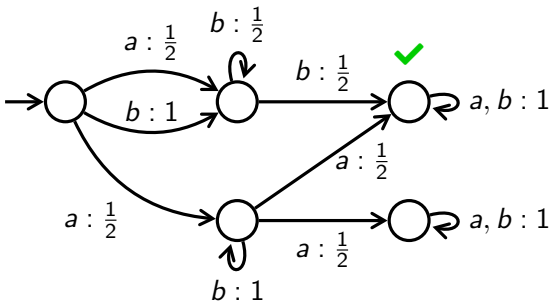
The Short-Ends Factorisation Theorem and applications

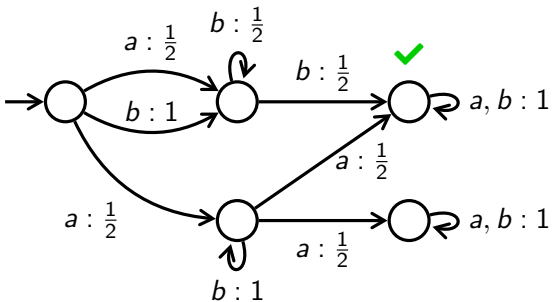
Corto Mascle
MPI-SWS Kaiserslautern

Based on joint work with
Hugo Gimbert + Patrick Totzke and Richard Mandel + Georg Zetzsche
LaBRI, Bordeaux University of Liverpool MPI-SWS Kaiserslautern

Controlling a random population

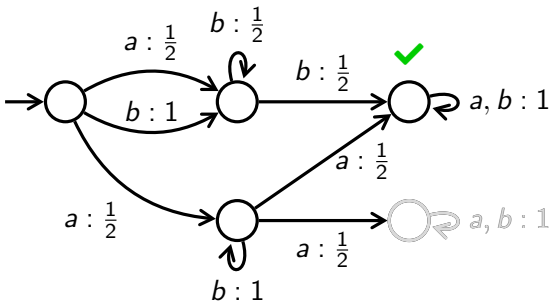


Controlling a random population



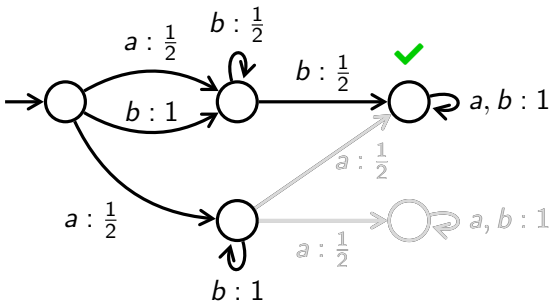
Given a Markov decision process, make it reach ✔ with probability 1.

Controlling a random population



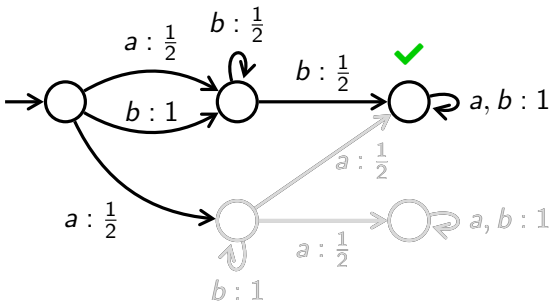
Given a Markov decision process, make it reach ✔ with probability 1.

Controlling a random population



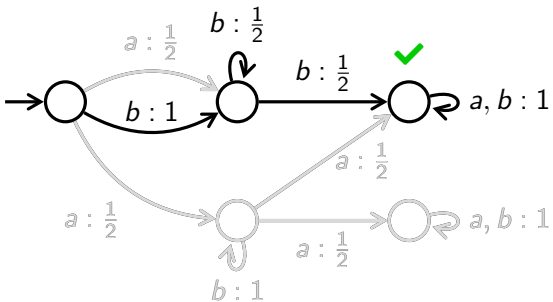
Given a Markov decision process, make it reach ✔ with probability 1.

Controlling a random population



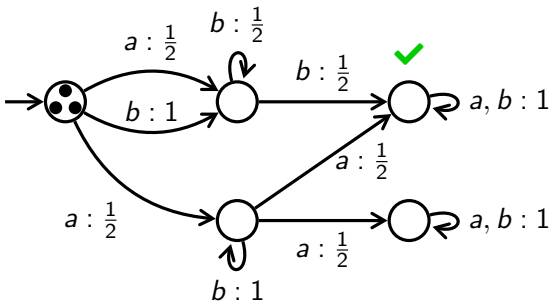
Given a Markov decision process, make it reach ✔ with probability 1.

Controlling a random population



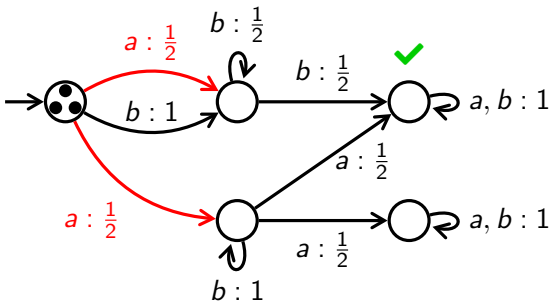
Given a Markov decision process, make it reach ✔ with probability 1.

Controlling a random population



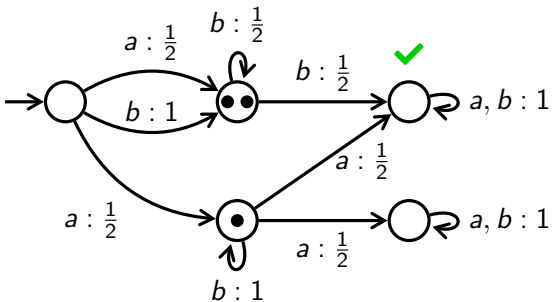
Given N identical Markov decision process, make them all reach ✔ with probability 1.

Controlling a random population



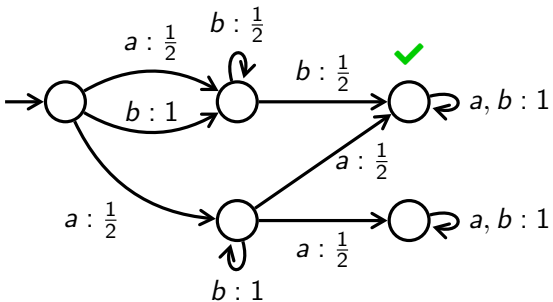
Given N identical Markov decision process, make them all reach ✔ with probability 1.

Controlling a random population



Given N identical Markov decision process, make them all reach ✔ with probability 1.

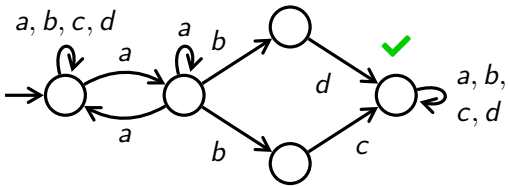
Controlling a random population



Given N identical Markov decision process, make them all reach ✔ with probability 1.

$\Rightarrow$ Product of N MDPs.

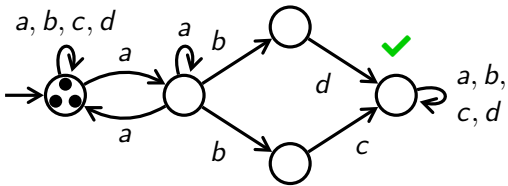
Controlling a random population



Random population control problem

Given an MDP $\mathcal{M}$, is there a winning strategy against N tokens, for all N?

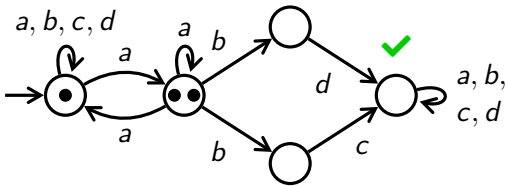
Controlling a random population



Random population control problem

Given an MDP $\mathcal{M}$, is there a winning strategy against N tokens, for all N?

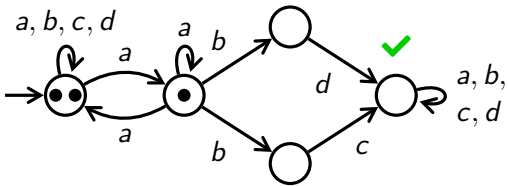
Controlling a random population



Random population control problem

Given an MDP $\mathcal{M}$, is there a winning strategy against N tokens, for all N?

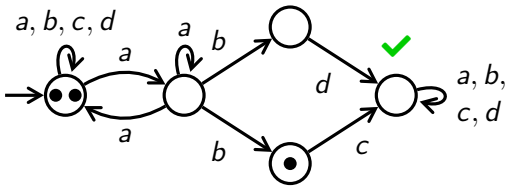
Controlling a random population



Random population control problem

Given an MDP $\mathcal{M}$, is there a winning strategy against N tokens, for all N?

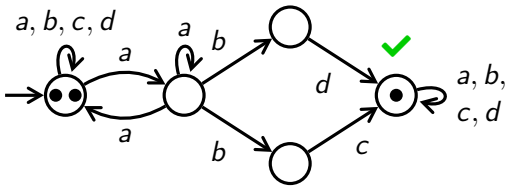
Controlling a random population



Random population control problem

Given an MDP $\mathcal{M}$, is there a winning strategy against N tokens, for all N?

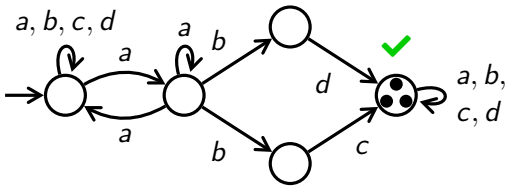
Controlling a random population



Random population control problem

Given an MDP $\mathcal{M}$, is there a winning strategy against N tokens, for all N?

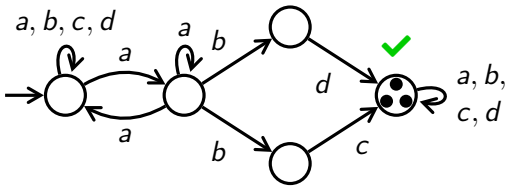
Controlling a random population



Random population control problem

Given an MDP $\mathcal{M}$, is there a winning strategy against N tokens, for all N?

Controlling a random population



Random population control problem

Given an MDP $\mathcal{M}$, is there a winning strategy against N tokens, for all N?

Adapted from [Bertrand Dewaskar Genest Gimbert '15]

There are MDPs of size k for which we can win against 2^{2^k} tokens and not more.

Controlling a random population

- Less tokens is always better $\rightsquigarrow$ The set of winning configurations is *downward-closed*.

Controlling a random population

- Less tokens is always better $\rightsquigarrow$ The set of winning configurations is *downward-closed*.
- The set of configurations from which action a is safe is *downward-closed*.

Controlling a random population

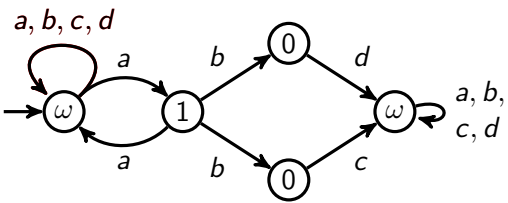
- Less tokens is always better $\rightsquigarrow$ The set of winning configurations is *downward-closed*.
- The set of configurations from which action a is safe is *downward-closed*.
- Downward-closed sets of configurations can be represented as finite unions of *ideals*.

$$(\omega, 5, \omega, 8)\downarrow \cup (4, \omega, \omega, 4)\downarrow \cup (9, 6, 1, 4)\downarrow$$

Controlling a random population

- Less tokens is always better $\rightsquigarrow$ The set of winning configurations is *downward-closed*.
- The set of configurations from which action a is safe is *downward-closed*.
- Downward-closed sets of configurations can be represented as finite unions of *ideals*.

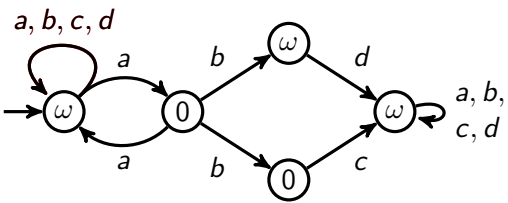
$$(\omega, 5, \omega, 8)\downarrow \cup (4, \omega, \omega, 4)\downarrow \cup (9, 6, 1, 4)\downarrow$$



Controlling a random population

- Less tokens is always better $\rightsquigarrow$ The set of winning configurations is *downward-closed*.
- The set of configurations from which action a is safe is *downward-closed*.
- Downward-closed sets of configurations can be represented as finite unions of *ideals*.

$$(\omega, 5, \omega, 8)\downarrow \cup (4, \omega, \omega, 4)\downarrow \cup (9, 6, 1, 4)\downarrow$$



Controlling a random population [Colcombet Fijalkow Ohlmann '20]

Given N identical Markov decision processes, make them all reach ✔ with probability 1.

$S \leftarrow \mathbb{N}^Q$ → configurations
$C \leftarrow \mathbb{N}^Q \times \Sigma$ → commits
while not fixpoint **do**
  if $\exists (s,a) \in C$, $s \xrightarrow{a} s'$, $s' \notin S$ **then**
    $C \leftarrow C \setminus \{(s,a)\} \uparrow$
  if $\exists s \in S$, no path from s to F in C **then**
    $S \leftarrow S \setminus \{s\} \uparrow$, $C \leftarrow C \cap S \times \Sigma$
return $I \subseteq S$

Controlling a random population [Colcombet Fijalkow Ohlmann '20]

Given N identical Markov decision processes, make them all reach ✔ with probability 1.

Terminates by well quasi-order argument

$S \leftarrow \mathbb{N}^Q$ → configurations
$C \leftarrow \mathbb{N}^Q \times \Sigma$ → commits

while not fixpoint **do**
  if $\exists (s,a) \in C$, $s \xrightarrow{a} s'$, $s' \notin S$ **then**
    $C \leftarrow C \setminus \{(s,a)\}\uparrow$
  if $\exists s \in S$, no path from s to F in C **then**
    $S \leftarrow S \setminus \{s\}\uparrow$, $C \leftarrow C \cap S \times \Sigma$
return $I \subseteq S$

Controlling a random population [Colcombet Fijalkow Ohlmann '20]

Given N identical Markov decision processes, make them all reach ✔ with probability 1.

$S \leftarrow \mathbb{N}^Q$ → configurations

$C \leftarrow \mathbb{N}^Q \times \Sigma$ → commits

while not fixpoint **do**

  if $\exists (s,a) \in C$, $s \xrightarrow{a} s'$, $s' \notin S$ **then**

    $C \leftarrow C \setminus \{(s,a)\}\uparrow$

  if $\exists s \in S$, no path from s to F in C **then**

    $S \leftarrow S \setminus \{s\}\uparrow$, $C \leftarrow C \cap S \times \Sigma$

return $I \subseteq S$

Easy to check

Terminates by well quasi-order argument

Controlling a random population [Colcombet Fijalkow Ohlmann '20]

Given N identical Markov decision processes, make them all reach ✔ with probability 1.

$S \leftarrow \mathbb{N}^Q$ → configurations

$C \leftarrow \mathbb{N}^Q \times \Sigma$ → commits

while not fixpoint **do**

  if $\exists (s,a) \in C,\ s \xrightarrow{a} s',\ s' \notin S$ **then**

    $C \leftarrow C \setminus \{(s,a)\}\uparrow$

  if $\exists s \in S$, no path from s to F in C **then**

    $S \leftarrow S \setminus \{s\}\uparrow,\ C \leftarrow C \cap S \times \Sigma$

return $I \subseteq S$

Easy to check

Unclear

Terminates by well quasi-order argument

Controlling a random population [Colcombet Fijalkow Ohlmann '20]

Given N identical Markov decision processes, make them all reach ✔ with probability 1.

$S \leftarrow \mathbb{N}^Q$ → configurations

$C \leftarrow \mathbb{N}^Q \times \Sigma$ → commits

while not fixpoint **do**

  if $\exists (s,a) \in C,\ s \xrightarrow{a} s',\ s' \notin S$ **then**

    $C \leftarrow C \setminus \{(s,a)\}\uparrow$

  if $\exists s \in S$, no path from s to F in C **then**

    $S \leftarrow S \setminus \{s\}\uparrow,\ C \leftarrow C \cap S \times \Sigma$

return $I \subseteq S$

Terminates by well quasi-order argument

Easy to check

Unclear

The remaining problem

$$C = ((\omega, 5, \omega, 8), a)\downarrow \cup ((4, \omega, \omega, 4), a)\downarrow \cup ((9, 6, 1, 4), b)\downarrow$$

Is there a path in C from every configuration in $(\omega, 5, \omega, 8)\downarrow$to $(0, 0, 0, \omega)\downarrow$?

The remaining problem

$$C = ((\omega, 5, \omega, 8), a)\downarrow \cup ((4, \omega, \omega, 4), a)\downarrow \cup ((9, 6, 1, 4), b)\downarrow$$

Is there a path in C from every configuration in $(\omega, 0, 0, 0)\downarrow$ to $(0, 0, 0, \omega)\downarrow$?

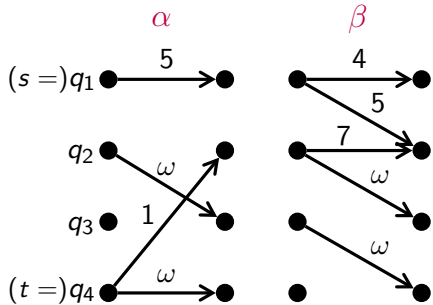
Put constraints on transitions instead of states!

Sequential Flow Problem

A *tile* is a function $Q \times Q \to \mathbb{N} \cup \{\omega\}$
describing *capacities*.

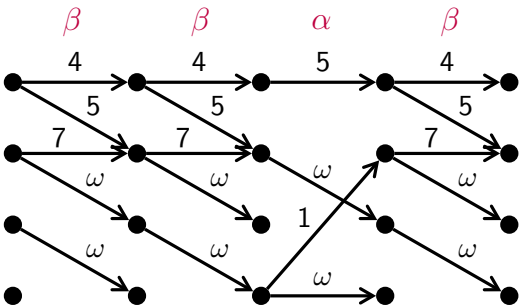
Sequential Flow Problem

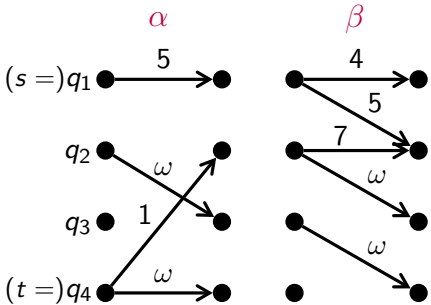
A *tile* is a function $Q \times Q \to \mathbb{N} \cup \{\omega\}$ describing *capacities*.



Sequential Flow Problem

A *tile* is a function $Q \times Q \to \mathbb{N} \cup \{\omega\}$ describing *capacities*.

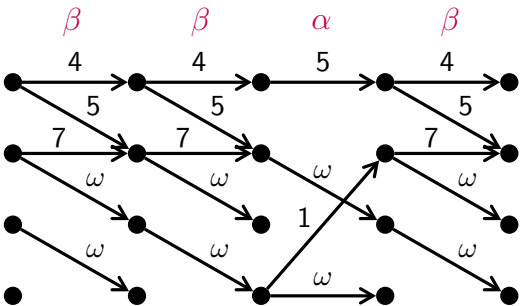


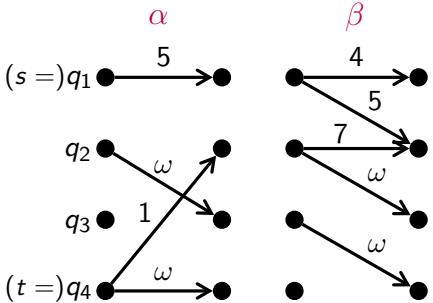


Sequential Flow Problem

A *tile* is a function $Q \times Q \to \mathbb{N} \cup \{\omega\}$ describing *capacities*.

$\texttt{MaxFlow} : \textbf{Tiles}^* \to \mathbb{N} \cup \{\omega\}$

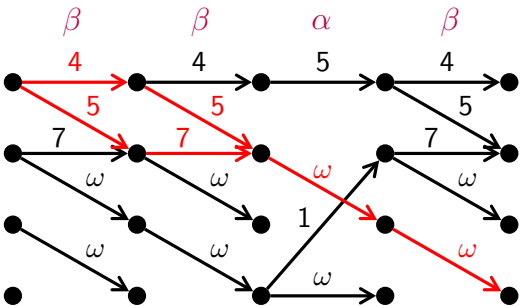


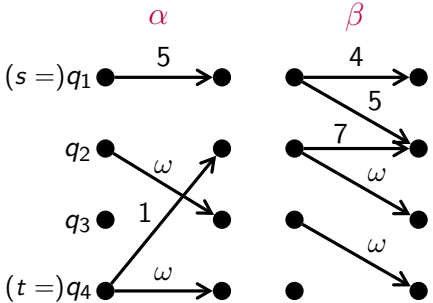


Sequential Flow Problem

A *tile* is a function $Q \times Q \to \mathbb{N} \cup \{\omega\}$ describing *capacities*.

$\texttt{MaxFlow} : \textbf{Tiles}^* \to \mathbb{N} \cup \{\omega\}$

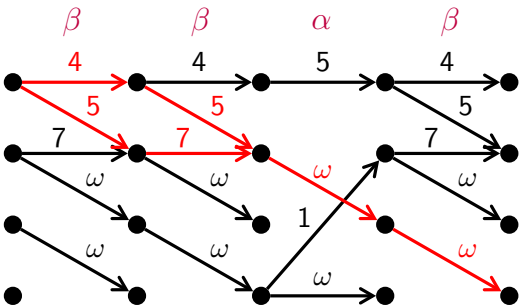


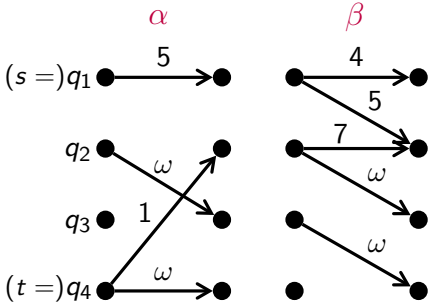


Sequential Flow Problem

A *tile* is a function $Q \times Q \to \mathbb{N} \cup \{\omega\}$ describing *capacities*.

$\texttt{MaxFlow} : \textbf{Tiles}^* \to \mathbb{N} \cup \{\omega\}$





Problem

Input: *A set of tiles* **Tiles**
Output: *Is* $\{\texttt{MaxFlow}(w) \mid w \in \textbf{Tiles}^*\}$ *unbounded?*

Sequential Flow Problem (SFP)

[Blumensath Colcombet Parys '16]

Satisfiability in some extension of MSO reduces to SFP

[Blumensath Colcombet Parys '16]

SFP is decidable.

Sequential Flow Problem (SFP)

[Blumensath Colcombet Parys '16]

Satisfiability in some extension of MSO reduces to SFP

[Colcombet Fijalkow Ohlmann '20]

The Randomised Population Control Problem reduces to SFP.

[Blumensath Colcombet Parys '16]

SFP is decidable.

[Colcombet Fijalkow Ohlmann '20]

SFP is PSPACE-hard and in EXPSPACE.

Sequential Flow Problem (SFP)

[Blumensath Colcombet Parys '16]

Satisfiability in some extension of MSO reduces to SFP

[Blumensath Colcombet Parys '16]

SFP is decidable.

[Colcombet Fijalkow Ohlmann '20]

The Randomised Population Control Problem reduces to SFP.

[Colcombet Fijalkow Ohlmann '20]

SFP is PSPACE-hard and in EXPSPACE.

[Gimbert Mascle Totzke '25]

The Randomised Population Control Problem reduces to SFP in exponential time.

[Gimbert Mascle Totzke '25]

SFP is PSPACE-complete.

Sequential Flow Problem (SFP)

[Blumensath Colcombet Parys '16]

Satisfiability in some extension of MSO reduces to SFP

[Blumensath Colcombet Parys '16]

SFP is decidable.

[Colcombet Fijalkow Ohlmann '20]

The Randomised Population Control Problem reduces to SFP.

[Colcombet Fijalkow Ohlmann '20]

SFP is PSPACE-hard and in EXPSPACE.

[Gimbert Mascle Totzke '25]

The Randomised Population Control Problem reduces to SFP in exponential time.

+

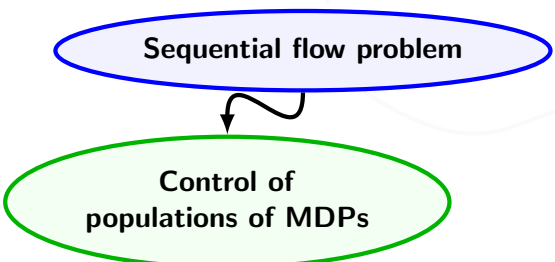
[Gimbert Mascle Totzke '25]

SFP is PSPACE-complete.

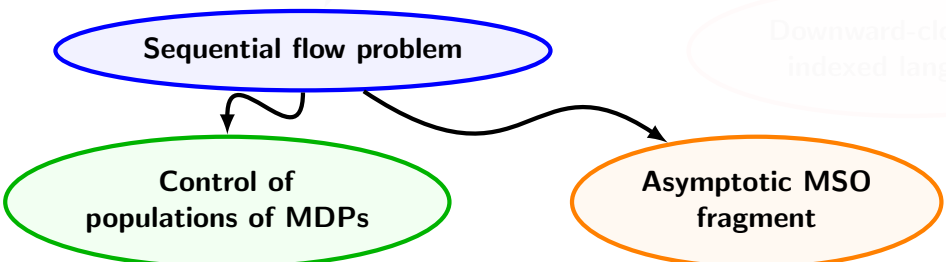
Theorem ([GMT '25])

The Randomised Population Control Problem is EXPTIME-complete.

Summary



Summary


Sequential flow problem
Control of
populations of MDPs
Asymptotic MSO
fragment

Sequential Flow Problem

Problem

Input: *A set of tiles* **Tiles**
Output: *Is* $\{\texttt{MaxFlow}(w) \mid w \in \textbf{Tiles}^*\}$ *unbounded?*

Sequential Flow Problem

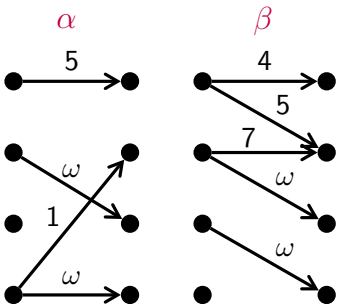
Problem

Input: *A set of tiles* **Tiles**
Output: *Is* $\{\texttt{MaxFlow}(w) \mid w \in \textbf{Tiles}^*\}$ *unbounded?*

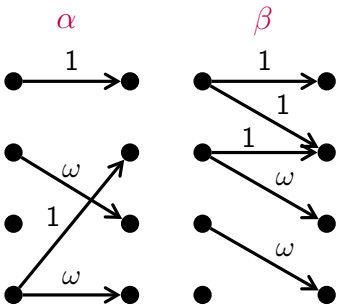
Problem

Input: *A set of tiles* **Tiles**
Output: *Compute* $\sup\{\texttt{MaxFlow}(w) \mid w \in \textbf{Tiles}^*\}$.

Abstraction

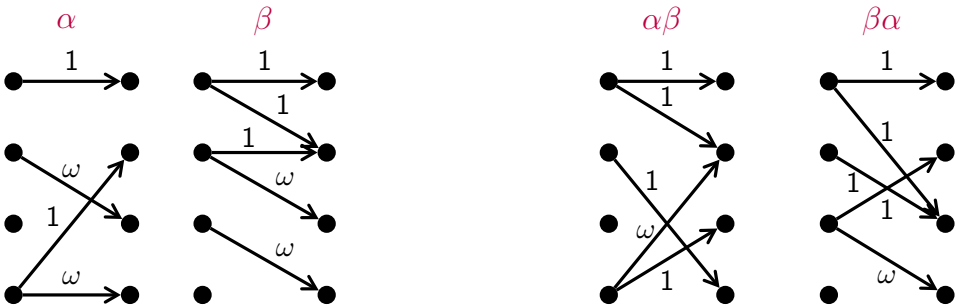


Abstraction



Morphism $\varphi : \textbf{Tiles}^* \to \{0, 1, \omega\}^{Q \times Q}$ with max-min product.

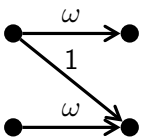
Abstraction



Morphism $\varphi : \mathbf{Tiles}^* \to \{0, 1, \omega\}^{Q \times Q}$ with max-min product.

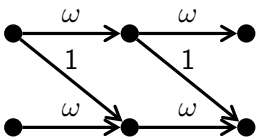
Iteration (Inspired by work of Simon, Hashigushi, Leung)

α

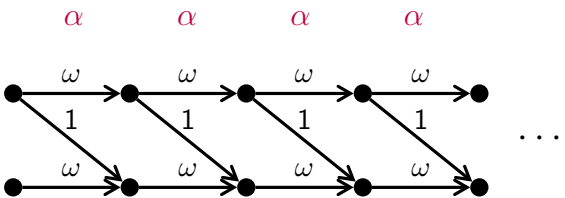


Iteration (Inspired by work of Simon, Hashigushi, Leung)

α α

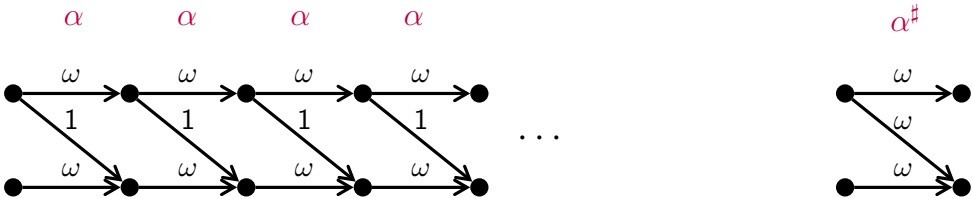


Iteration (Inspired by work of Simon, Hashigushi, Leung)



$\alpha^N = \alpha$ (α is *idempotent*) but α^N has flow N.

Iteration (Inspired by work of Simon, Hashigushi, Leung)

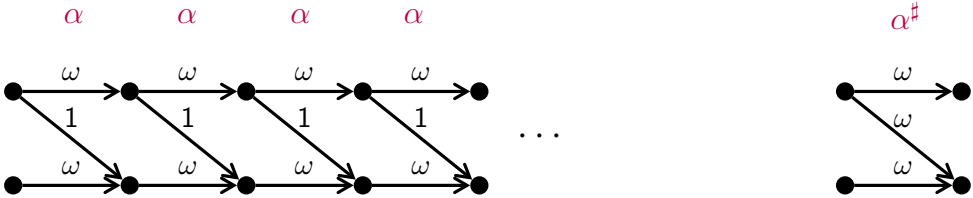


$\alpha^N = \alpha$ (α is *idempotent*) but α^N has flow N.

Define a new operator $\sharp$ on idempotents:

$$\alpha^\sharp(s,t) = \begin{cases} \omega \text{ if } \exists s_0, t_0, s \xrightarrow{\omega} s_0 \xrightarrow{1} t_0 \xrightarrow{\omega} t \text{ in } \alpha \\ \alpha(s,t) \text{ otherwise} \end{cases}$$

Iteration (Inspired by work of Simon, Hashigushi, Leung)



$\alpha^N = \alpha$ (α is *idempotent*) but α^N has flow N.

Define a new operator $\sharp$ on idempotents:

$$\alpha^\sharp(s,t) = \begin{cases} \omega \text{ if } \exists s_0, t_0, s \xrightarrow{\omega} s_0 \xrightarrow{1} t_0 \xrightarrow{\omega} t \text{ in } \alpha \\ \alpha(s,t) \text{ otherwise} \end{cases}$$

If $\alpha^\sharp(s,t) = \omega$ then we have unbounded flows between s and t.

Abstraction

$\mathcal{F}_\sharp$ the closure of $\varphi(\textbf{Tiles})$ under product and $\sharp$.

Lemma

$\mathcal{F}_\sharp$ contains some α with $\alpha(s,t) = \omega$ if and only if there are unbounded flows between s and t.

Abstraction

$\mathcal{F}_\sharp$ the closure of $\varphi(\textbf{Tiles})$ under product and $\sharp$.

Lemma

$\mathcal{F}_\sharp$ contains some α with $\alpha(s,t) = \omega$ if and only if there are unbounded flows between s and t.

Checkable in PSPACE!

Abstraction

$\mathcal{F}_\sharp$ the closure of $\varphi(\textbf{Tiles})$ under product and $\sharp$.

Lemma

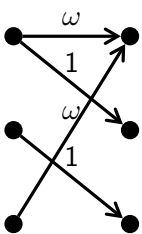
$\mathcal{F}_\sharp$ contains some α with $\alpha(s,t) = \omega$ if and only if there are unbounded flows between s and t.

Checkable in PSPACE!

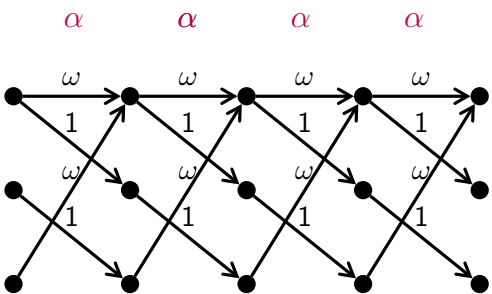
`How large can a bounded flow be?`

Iteration dichotomy

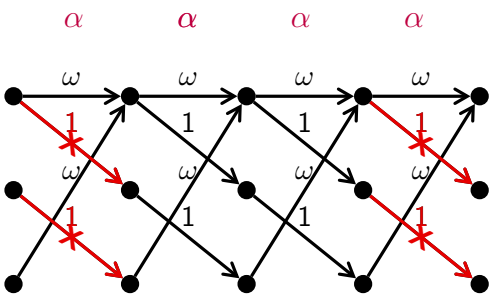
α



Iteration dichotomy



Iteration dichotomy



For all idempotent α and $s, t \in Q$,

- either $\alpha^\sharp(s,t) = \omega$
- or there is a cut between s and t in the first and last α.

Short-ends factorisation theorem (based on work by [Simon '90], [Colcombet '11])

Σ finite alphabet, $(\boldsymbol{M}, \cdot)$ finite monoid, $\varphi : \Sigma^* \to \mathbf{M}$ morphism.

Short-ends factorisation theorem (based on work by [Simon '90], [Colcombet '11])

Σ finite alphabet, $(\boldsymbol{M}, \cdot)$ finite monoid, $\varphi : \Sigma^* \to \mathbf{M}$ morphism.
`Factorisation tree:` Node labels in $\Sigma^* \times \boldsymbol{M}$.

3 types of nodes:

Leaves

$(a, \varphi(a))$

Product nodes

$(uv, x \cdot y)$

(u, x) (v, y)

Idempotent nodes

$(u_1 u_2 \cdots u_n, e)$

(u_1, e) (u_n, e)

$\varphi(u_i) = e$ for all i.

Short-ends factorisation theorem (based on work by [Simon '90], [Colcombet '11])

Σ finite alphabet, $(\boldsymbol{M}, \cdot)$ finite monoid, $\varphi : \Sigma^* \to \mathbf{M}$ morphism.
`Factorisation tree:` Node labels in $\Sigma^* \times \boldsymbol{M}$.

3 types of nodes:

Leaves

$(a, \varphi(a))$

Product nodes

$(uv, x \cdot y)$

(u, x) (v, y)

Idempotent nodes

$(u_1 u_2 \cdots u_n, e)$

(u_1, e) (u_n, e)

$\varphi(u_i) = e$ for all i.

`Example on the board` ⟶

Ramsey bounds

$R_{\mathbf{M}}(k)$ = minimal n such that every word w of length n has a factor $u_1 \dots u_k$ with $\varphi(u_1) = \cdots = \varphi(u_k) = e$ an idempotent of $\mathbf{M}$.

Ramsey bounds

$R_{\mathbf{M}}(k)$ = minimal n such that every word w of length n has a factor $u_1 \dots u_k$ with $\varphi(u_1) = \cdots = \varphi(u_k) = e$ an idempotent of $\mathbf{M}$.

Theorem

For all $w \in \Sigma^*$, there is a factorization tree of height $poly(log(\mathbf{M}), log(R_{\mathbf{M}}(3)))$.

Corollary

In a transition monoid of dimension n, every word has a factorization tree of height $poly(n)$.

Based on bounds by [Jecker '21].

Exponential bound

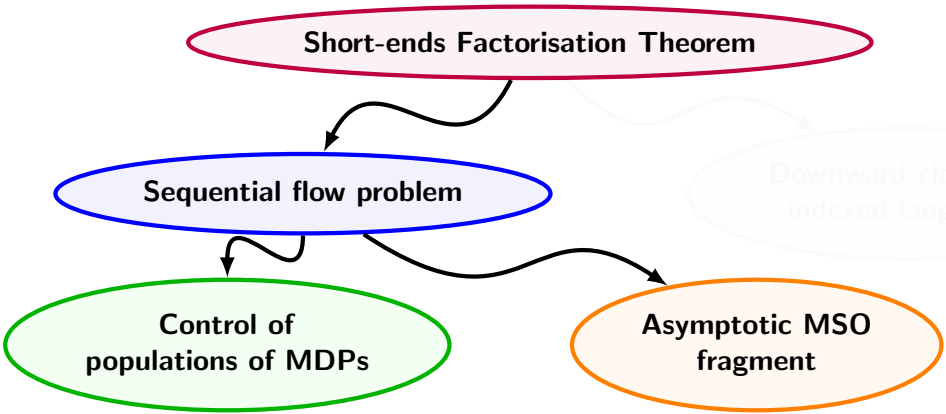
We can show:

- Every word has a factorisation of polynomial height in the flow monoid.
- If there is a bounded cut between s and t in all w, then there is one that lives within the factorisation.

Theorem

If the flow between s and t is bounded then it is at most exponential in $|Q|$.

Summary


Short-ends Factorisation Theorem
Sequential flow problem
Control of
populations of MDPs
Asymptotic MSO
fragment

Another application: Indexed grammars

Indexed grammar= Context-free grammar where each non-terminal carries a stack.
N set of *non-terminals*, T set of *terminals*, Γ set of *stack symbols*.

Another application: Indexed grammars

Indexed grammar= Context-free grammar where each non-terminal carries a stack.
N set of *non-terminals*, T set of *terminals*, Γ set of *stack symbols*.

$$A \to w \in T^*$$
$$A \to BC$$
$$A \xrightarrow{push(\gamma)} B\gamma$$
$$A\gamma \xrightarrow{pop(\gamma)} B$$

Indexed grammars : Examples

$$S \xrightarrow{push(\gamma_\perp)} S\gamma_\perp$$

$$S \xrightarrow{push(\gamma_a)} S\gamma_a$$

$$S \xrightarrow{push(\gamma_b)} S\gamma_b$$

$$S \to WW$$

$$W\gamma_a \xrightarrow{pop(\gamma_a)} Wa$$

$$W\gamma_b \xrightarrow{pop(\gamma_b)} Wb$$

$$W\gamma_\perp \xrightarrow{pop(\gamma_\perp)} \varepsilon$$

Indexed grammars : Examples

$$S \xrightarrow{push(\gamma_\perp)} S\gamma_\perp$$

$$S \xrightarrow{push(\gamma_a)} S\gamma_a$$

$$S \xrightarrow{push(\gamma_b)} S\gamma_b$$

$$S \to WW$$

$$W\gamma_a \xrightarrow{pop(\gamma_a)} Wa$$

$$W\gamma_b \xrightarrow{pop(\gamma_b)} Wb$$

$$W\gamma_\perp \xrightarrow{pop(\gamma_\perp)} \varepsilon$$

S

Indexed grammars : Examples

$$S \xrightarrow{push(\gamma_\perp)} S\gamma_\perp$$

$$S \xrightarrow{push(\gamma_a)} S\gamma_a$$

$$S \xrightarrow{push(\gamma_b)} S\gamma_b$$

$$S \to WW$$

$$W\gamma_a \xrightarrow{pop(\gamma_a)} Wa$$

$$W\gamma_b \xrightarrow{pop(\gamma_b)} Wb$$

$$W\gamma_\perp \xrightarrow{pop(\gamma_\perp)} \varepsilon$$

$$\begin{array}{c} S \\ | \\ S[\gamma_\perp] \\ | \\ S[\gamma_a\gamma_\perp] \\ | \\ S[\gamma_b\gamma_a\gamma_\perp] \end{array}$$

Indexed grammars : Examples

$$S \xrightarrow{push(\gamma_\perp)} S\gamma_\perp$$

$$S \xrightarrow{push(\gamma_a)} S\gamma_a$$

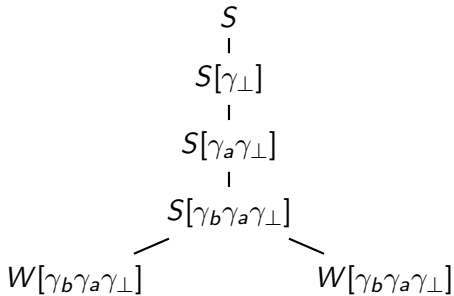
$$S \xrightarrow{push(\gamma_b)} S\gamma_b$$

$$S \to WW$$

$$W\gamma_a \xrightarrow{pop(\gamma_a)} Wa$$

$$W\gamma_b \xrightarrow{pop(\gamma_b)} Wb$$

$$W\gamma_\perp \xrightarrow{pop(\gamma_\perp)} \varepsilon$$



Indexed grammars : Examples

$$S \xrightarrow{push(\gamma_\perp)} S\gamma_\perp$$

$$S \xrightarrow{push(\gamma_a)} S\gamma_a$$

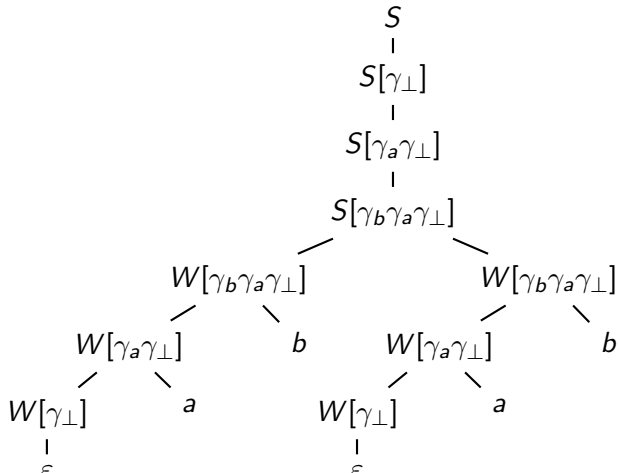
$$S \xrightarrow{push(\gamma_b)} S\gamma_b$$

$$S \to WW$$

$$W\gamma_a \xrightarrow{pop(\gamma_a)} Wa$$

$$W\gamma_b \xrightarrow{pop(\gamma_b)} Wb$$

$$W\gamma_\perp \xrightarrow{pop(\gamma_\perp)} \varepsilon$$



Indexed grammars : Examples

$$S \xrightarrow{push(\gamma_\perp)} S\gamma_\perp$$

$$S \xrightarrow{push(\gamma_a)} S\gamma_a$$

$$S \xrightarrow{push(\gamma_b)} S\gamma_b$$

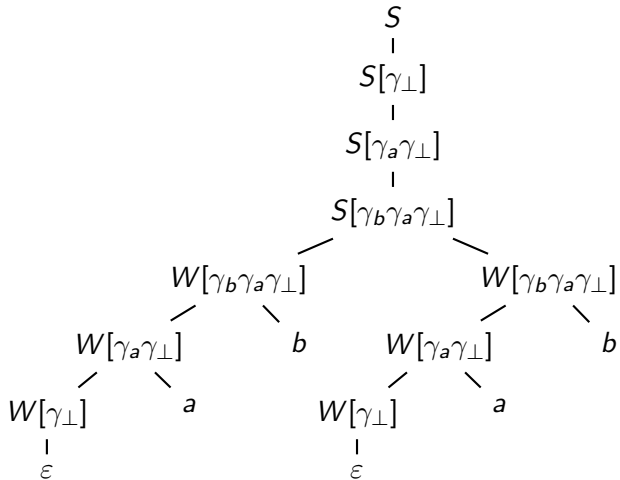
$$S \to WW$$

$$W\gamma_a \xrightarrow{pop(\gamma_a)} Wa$$

$$W\gamma_b \xrightarrow{pop(\gamma_b)} Wb$$

$$W\gamma_\perp \xrightarrow{pop(\gamma_\perp)} \varepsilon$$

$$\{ww \mid w \in \{a, b\}^*\}$$



Subword-closures

Subword

w is a **subword** of w' ($w \preceq w'$) if we can remove letters of w' to get w.

Ex: bab is a subword of $a\underline{b}b\underline{a}\underline{b}a$.

Subword-closures

Subword

w is a **subword** of w' ($w \preceq w'$) if we can remove letters of w' to get w.

Ex: bab is a subword of $a\underline{b}b\underline{a}b\underline{a}$.

Theorem (Corollary of [Higman '52])

The subword-closure of a language $L \subseteq \Sigma^*$ is always a regular language.

Subword-closures

Subword

w is a **subword** of w' ($w \preceq w'$) if we can remove letters of w' to get w.

Ex: *bab* is a subword of $a\underline{b}b\underline{a}b\underline{a}$.

Theorem (Corollary of [Higman '52])

The subword-closure of a language $L \subseteq \Sigma^*$ is always a regular lan

Not effective!

Subword-closures

Subword

w is a **subword** of w' ($w \preceq w'$) if we can remove letters of w' to get w.

Ex: *bab* is a subword of $a\underline{b}b\underline{a}b\underline{a}$.

Theorem (Corollary of [Higman '52])

The subword-closure of a language $L \subseteq \Sigma^*$ is always a regular lan

Not effective!

Given a class of languages $\mathcal{C}$, is the subword-closure *computable* for these languages? How large is the resulting automaton?

Subword-closures

Subword

w is a **subword** of w' ($w \preceq w'$) if we can remove letters of w' to get w.

Ex: *bab* is a subword of *abbaba*.

Theorem (Corollary of [Higman '52])

The subword-closure of a language $L \subseteq \Sigma^*$ is always a regular lan

Not effective!

Given a class of languages $\mathcal{C}$, is the subword-closure *computable* for these languages? How large is the resulting automaton?

- Verification of thread pools with bounded context switching
- Separation by piecewise-testable languages
- Lossy channel machines

Subword-closures

[Courcelles '91]

The subword-closure of a context-free grammar is computable and accepted by an automaton of exponential size.

Subword-closures

[Courcelles '91]

The subword-closure of a context-free grammar is computable and accepted by an automaton of exponential size.

[Zetzsche '15]

The subword-closure of an indexed grammar is computable.

Subword-closures

[Courcelles '91]

The subword-closure of a context-free grammar is computable and accepted by an automaton of exponential size.

[Zetzsche '15]

The subword-closure of an indexed grammar is computable.

Subword-closure for indexed grammars

Given an indexed grammar $\mathcal{I}$ of size k, how large can an automaton recognising $L(\mathcal{I})\downarrow$ be?

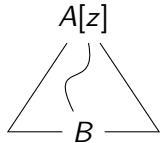
Production monoid

A stack content $z \in \Gamma^*$ is mapped to a boolean matrix $M(z) \in \{\top, \bot\}^{N \times N}$.

Production monoid

A stack content $z \in \Gamma^*$ is mapped to a boolean matrix $M(z) \in \{\top, \bot\}^{N \times N}$.

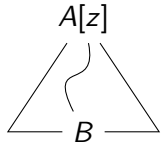
$M(z)(A, B) = \top$ if and only if we can obtain a B from $A[z]$.



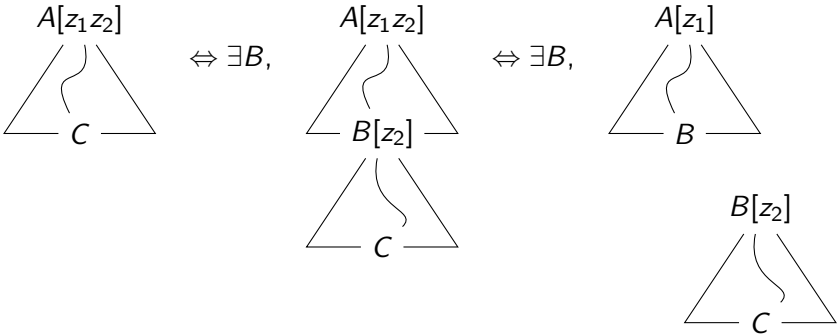
Production monoid

A stack content $z \in \Gamma^*$ is mapped to a boolean matrix $M(z) \in \{\top, \bot\}^{N \times N}$.

$M(z)(A, B) = \top$ if and only if we can obtain a B from $A[z]$.



M is a morphism: $M(z_1 z_2) = M(z_1)M(z_2)$ because



Abstraction

Replace stacks with their short-ends factorization in the production monoid!

Lemma

The short-ends factorization of γz is determined by the one of z.

Abstraction

Replace stacks with their short-ends factorization in the production monoid!

Lemma

The short-ends factorization of γz is determined by the one of z.

For each γ define $\mathtt{push}_\gamma$ the function mapping the factorization of z to the factorization of γz and $\mathtt{pop}_\gamma = \mathtt{push}_\gamma^{-1}$.

Abstraction

Define a context-free grammar with non-terminals $A[\sigma]$,

- $A \in N$
- σ short-ends factorisation

$$
\begin{aligned}
&A \to w \in T^* \\
&A \to BC \\
&A \xrightarrow{push(\gamma)} B\gamma \\
&A\gamma \xrightarrow{pop(\gamma)} B
\end{aligned}
\quad \Rightarrow \quad
\begin{aligned}
&A[\sigma] \to w \in T^* \\
&A[\sigma] \to B[\sigma]C[\sigma] \\
&A[\sigma] \to B[\mathtt{push}_\gamma(\sigma)] \\
&A[\sigma] \to B[\mathtt{pop}_\gamma(\sigma)]
\end{aligned}
$$

Theorem

This CFG has the same subword-closure as the indexed grammar.

Subword-closures

Theorem

The subword-closure of $L(\mathcal{G})$ is recognised by a context-free grammar of 2-exponential size.

Subword-closures

Theorem

The subword-closure of $L(\mathcal{G})$ is recognised by a context-free grammar of 2-exponential size.

Theorem

The subword-closure of $L(\mathcal{G})$ is recognised by an automaton of 3-exponential size, and this bound is tight.

Subword-closures

Theorem

The subword-closure of $L(\mathcal{G})$ is recognised by a context-free grammar of 2-exponential size.

Theorem

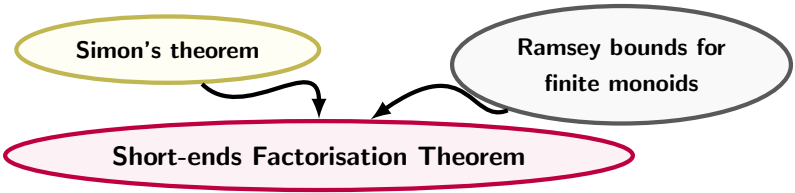
The subword-closure of $L(\mathcal{G})$ is recognised by an automaton of 3-exponential size, and this bound is tight.

Extensions to higher-order pushdown automata...

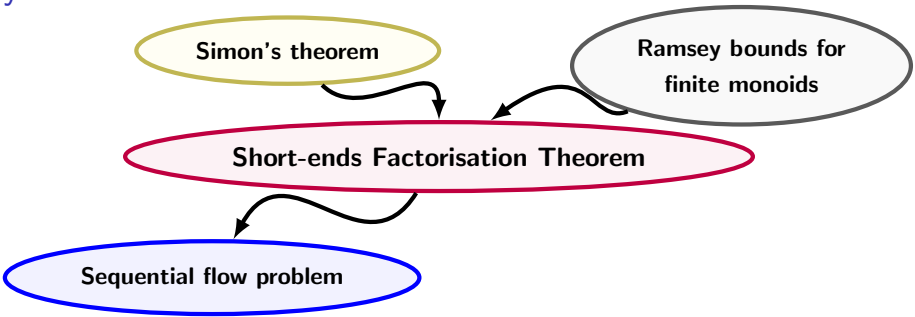
Summary

Short-ends Factorisation Theorem

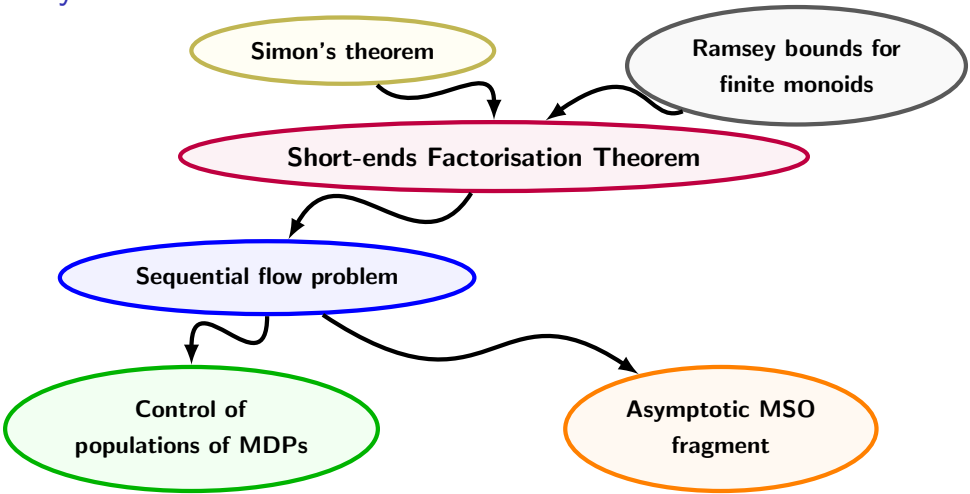
Summary


Simon's theorem
Ramsey bounds for finite monoids
Short-ends Factorisation Theorem

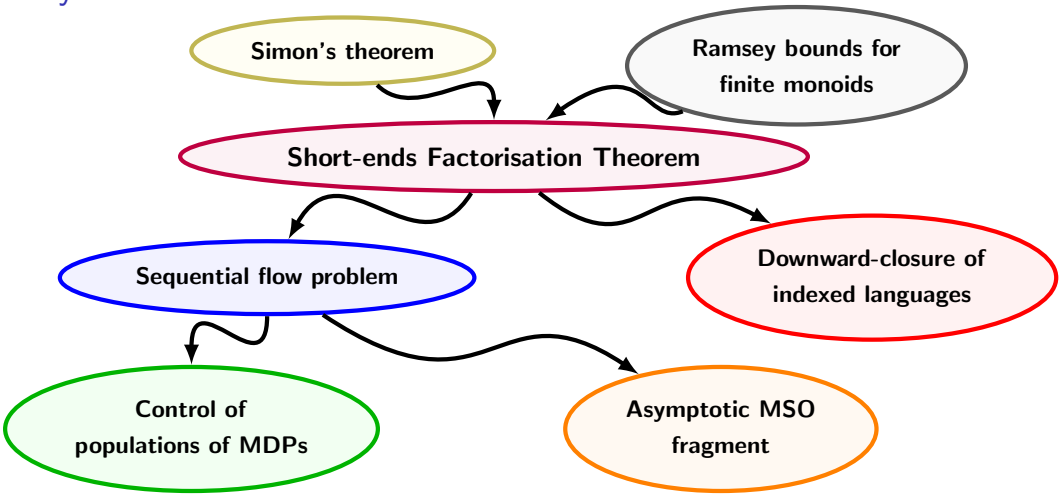
Summary


Simon's theorem
Ramsey bounds for finite monoids
Short-ends Factorisation Theorem
Sequential flow problem

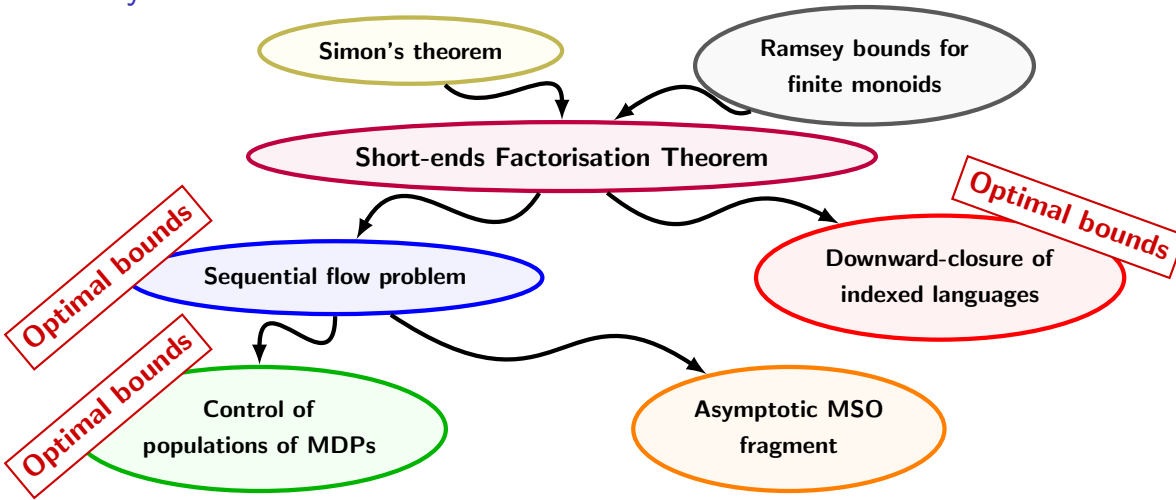
Summary


Simon's theorem
Ramsey bounds for finite monoids
Short-ends Factorisation Theorem
Sequential flow problem
Control of populations of MDPs
Asymptotic MSO fragment

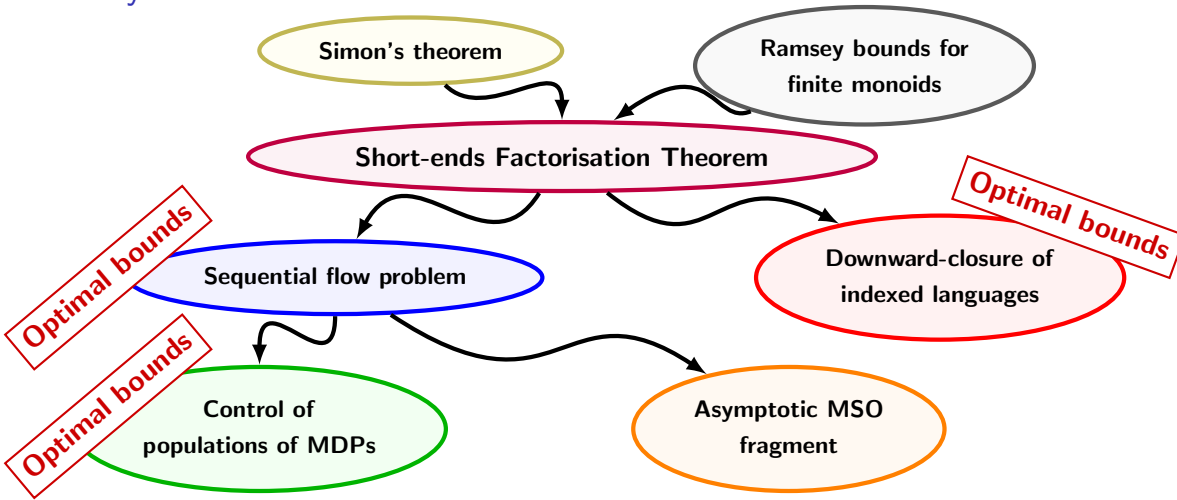
Summary


Simon's theorem
Ramsey bounds for finite monoids
Short-ends Factorisation Theorem
Sequential flow problem
Downward-closure of indexed languages
Control of populations of MDPs
Asymptotic MSO fragment

Summary


Simon's theorem
Ramsey bounds for finite monoids
Short-ends Factorisation Theorem
Optimal bounds
Optimal bounds
Sequential flow problem
Downward-closure of indexed languages
Optimal bounds
Control of populations of MDPs
Asymptotic MSO fragment

Summary



Thanks!