x \geq 20$            | 21   | 39   | 18s    |
| Threshold [8][22]            | $\sum_i \alpha_i x_i \geq 8$ | 76   | 2148 | 1089s  |
| Threshold [20] (“succinct”)  | $\sum_i \alpha_i x_i \geq 511$ | 25   | 91   | 2659s  |
| Remainder [22]               | $\sum_i \alpha_i x_i \equiv_{20} 1$ | 22   | 230  | 1646s  |

\(^1\)Intel Xeon CPU E5-2630 v4 @ 2.20GHz and 8GB of RAM
## Experimental Results

Automatically verifying leader election algorithms:

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Processes</th>
<th></th>
<th>Q</th>
<th></th>
<th>T</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Israeli-Jalfon [44]</td>
<td>20</td>
<td>40</td>
<td>80</td>
<td>7s</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Israeli-Jalfon [44]</td>
<td>60</td>
<td>120</td>
<td>240</td>
<td>1493s</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Israeli-Jalfon [44]</td>
<td>70</td>
<td>140</td>
<td>280</td>
<td>3295s</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Herman [42]</td>
<td>21</td>
<td>42</td>
<td>42</td>
<td>9s</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Herman [42]</td>
<td>51</td>
<td>102</td>
<td>102</td>
<td>300s</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Herman [42]</td>
<td>81</td>
<td>162</td>
<td>162</td>
<td>2800s</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\(^1\)Intel Xeon CPU E5-2630 v4 @ 2.20GHz and 8GB of RAM
THANK YOU!