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

Technical University of Munich

European Research Council
Established by the European Commission

Replicated Systems

- **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 $M \to M'$ 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 $M \to M'$ 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 $M \to M'$ 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 $M \to M'$ 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

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

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

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

"Eventually some postcondition i holds forever."

Qualitative Model Checking

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

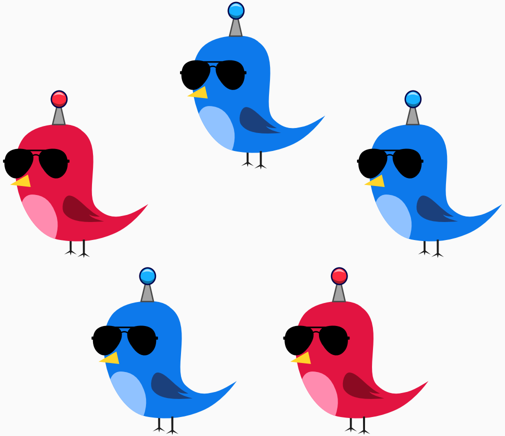
$$\mathrm{Pre} \rightarrow \mathbf{F}\left(\bigvee_{i=1}^{k} \mathbf{G}\, \mathrm{Post}_i\right)$$

"Eventually some postcondition i holds forever."

Typical properties: Leader election, consensus

Example: majority voting protocol

At least as many blue birds as red birds?

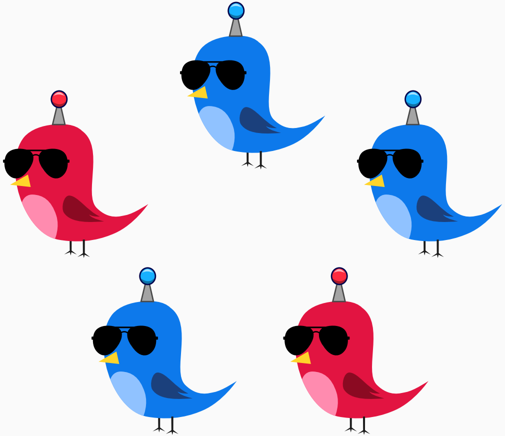


Example: majority voting protocol

At least as many blue birds as red birds?

Protocol:

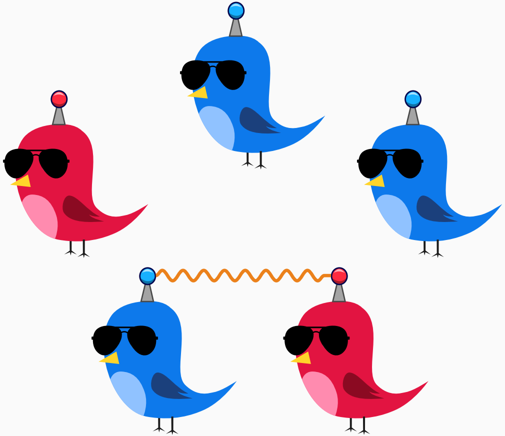
- 4 states: blue/red, large/small
- Two large birds of different colors become small and blue
- Large birds convert small birds to their color
- Small blue birds convert small red birds



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

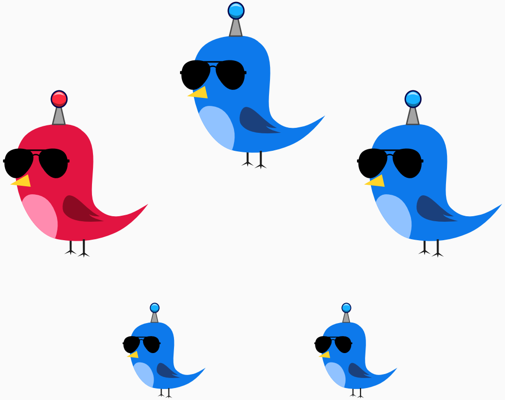


Example: majority voting protocol

At least as many blue birds as red birds?

Protocol:

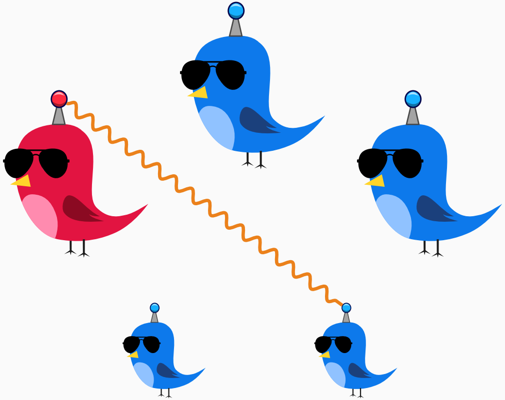
- 4 states: blue/red, large/small
- Two large birds of different colors become small and blue
- Large birds convert small birds to their color
- Small blue birds convert small red birds



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

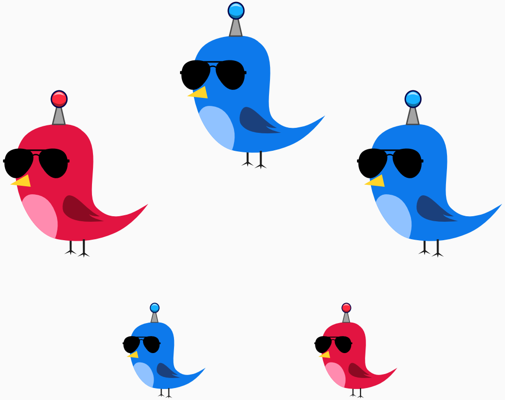


Example: majority voting protocol

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

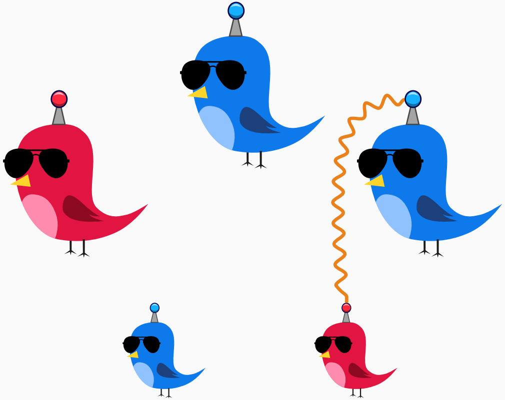


Example: majority voting protocol

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

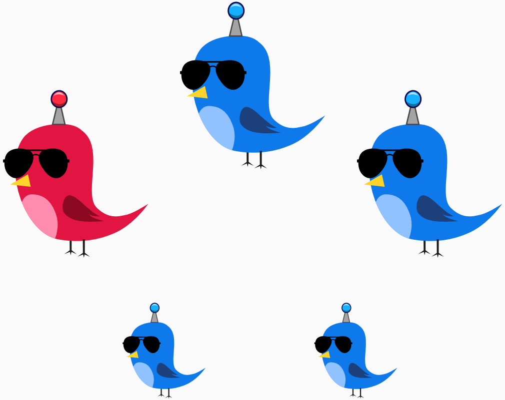


Example: majority voting protocol

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

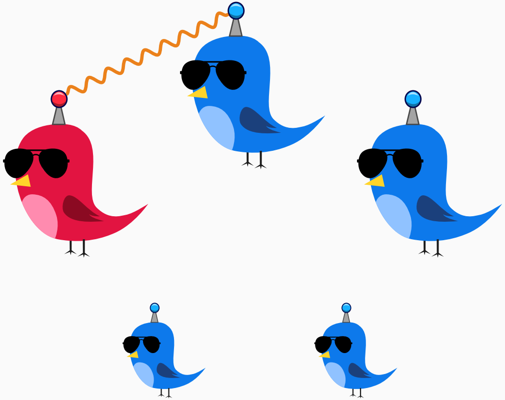


Example: majority voting protocol

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

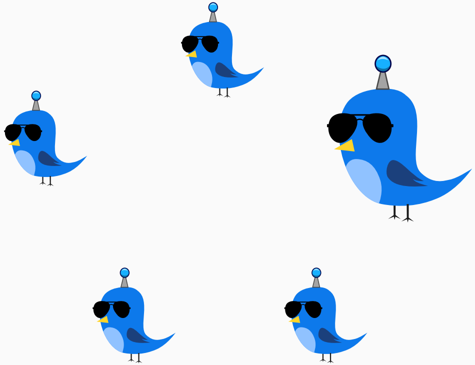


Example: majority voting protocol

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

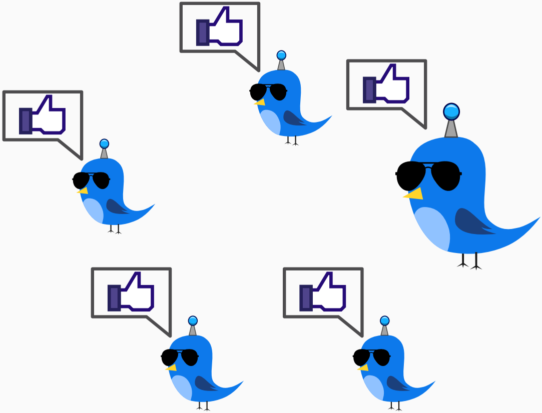


Example: majority voting protocol

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



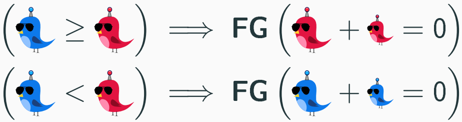
Example: majority voting protocol

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

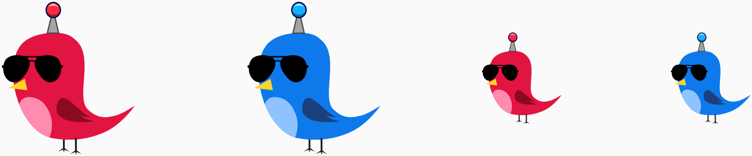
Stable termination properties:



"Birds converge to color of majority."

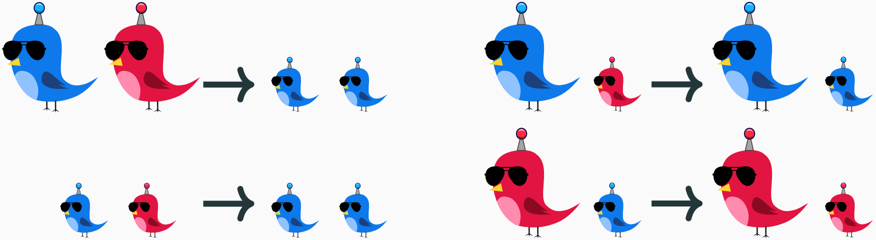
Replicated Systems: formal model

- States: finite set Q
- Transitions: $T \subseteq \bigcup_{k \geq 2} Q^{\langle k \rangle} \times Q^{\langle k \rangle}$



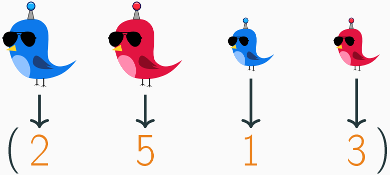
Replicated Systems: formal model

- States: finite set Q
- Transitions: $T \subseteq \bigcup_{k \geq 2} Q^{\langle k \rangle} \times Q^{\langle k \rangle}$



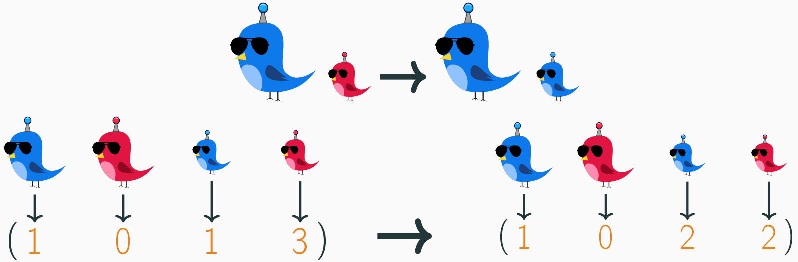
Replicated Systems: formal model

- Configurations: $Q \to \mathbb{N}$
- Transitions induce step relation $C \to C'$ between configurations



Replicated Systems: formal model

- Configurations: $Q \to \mathbb{N}$
- Transitions induce step relation $C \to C'$ between configurations



Our Results

We answer two questions:

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

Our Results

We answer two questions:

1. **Theory:** How to verify stable termination?
   $\rightarrow$ sound & complete procedure producing structural proofs
2. **Practice:** How to automatically verify stable termination?
   $\rightarrow$ semi-decision algorithm

Idea

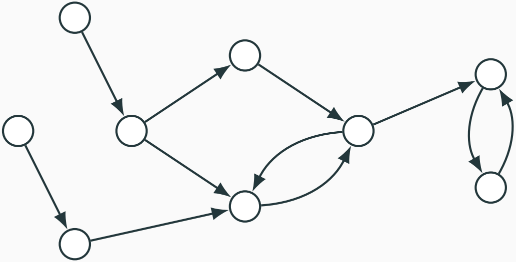
- Most stable termination proofs are structured in stages: milestones trapping the system in increasingly smaller sets of configurations, until it gets trapped in some Post_i

$$\Downarrow$$

Stage Graphs

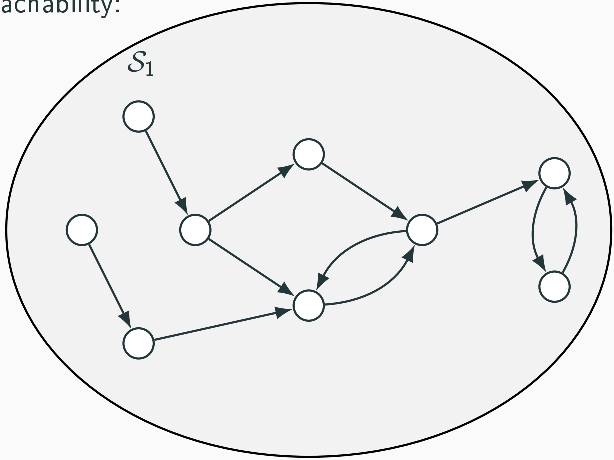
Preliminaries: inductive set

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



Preliminaries: inductive set

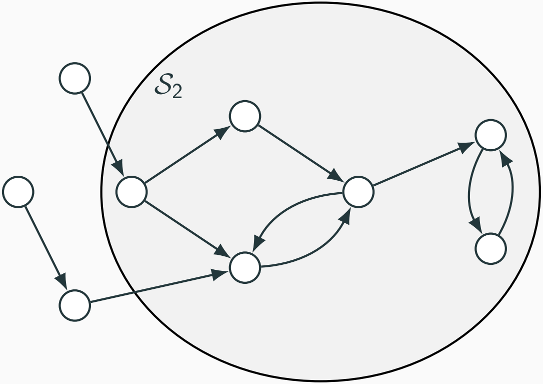
A (possibly infinite) set of configurations $\mathcal{S}$ is inductive iff it closed under reachability:



$\mathcal{S}_1$: inductive

Preliminaries: inductive set

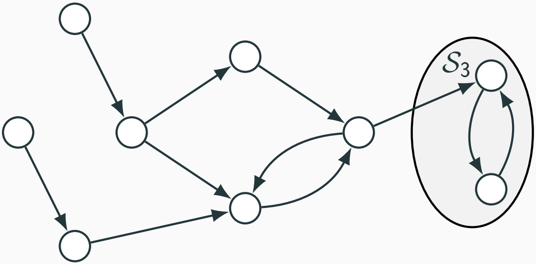
A (possibly infinite) set of configurations $\mathcal{S}$ is inductive iff it closed under reachability:



$\mathcal{S}_2$: inductive

Preliminaries: inductive set

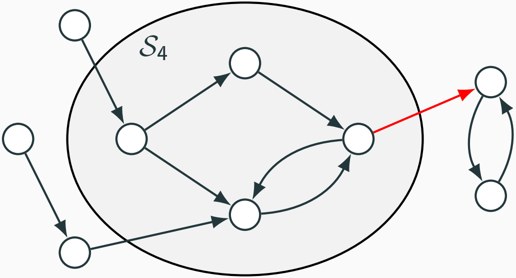
A (possibly infinite) set of configurations $\mathcal{S}$ is inductive iff it closed under reachability:



$\mathcal{S}_3$: inductive

Preliminaries: inductive set

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



$\mathcal{S}_4$: **not** inductive

Preliminaries: certificate

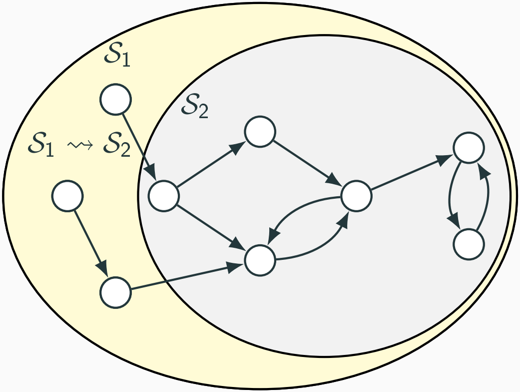
Let $\mathcal{S}, \mathcal{S}'$ be sets of configurations

- $\mathcal{S} \rightsquigarrow \mathcal{S}'$: runs starting at $\mathcal{S}$ visit $\mathcal{S}'$ with probability 1

Preliminaries: certificate

Let $\mathcal{S}, \mathcal{S}'$ be sets of configurations

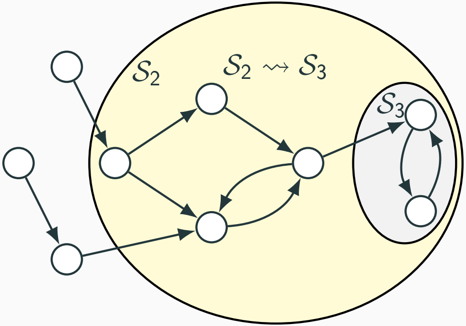
- $\mathcal{S} \rightsquigarrow \mathcal{S}'$: runs starting at $\mathcal{S}$ visit $\mathcal{S}'$ with probability 1



Preliminaries: certificate

Let $\mathcal{S}, \mathcal{S}'$ be sets of configurations

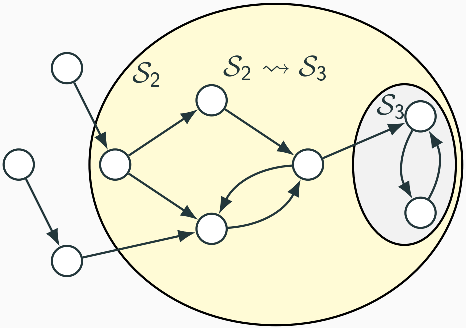
- $\mathcal{S} \rightsquigarrow \mathcal{S}'$: runs starting at $\mathcal{S}$ visit $\mathcal{S}'$ with probability 1



Preliminaries: certificate

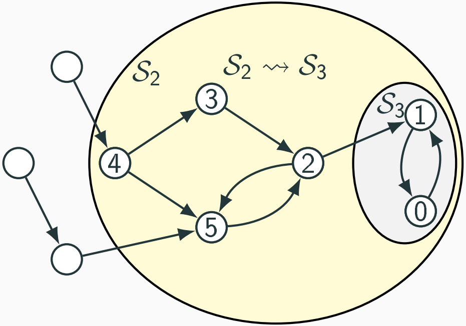
Let $\mathcal{S}, \mathcal{S}'$ be sets of configurations

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



Let $\mathcal{S}, \mathcal{S}'$ be sets of configurations

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



Stage Graphs

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

Stage Graphs

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

1. The nodes of the DAG, called stages, are inductive sets of configurations

Stage Graphs

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

1. The nodes of the DAG, called stages, are inductive sets of configurations
2. Every configuration of Pre belongs to some stage

Stage Graphs

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

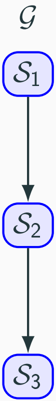
1. The nodes of the DAG, called stages, are inductive sets of configurations
2. Every configuration of Pre belongs to some stage
3. For every non-terminal stage $\mathcal{S}$ with children $\mathcal{S}_1, \ldots, \mathcal{S}_n$ there is a certificate for $\mathcal{S} \rightsquigarrow \mathcal{S}_1 \cup \cdots \cup \mathcal{S}_n$

Stage Graphs

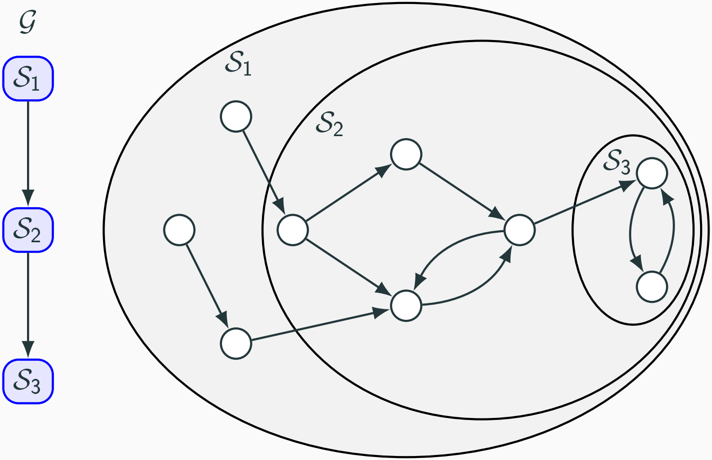
A stage graph for a given property $\mathrm{Pre} \to \mathbf{F}\left(\bigvee_{i=1}^{k} \mathbf{G}\, \mathrm{Post}_i\right)$ is a finite DAG satisfying:

1. The nodes of the DAG, called stages, are inductive sets of configurations
2. Every configuration of Pre belongs to some stage
3. For every non-terminal stage $\mathcal{S}$ with children $\mathcal{S}_1, \ldots, \mathcal{S}_n$ there is a certificate for $\mathcal{S} \rightsquigarrow \mathcal{S}_1 \cup \cdots \cup \mathcal{S}_n$
4. Every terminal stage is contained on some Post_i

Stage Graphs

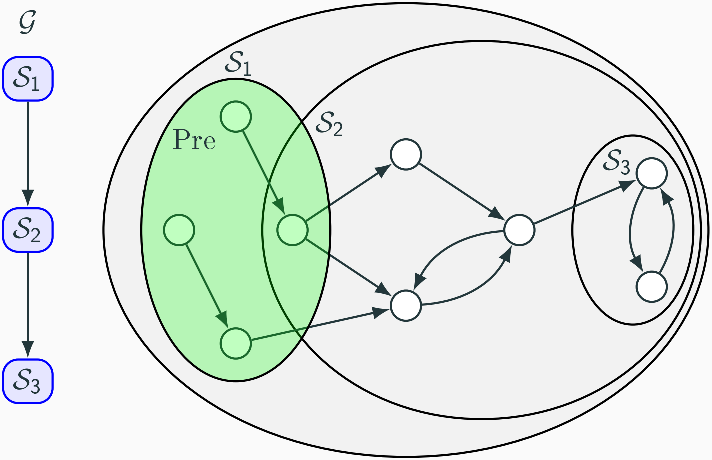


Stage Graphs



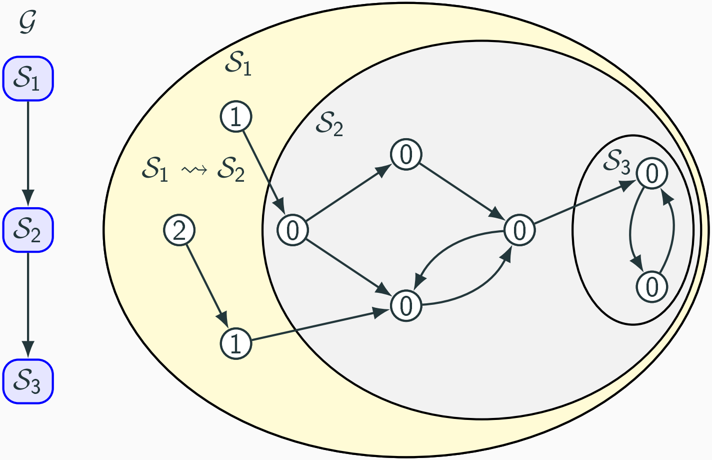
Stages $\mathcal{S}$ of $\mathcal{G}$ are inductive sets

Stage Graphs



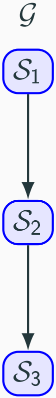
$\mathrm{Pre} \subseteq \bigcup_{\mathcal{S} \in \mathcal{G}} \mathcal{S}$

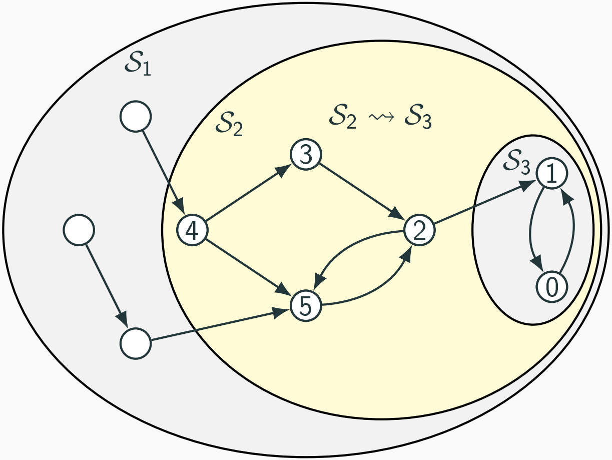
Stage Graphs



Certificates for non-terminal stages $\mathcal{S} \rightsquigarrow \underbrace{\mathcal{S}_1 \cup \ldots \cup \mathcal{S}_k}_{\text{children of } \mathcal{S}}$

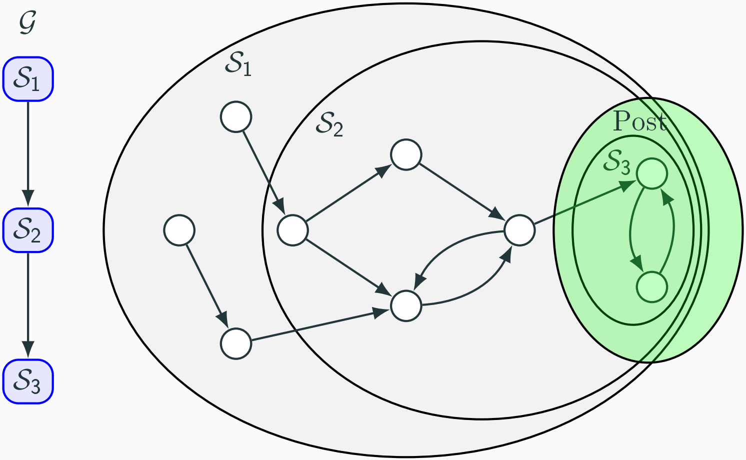
Stage Graphs





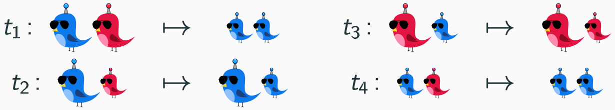
Certificates for non-terminal stages $\mathcal{S} \rightsquigarrow \underbrace{\mathcal{S}_1 \cup \ldots \cup \mathcal{S}_k}_{\text{children of } \mathcal{S}}$

Stage Graphs

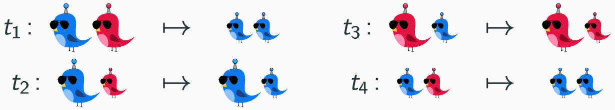


Terminal stages $\mathcal{S} \subseteq \mathrm{Post}$

Stage Graph Example: majority voting protocol

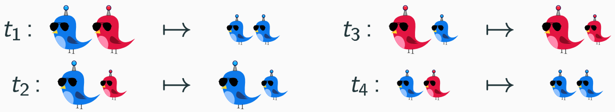


Stage Graph Example: majority voting protocol

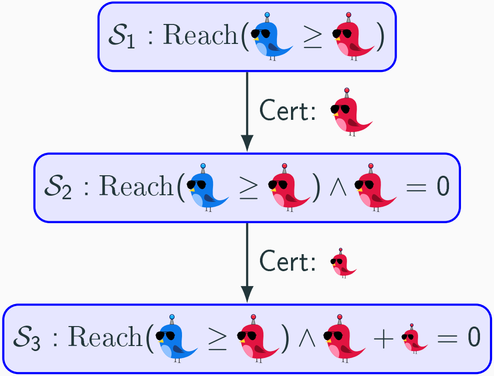


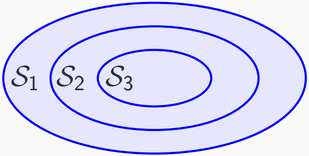
Stage graph for property ( $\geq$ ) $\implies$ **FG** ( $+$  $= 0$)

Stage Graph Example: majority voting protocol



Stage graph for property





Stage Graphs: theory

Soundness **CAV 2020**

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

Completeness **CAV 2020**

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

Stage Graphs: theory

Soundness **CAV 2020**

If there is a stage graph for a prop[illegible]en it holds.

EASY

Completeness **CAV 2020**

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

Stage Graphs: theory

Soundness **CAV 2020**

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

Completeness **CAV 2020**

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

What about decidability?

$\rightarrow$ unknown (stages can be arbitrarily complicated!)

Stage Graphs: theory

Soundness **CAV 2020**

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

Completeness **CAV 2020**

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

A Presburger stage graph is a stage graph where

- nodes are Presburger sets, and

$$C \in \mathcal{S} \iff \phi(C)$$

Stage Graphs: theory

Soundness **CAV 2020**

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

Completeness **CAV 2020**

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

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

Soundness **CAV 2020**

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

Completeness **CAV 2020**

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

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

Soundness **CAV 2020**

If there is a Presburger stage graph [illegible] property, then it holds.

HARD

Completeness **CAV 2020**

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

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

Soundness **CAV 2020**

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

Completeness **CAV 2020**

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

Presburger stage graph can be independently checked!

$\rightarrow$ everything reduces to checking Presburger formulas

Stage Graphs: theory

Soundness **CAV 2020**

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

Completeness **CAV 2020**

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

Decidability **CAV 2020**

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

Stage Graphs: theory

Soundness **CAV 2020**

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

Completeness **CAV 2020**

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

Decidability **CAV 2020**

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

(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

□

Stage Graphs: theory

Soundness **CAV 2020**

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

Completeness **CAV 2020**

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

Decidability **CAV 2020**

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

Problem: stage graphs might be huge (non-elementary)

$\rightarrow$ How can stage graphs help with automatic verification?

Stage Graphs: practice

Ideas:

- Most systems have small stage graphs

Stage Graphs: practice

Ideas:

- 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
  $\rightarrow$ search for stages with more and more dead 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
  $\rightarrow$ search for stages with more and more dead transitions

Definition

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

$$\Downarrow$$

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
  $\rightarrow$ search for stages with more and more dead transitions

Definition

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

$$\Downarrow$$

Algorithm:

SMT based semi-algorithm to automatically construct Presburger stage graphs

- reachability is TOWER-hard & not Presburger
  $\rightarrow$ overapproximate

Stage Graphs: practice

Ideas:

- Most systems have small stage graphs
- Most systems "make progress" by "killing" transitions
  $\rightarrow$ search for stages with more and more dead transitions

Definition

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

$$\Downarrow$$

Algorithm:

SMT based semi-algorithm to automatically construct Presburger stage graphs

- reachability is TOWER-hard & not Presburger
  $\rightarrow$ overapproximate
- use heuristics to find eventually dead transitions
  $\rightarrow$ find linear ranking functions

Experimental Results [1]

Automatically verifying population protocols:

Population Protocol	Predicate	$\|Q\|$	$\|T\|$	Time
Broadcast [31,22]	$x_1 \vee \ldots \vee x_n$	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]	$\Sigma_i \alpha_i x_i \geq 8$	76	2148	$1089s$
Threshold [20] ("succinct")	$\Sigma_i \alpha_i x_i \geq 511$	25	91	$2659s$
Remainder [22]	$\Sigma_i \alpha_i x_i \equiv_{20} 1$	22	230	$1646s$

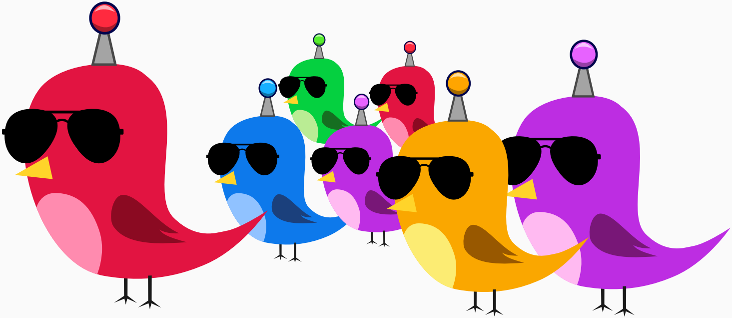
[1] Intel Xeon CPU E5-2630 v4 @ 2.20GHz and 8GB of RAM

Experimental Results [1]

Automatically verifying leader election algorithms:

Algorithm	Processes	$\|Q\|$	$\|T\|$	Time
Israeli-Jalfon [44]	20	40	80	$7s$
Israeli-Jalfon [44]	60	120	240	$1493s$
Israeli-Jalfon [44]	70	140	280	$3295s$
Herman [42]	21	42	42	$9s$
Herman [42]	51	102	102	$300s$
Herman [42]	81	162	162	$2800s$

[1] Intel Xeon CPU E5-2630 v4 @ 2.20GHz and 8GB of RAM



THANK YOU!