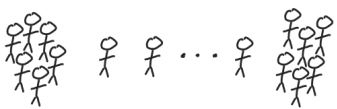
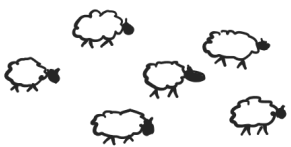
Gathering the herd
Control strategies for randomised populations



Corto Mascle

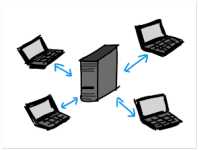
Journée EDMI 2026

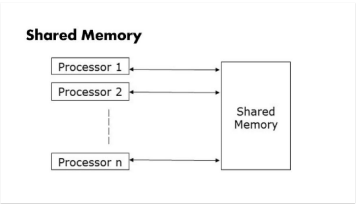



MPI-SWS
Kaiserslautern
2025 – ?
LaBRI
2021-2024

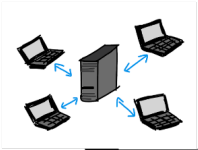

Paris
2h
2h30
MPI-SWS
Kaiserslautern
2025 – ?
LaBRI
2021-2024

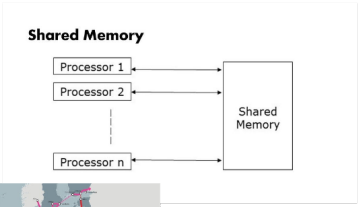
Distributed systems

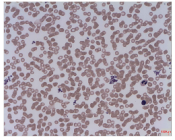


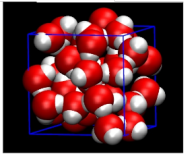


Distributed systems

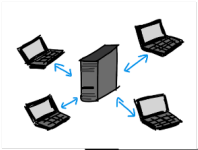


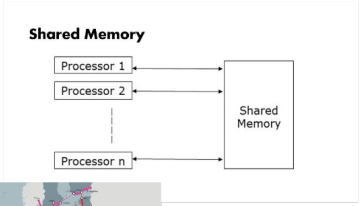


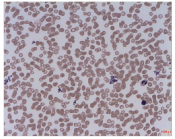


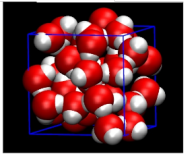


Distributed systems



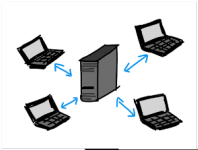


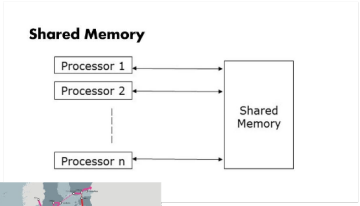


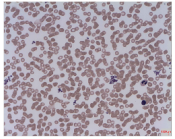


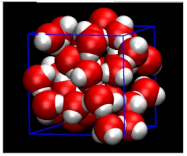
- Difficulties : combinatorial explosion, network failures, ...

Distributed systems



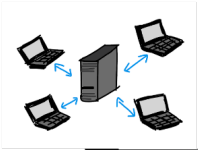


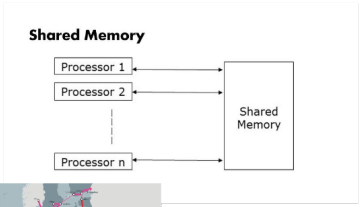


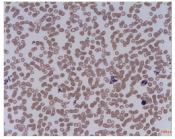


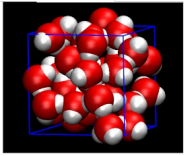
- Difficulties : combinatorial explosion, network failures, ...

Distributed systems









- Difficulties : combinatorial explosion, network failures, ...
- Deadlocks, starvations, race conditions, ...

Context - Models

Distributed models with an unbounded number of agents.

Context - Models

Distributed models with an unbounded number of agents.

Parameterized models

An arbitrary number of identical agents

Ad-hoc networks, rendez-vous protocols, ...

Dynamic models

The number of processes may increase during runtime

Multi-threaded programs, Petri nets, ...

Context - Models

Distributed models with an unbounded number of agents.

Parameterized models

An arbitrary number of identical agents

Ad-hoc networks, rendez-vous protocols, ...

Dynamic models

The number of processes may increase during runtime

Multi-threaded programs, Petri nets, ...

System >>>> **Mathematical abstraction** >>>> **Formal analysis**

Context - Problems

Verification

Input - System S + specification φ
Output - Does S satisfy φ

Context - Problems

Verification

Input - System S + specification φ
Output - Does S satisfy φ

S: Automatic Ship

φ: "If no storm
then - does not sink
- reaches destination"

Context - Problems

Verification

Input - System S + specification φ
Output - Does S satisfy φ

S: Automatic Ship

φ: "If no storm then - does not sink - reaches destination"

Controller synthesis

Input - **Controllable** system S + specification φ
Output - A controller guaranteeing φ

Context - Problems

Verification

Input - System S + specification φ
Output - Does S satisfy φ

S: Automatic Ship

φ: "If no storm then - does not sink - reaches destination"

Controller synthesis

Input - **Controllable** system S + specification φ
Output - A controller guaranteeing φ

S: Boat

φ: "

Output: Programs that maneuvers the boat so that φ holds

Control of Markovian populations

Based on work with Hugo Gimbert and Patrick Totzke

Control of Markovian populations

Based on work with Hugo Gimbert and Patrick Totzke

Context

- Control a population of yeasts to make them reach a target state

Control of Markovian populations

Based on work with Hugo Gimbert and Patrick Totzke

Context

- Control a population of yeasts to make them reach a target state
- [Bertrand Dewaskar Genest Gimbert '17]$\rightarrow$ Modelled by a two-player game

Control of Markovian populations

Based on work with Hugo Gimbert and Patrick Totzke

Context

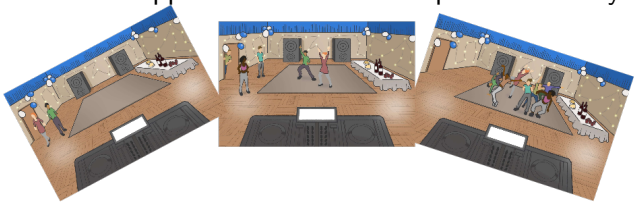
- Control a population of yeasts to make them reach a target state
- [Bertrand Dewaskar Genest Gimbert '17]$\rightarrow$ Modelled by a two-player game
- What happens when cells react probabilistically?

Control of Markovian populations

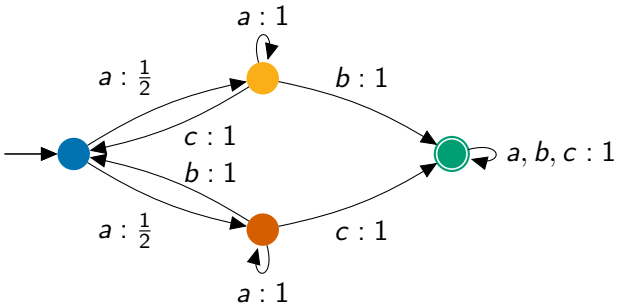
Based on work with Hugo Gimbert and Patrick Totzke

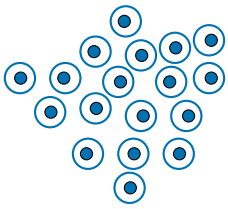
Context

- Control a population of yeasts to make them reach a target state
- [Bertrand Dewaskar Genest Gimbert '17]→ Modelled by a two-player game
- What happens when cells react probabilistically?

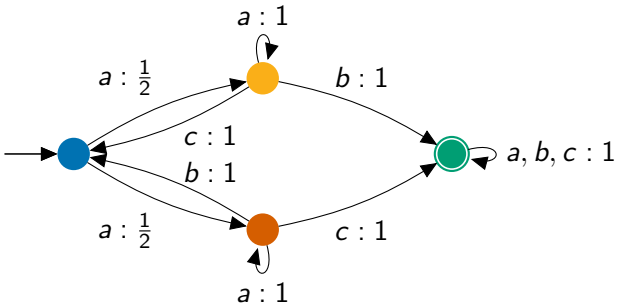


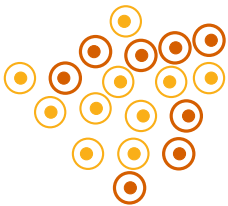
Control of Markovian processes



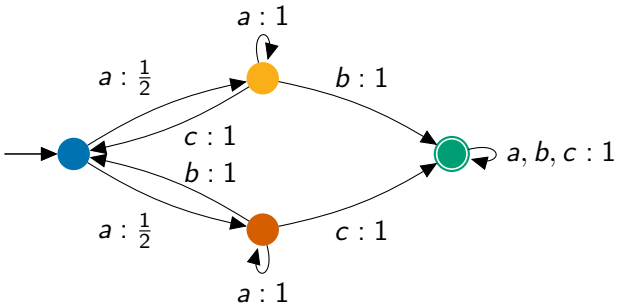


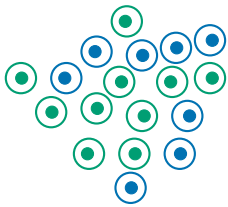
Control of Markovian processes



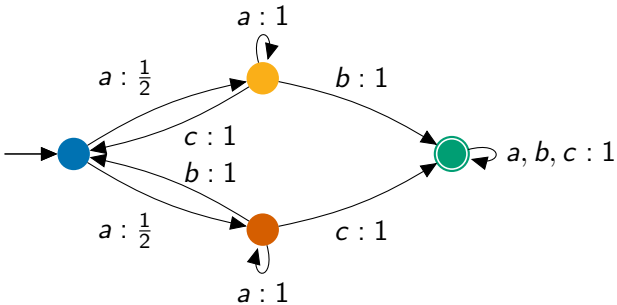


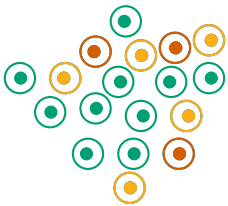
Control of Markovian processes



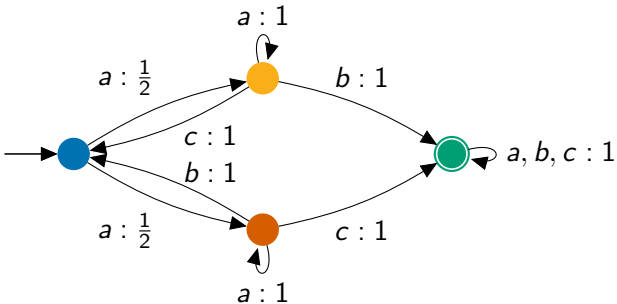


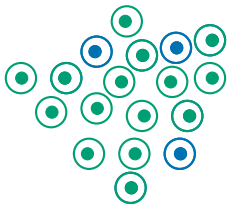
Control of Markovian processes



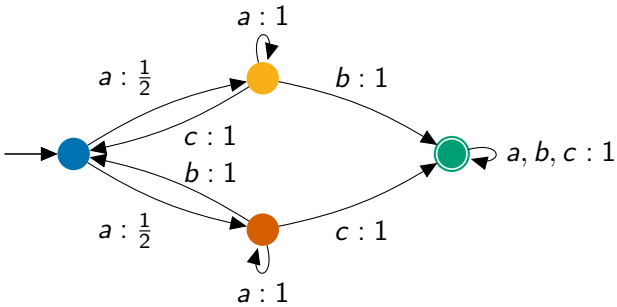


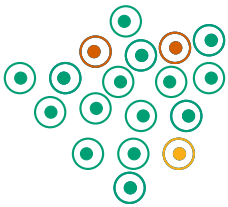
Control of Markovian processes



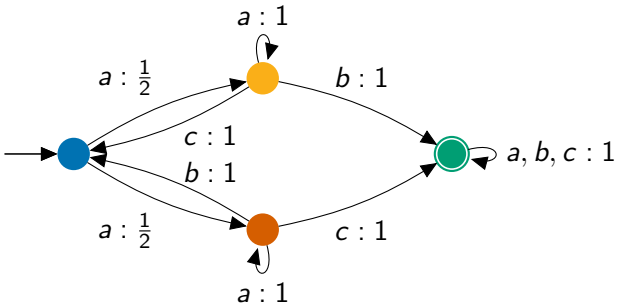


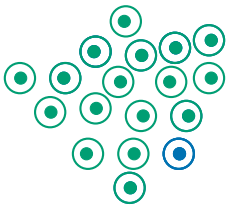
Control of Markovian processes



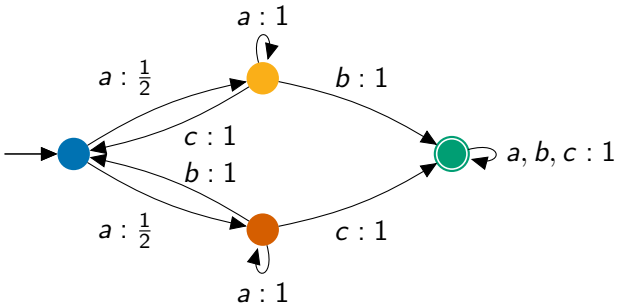


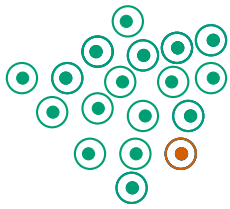
Control of Markovian processes



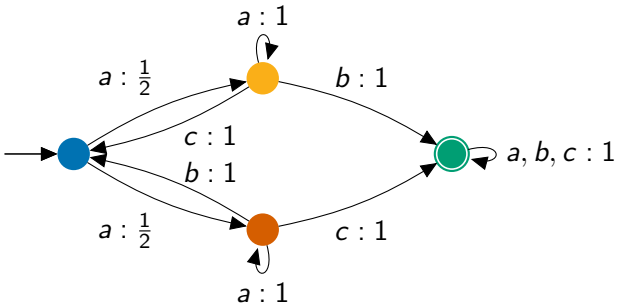


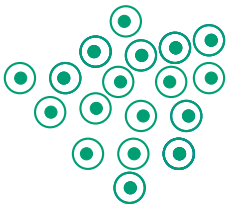
Control of Markovian processes



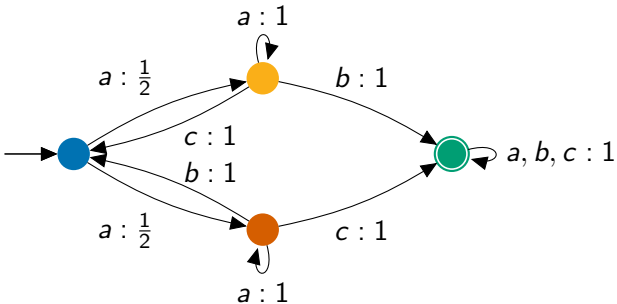


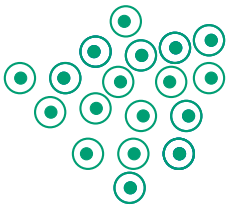
Control of Markovian processes





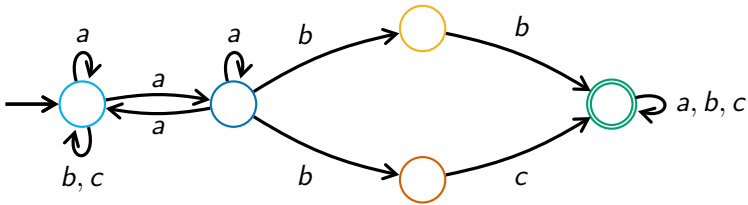
Control of Markovian processes



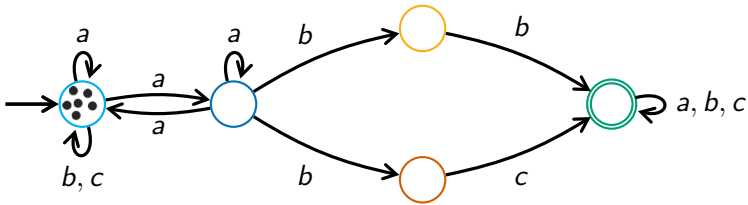


Problem: Compute a **control strategy** to **reach**  with proba 1 **for any number of cells**.

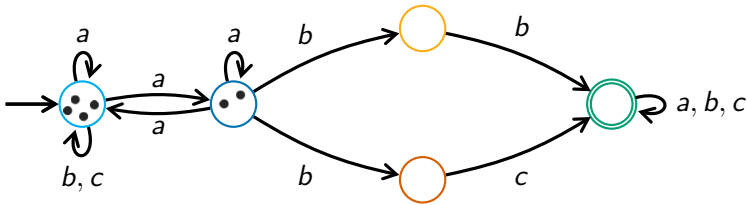
Control of Markovian processes



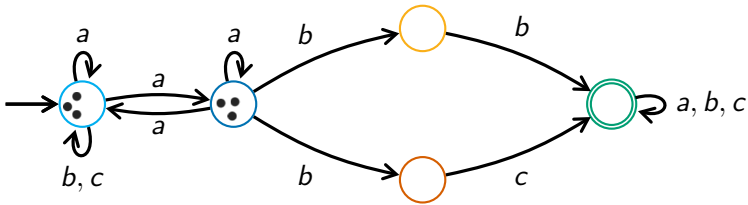
Control of Markovian processes



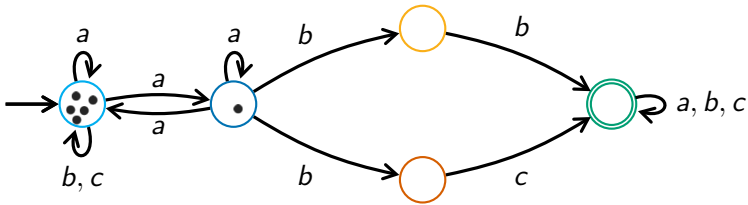
Control of Markovian processes



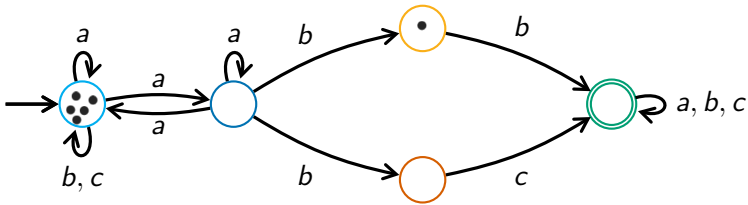
Control of Markovian processes



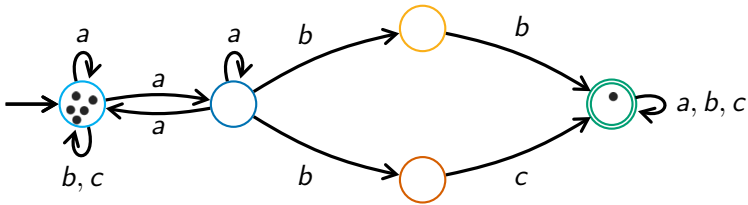
Control of Markovian processes



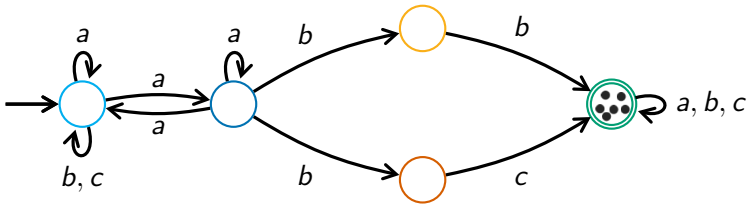
Control of Markovian processes



Control of Markovian processes

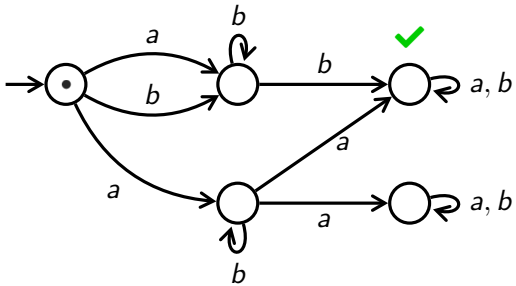


Control of Markovian processes



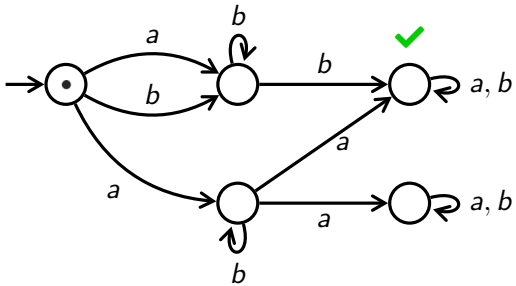
Ingredient I - Winning regions

Let's say we have only one cell.



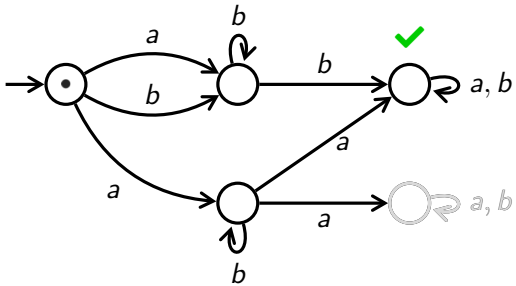
Ingredient I - Winning regions

Let's say we have only one cell.



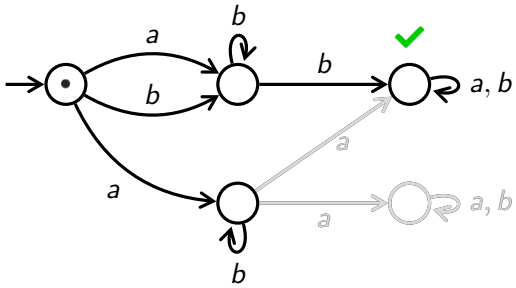
Ingredient I - Winning regions

Let's say we have only one cell.



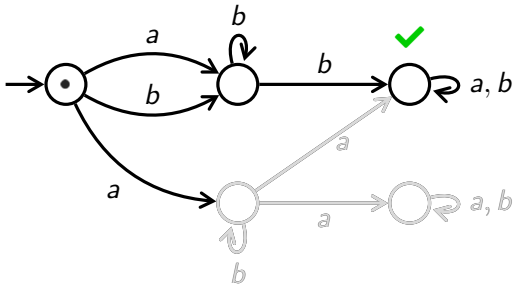
Ingredient I - Winning regions

Let's say we have only one cell.



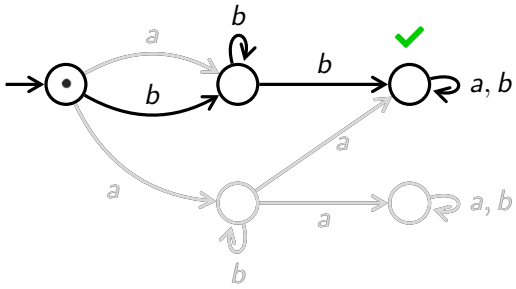
Ingredient I - Winning regions

Let's say we have only one cell.



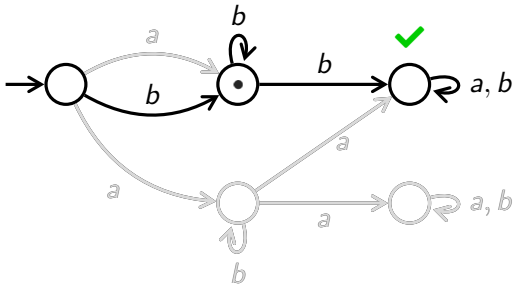
Ingredient I - Winning regions

Let's say we have only one cell.



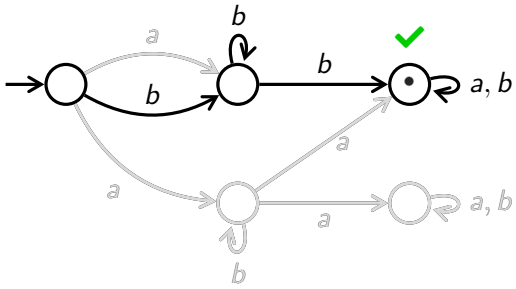
Ingredient I - Winning regions

Let's say we have only one cell.



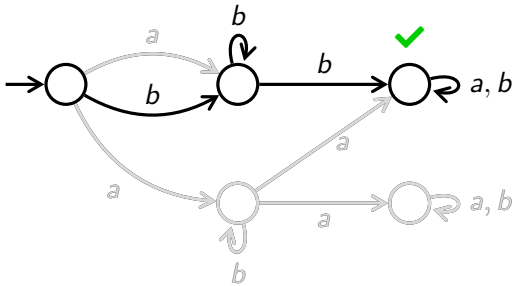
Ingredient I - Winning regions

Let's say we have only one cell.



Ingredient I - Winning regions

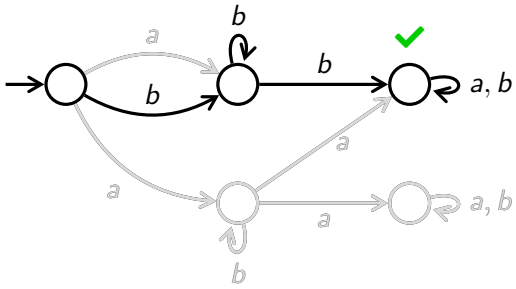
Let's say we have only one cell.



Winning region = set of configurations from which we can reach the target with probability 1.

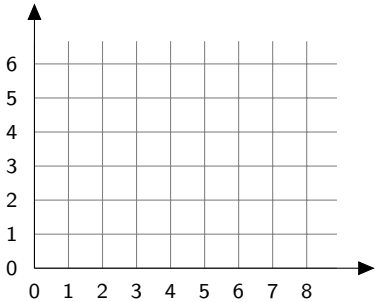
Ingredient I - Winning regions

Let's say we have only one cell.



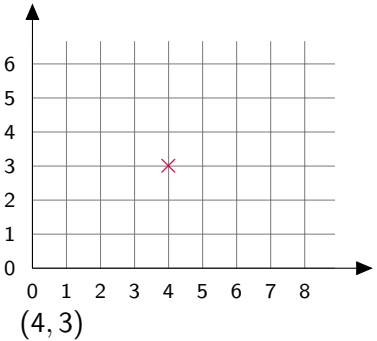
Winning region = set of configurations from which we can reach the target with probability 1.
In general, the winning region is a subset of $\mathbb{N}^S$.

Ingredient II - Well quasi-orders



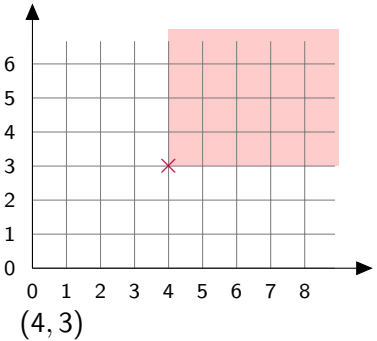
- Product ordering: $(x, y) \leq (z, t)$ when $x \leq z$ and $y \leq t$.
- You cannot pick a point higher than one of the previous ones.

Ingredient II - Well quasi-orders



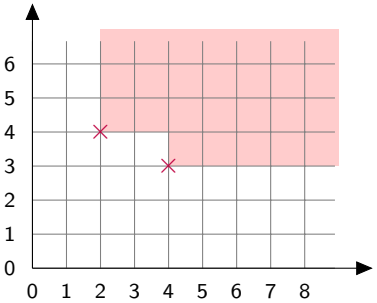
- Product ordering: $(x, y) \leq (z, t)$ when $x \leq z$ and $y \leq t$.
- You cannot pick a point higher than one of the previous ones.

Ingredient II - Well quasi-orders



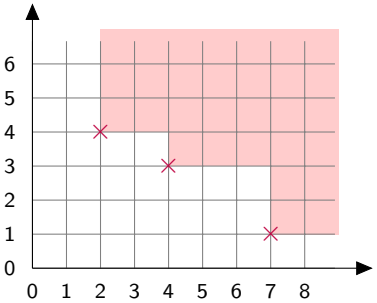
- Product ordering: $(x, y) \leq (z, t)$ when $x \leq z$ and $y \leq t$.
- You cannot pick a point higher than one of the previous ones.

Ingredient II - Well quasi-orders



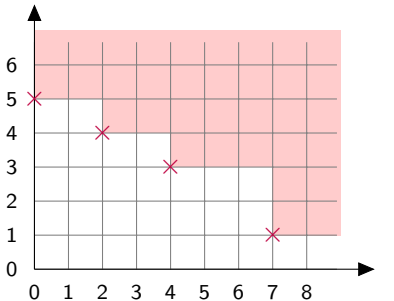
- Product ordering: $(x, y) \leq (z, t)$ when $x \leq z$ and $y \leq t$.
- You cannot pick a point higher than one of the previous ones.

Ingredient II - Well quasi-orders



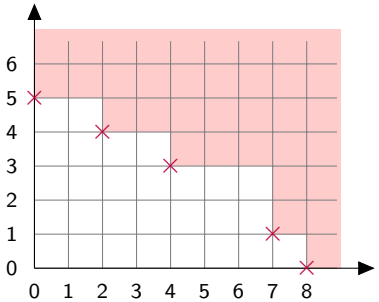
- Product ordering: $(x, y) \leq (z, t)$ when $x \leq z$ and $y \leq t$.
- You cannot pick a point higher than one of the previous ones.

Ingredient II - Well quasi-orders



- Product ordering: $(x, y) \leq (z, t)$ when $x \leq z$ and $y \leq t$.
- You cannot pick a point higher than one of the previous ones.

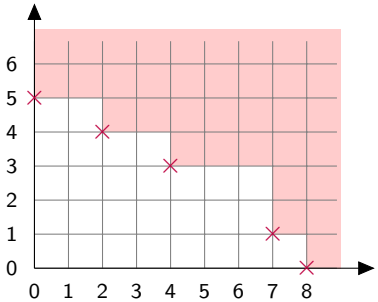
Ingredient II - Well quasi-orders



$(4,3) \to (2,4) \to (7,1) \to (0,5) \to (8,0)$

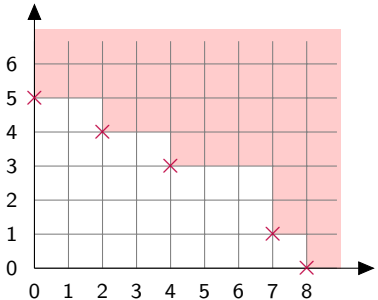
- Product ordering: $(x, y) \leq (z, t)$ when $x \leq z$ and $y \leq t$.
- You cannot pick a point higher than one of the previous ones.

Ingredient II - Well quasi-orders



- Product ordering: $(x, y) \leq (z, t)$ when $x \leq z$ and $y \leq t$.
- You cannot pick a point higher than one of the previous ones.

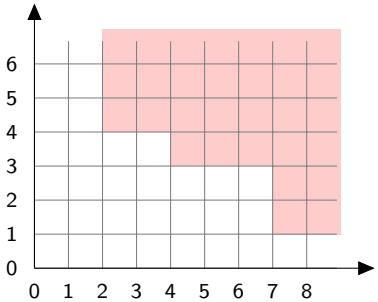
Ingredient II - Well quasi-orders



- Product ordering: $(x, y) \leq (z, t)$ when $x \leq z$ and $y \leq t$.
- You cannot pick a point higher than one of the previous ones.

Every set of incomparable elements for the product ordering is finite.

Ingredient II - Well quasi-orders

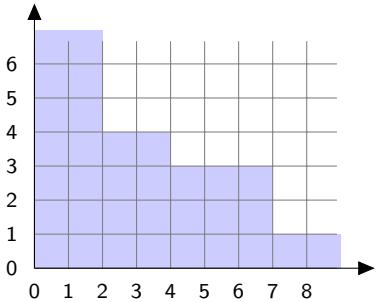


- Product ordering: $(x, y) \leq (z, t)$ when $x \leq z$ and $y \leq t$.
- You cannot pick a point higher than one of the previous ones.

Every set of incomparable elements for the product ordering is finite.

Every upward-closed set of $\mathbb{N}^S$ has a finite representation of the form $(2,4)\uparrow \cup (4,3)\uparrow \cup (7,1)\uparrow$.

Ingredient II - Well quasi-orders



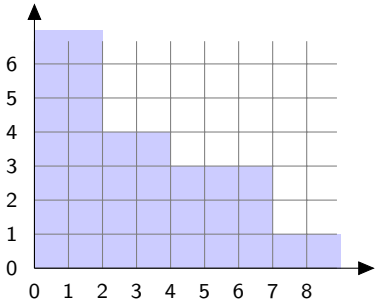
- Product ordering: $(x, y) \leq (z, t)$ when $x \leq z$ and $y \leq t$.
- You cannot pick a point higher than one of the previous ones.

Every set of incomparable elements for the product ordering is finite.

Every upward-closed set of $\mathbb{N}^S$ has a finite representation of the form $(2,4)\uparrow \cup (4,3)\uparrow \cup (7,1)\uparrow$.

Every downward-closed set of $\mathbb{N}^S$ has a finite representation of the form $(\omega,1)\downarrow \cup (2,\omega)\downarrow \cup (4,4)\downarrow \cup (7,3)\downarrow$.

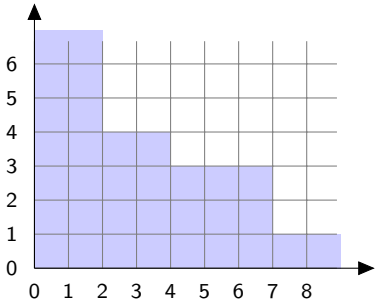
Ingredient II - Well quasi-orders



- Product ordering: $(x, y) \leq (z, t)$ when $x \leq z$ and $y \leq t$.
- You cannot pick a point higher than one of the previous ones.

Every downward-closed set of $\mathbb{N}^S$ has a finite representation of the form $(\omega, 1)\downarrow \cup (2, \omega)\downarrow \cup (4, 4)\downarrow \cup (7, 3)\downarrow$.

Ingredient II - Well quasi-orders



- Product ordering: $(x, y) \leq (z, t)$ when $x \leq z$ and $y \leq t$.
- You cannot pick a point higher than one of the previous ones.

Every downward-closed set of $\mathbb{N}^S$ has a finite representation of the form $(\omega, 1)\downarrow \cup (2, \omega)\downarrow \cup (4, 4)\downarrow \cup (7, 3)\downarrow$.

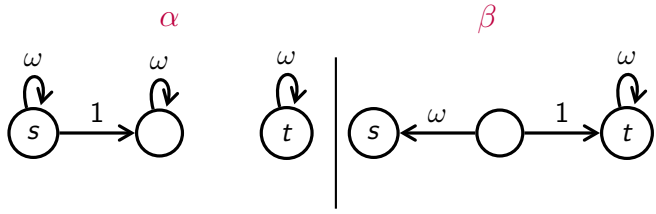
The **winning region** of a population control instance is downward-closed.

Ingredient III - Sequential flows

Set of states S
Tile = function
$S \times S \to \mathbb{N} \cup \{+\infty\}$

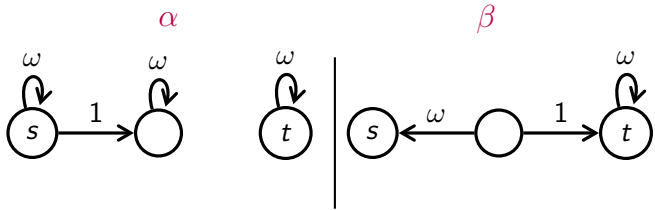
Ingredient III - Sequential flows

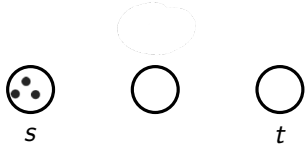
Set of states S
Tile = function
$S \times S \to \mathbb{N} \cup \{+\infty\}$
ω = unbounded, no edge = 0.



Ingredient III - Sequential flows

Set of states S
Tile = function
$S \times S \to \mathbb{N} \cup \{+\infty\}$
ω = unbounded, no edge = 0.



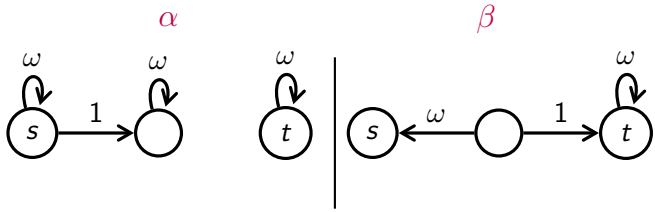


Problem [Colcombet, Fijalkow, Ohlmann '20]

Input: A set of tiles
Output: Maximum number of cells we can transfer from s to t?

Ingredient III - Sequential flows

Set of states S
Tile = function
$S \times S \to \mathbb{N} \cup \{+\infty\}$
ω = unbounded, no edge = 0.

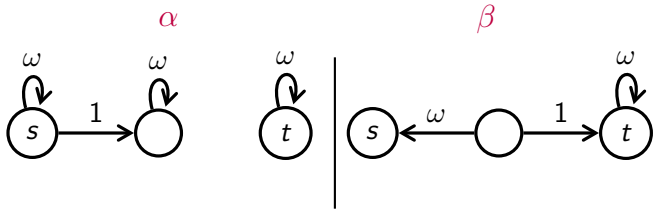


Problem [Colcombet, Fijalkow, Ohlmann '20]

Input: A set of tiles
Output: Maximum number of cells we can transfer from s to t?

Ingredient III - Sequential flows

Set of states S
Tile = function
$S \times S \to \mathbb{N} \cup \{+\infty\}$
ω = unbounded, no edge = 0.

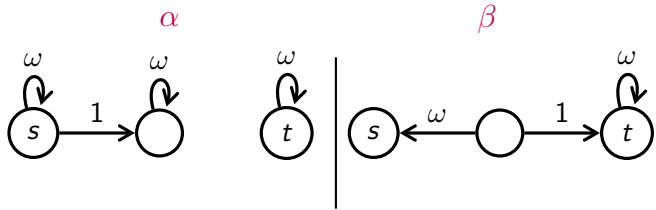


Problem [Colcombet, Fijalkow, Ohlmann '20]

Input: A set of tiles
Output: Maximum number of cells we can transfer from s to t?

Ingredient III - Sequential flows

Set of states S
Tile = function
$S \times S \to \mathbb{N} \cup \{+\infty\}$
ω = unbounded, no edge = 0.

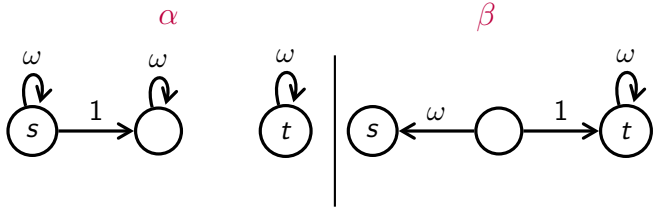


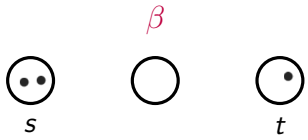
Problem [Colcombet, Fijalkow, Ohlmann '20]

Input: A set of tiles
Output: Maximum number of cells we can transfer from s to t?

Ingredient III - Sequential flows

Set of states S
Tile = function
$S \times S \to \mathbb{N} \cup \{+\infty\}$
ω = unbounded, no edge = 0.



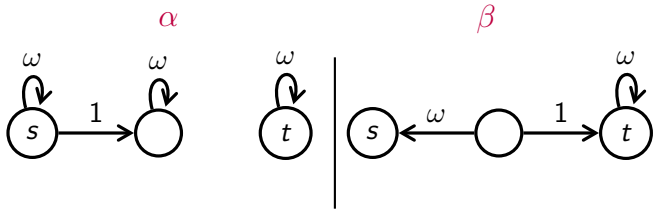


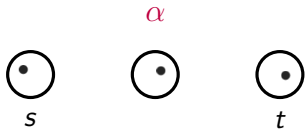
Problem [Colcombet, Fijalkow, Ohlmann '20]

Input: A set of tiles
Output: Maximum number of cells we can transfer from s to t?

Ingredient III - Sequential flows

Set of states S
Tile = function
$S \times S \to \mathbb{N} \cup \{+\infty\}$
ω = unbounded, no edge = 0.



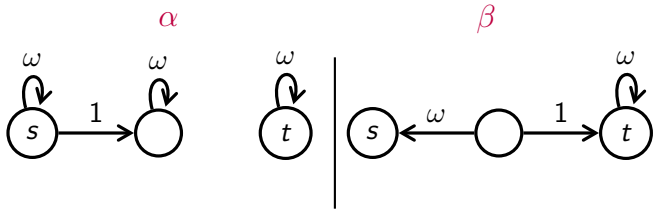


Problem [Colcombet, Fijalkow, Ohlmann '20]

Input: A set of tiles
Output: Maximum number of cells we can transfer from s to t?

Ingredient III - Sequential flows

Set of states S
Tile = function
$S \times S \to \mathbb{N} \cup \{+\infty\}$
ω = unbounded, no edge = 0.

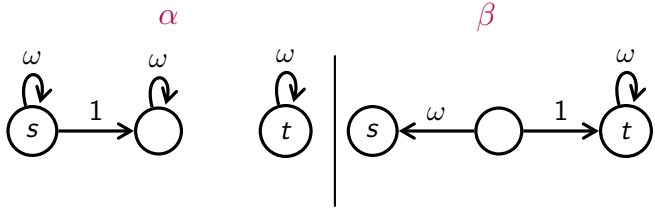


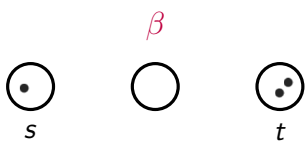
Problem [Colcombet, Fijalkow, Ohlmann '20]

Input: A set of tiles
Output: Maximum number of cells we can transfer from s to t?

Ingredient III - Sequential flows

Set of states S
Tile = function
$S \times S \to \mathbb{N} \cup \{+\infty\}$
ω = unbounded, no edge = 0.



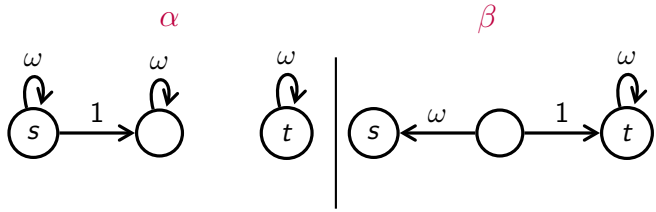


Problem [Colcombet, Fijalkow, Ohlmann '20]

Input: A set of tiles
Output: Maximum number of cells we can transfer from s to t?

Ingredient III - Sequential flows

Set of states S
Tile = function
$S \times S \to \mathbb{N} \cup \{+\infty\}$
ω = unbounded, no edge = 0.

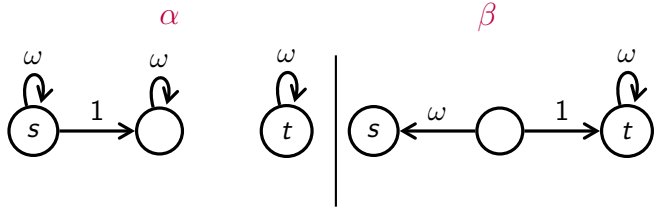


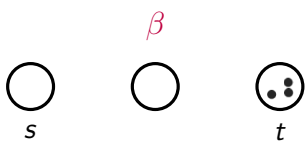
Problem [Colcombet, Fijalkow, Ohlmann '20]

Input: A set of tiles
Output: Maximum number of cells we can transfer from s to t?

Ingredient III - Sequential flows

Set of states S
Tile = function
$S \times S \to \mathbb{N} \cup \{+\infty\}$
ω = unbounded, no edge = 0.



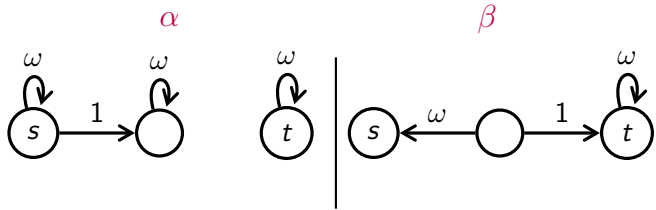


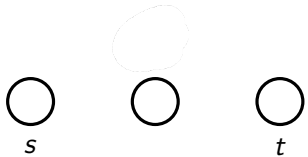
Problem [Colcombet, Fijalkow, Ohlmann '20]

Input: A set of tiles
Output: Maximum number of cells we can transfer from s to t?

Ingredient III - Sequential flows

Set of states S
Tile = function
$S \times S \to \mathbb{N} \cup \{+\infty\}$
ω = unbounded, no edge = 0.





Problem [Colcombet, Fijalkow, Ohlmann '20]

Input: A set of tiles
Output: Maximum number of cells we can transfer from s to t?

Solution: Algebra + abstractions $(\alpha^{\#}\beta)^{\#}$

Results

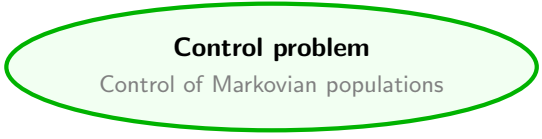
Theorem [Colcombet, Fijalkow, Ohlmann '20]

The Markovian population control problem is decidable.

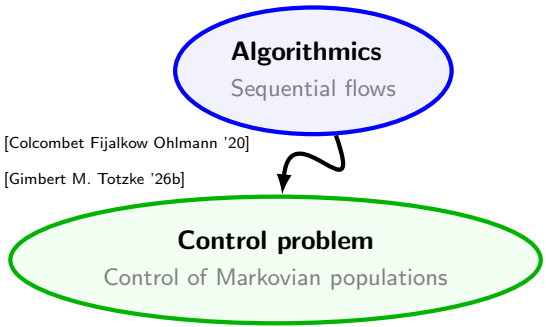
Theorem [Gimbert, M., Totzke '26]

The Markovian population control problem is EXPTIME-complete.

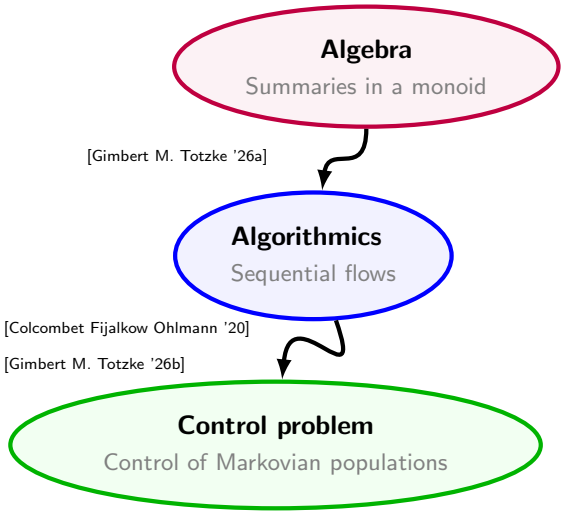
Consequences



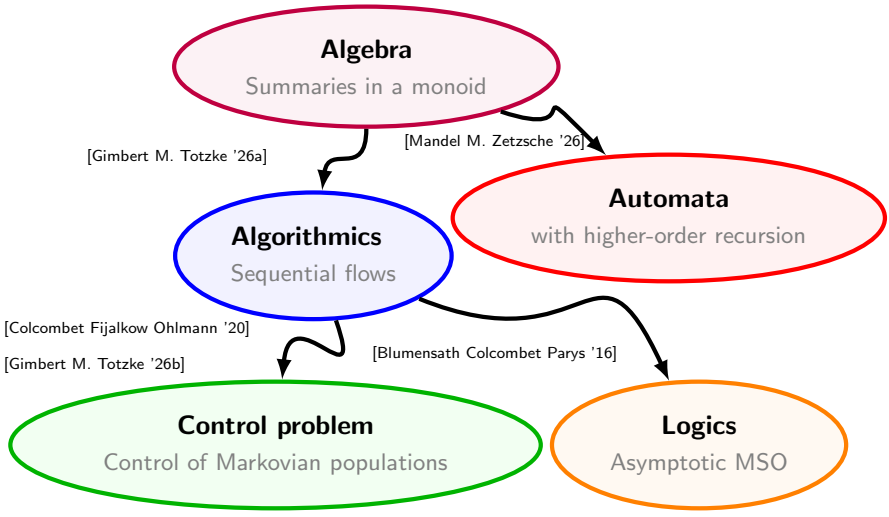
Consequences



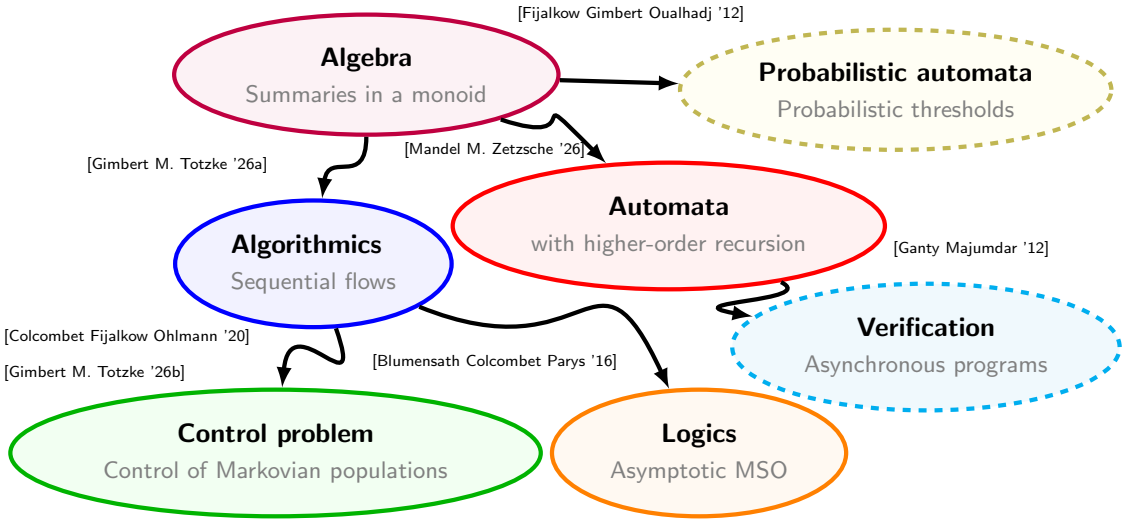
Consequences



Consequences



Consequences

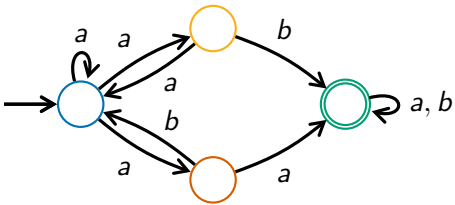


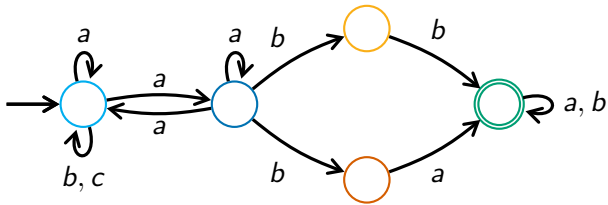
Future work

- Quantitative objectives
- Infinite-horizon specifications

Future work

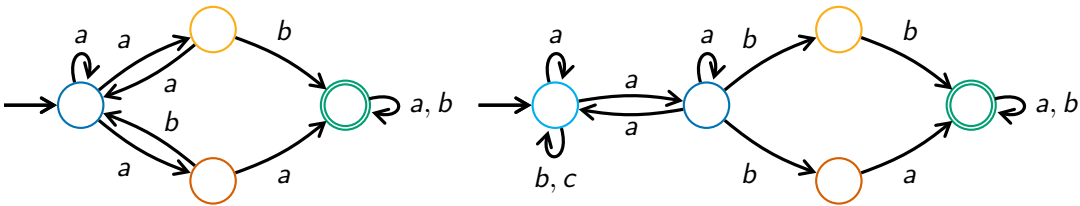
- Quantitative objectives
- Infinite-horizon specifications
- **Fast strategies**





Future work

- Quantitative objectives
- Infinite-horizon specifications
- **Fast strategies**



Thank you for your attention!