The limits of Nečiporuk's method and the power of programs over monoids taken from small varieties of finite monoids

Ph.D. thesis defence

Grosshans Nathan

ENS Paris-Saclay & Université de Montréal

September 25, 2018

What is computation?

First example: addition

$$\begin{array}{r} 537 \\ + \quad 71 \\ \hline \end{array}$$

What is computation?

First example: addition

$$\begin{array}{r} 537 \\ +\quad 71 \\ \hline 8 \end{array}$$

What is computation?

First example: addition

$$
\begin{array}{rr}
 & \overset{1}{5}37 \\
+ & 71 \\
\hline
 & 08
\end{array}
$$

What is computation?

First example: addition

$$\begin{array}{rr} & \overset{1}{5}37 \\ + & 71 \\ \hline & 608 \end{array}$$

What is computation?

First example: addition

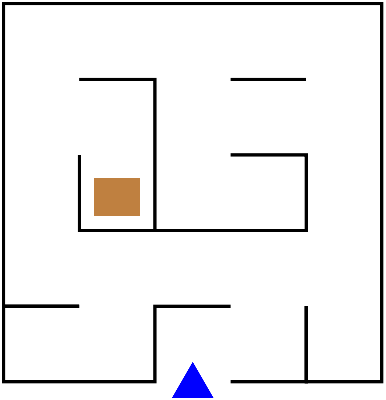
$$\begin{array}{r} \overset{1}{5}37 \\ +\quad 71 \\ \hline 608 \end{array}$$

How does one add two numbers?

One follows a certain procedure giving a succession of small operations.

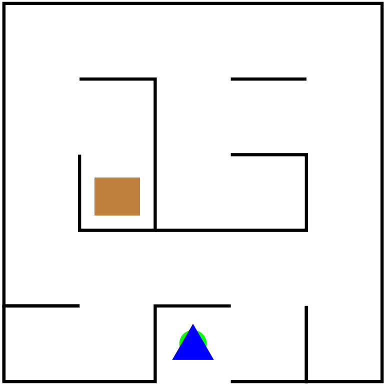
What is computation?

Second example: finding a treasure



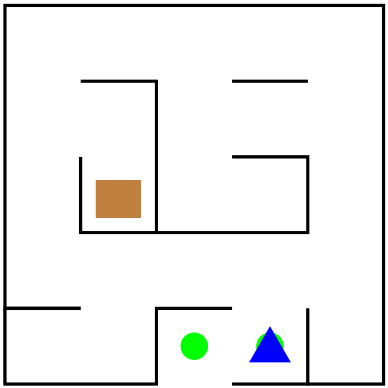
What is computation?

Second example: finding a treasure



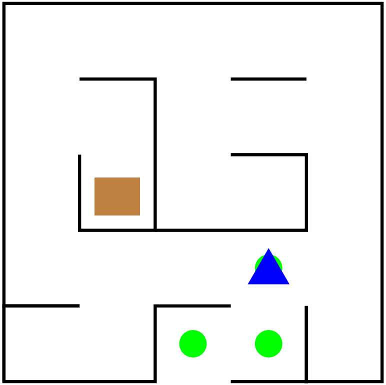
What is computation?

Second example: finding a treasure



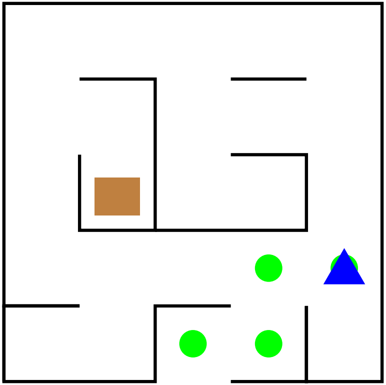
What is computation?

Second example: finding a treasure



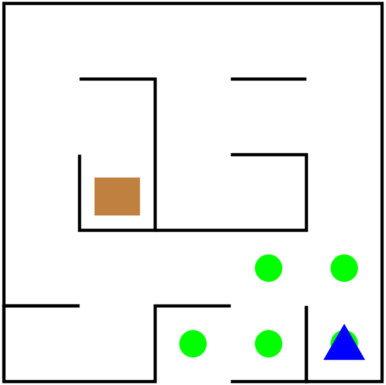
What is computation?

Second example: finding a treasure



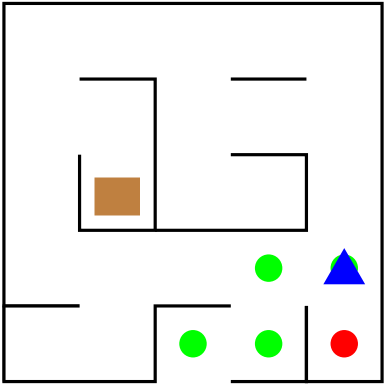
What is computation?

Second example: finding a treasure



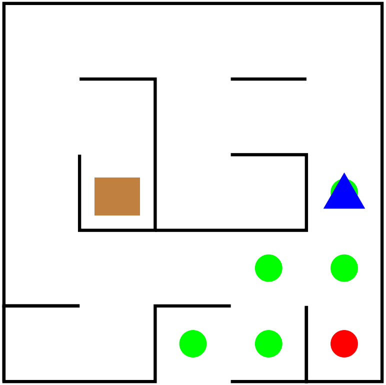
What is computation?

Second example: finding a treasure



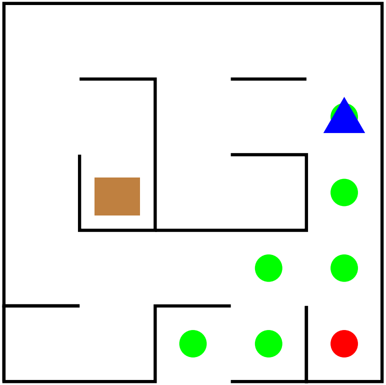
What is computation?

Second example: finding a treasure



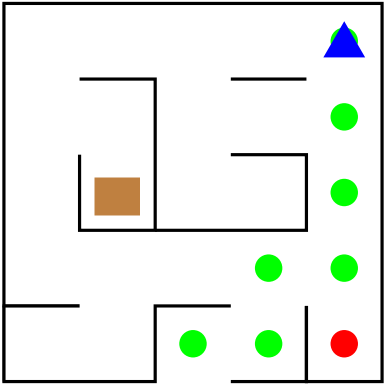
What is computation?

Second example: finding a treasure



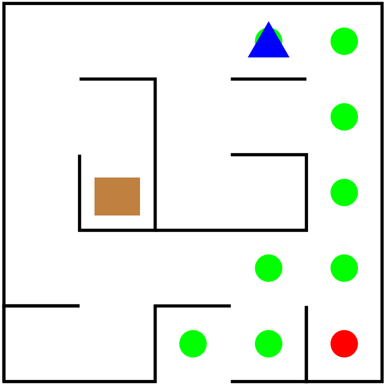
What is computation?

Second example: finding a treasure



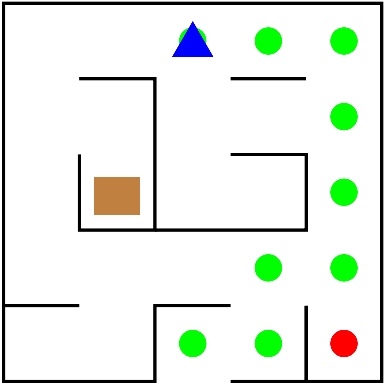
What is computation?

Second example: finding a treasure



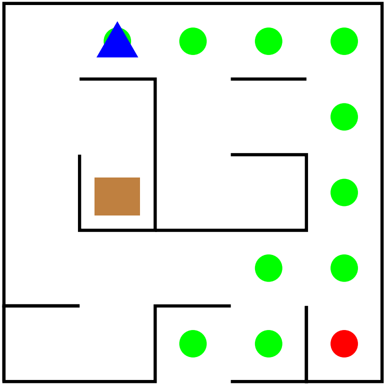
What is computation?

Second example: finding a treasure



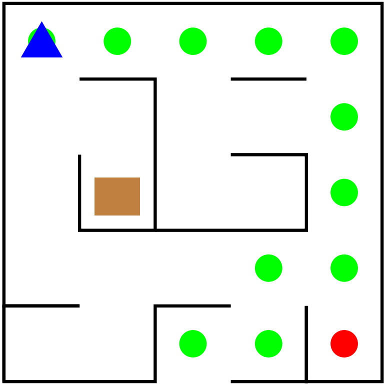
What is computation?

Second example: finding a treasure



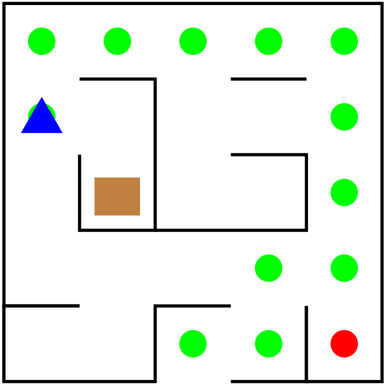
What is computation?

Second example: finding a treasure



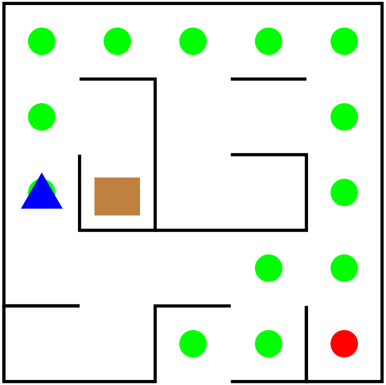
What is computation?

Second example: finding a treasure



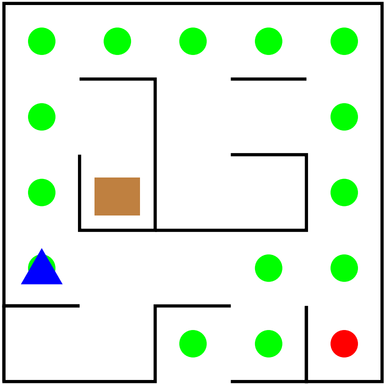
What is computation?

Second example: finding a treasure



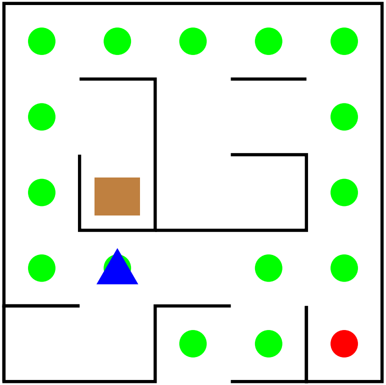
What is computation?

Second example: finding a treasure



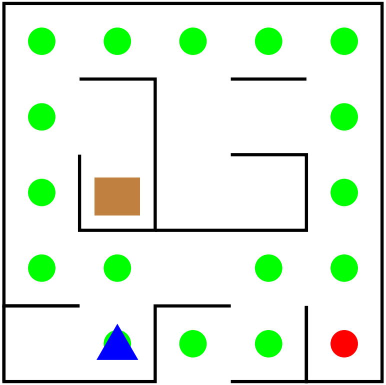
What is computation?

Second example: finding a treasure



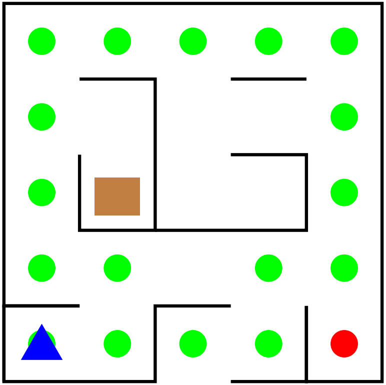
What is computation?

Second example: finding a treasure



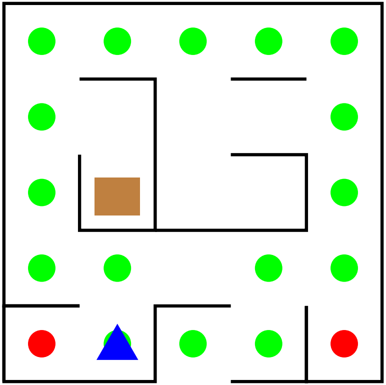
What is computation?

Second example: finding a treasure



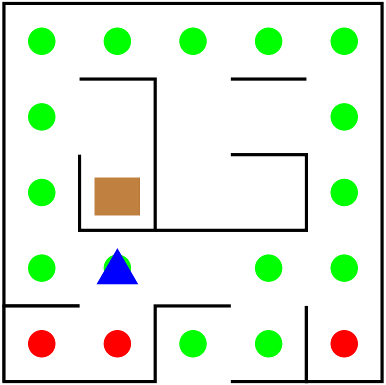
What is computation?

Second example: finding a treasure



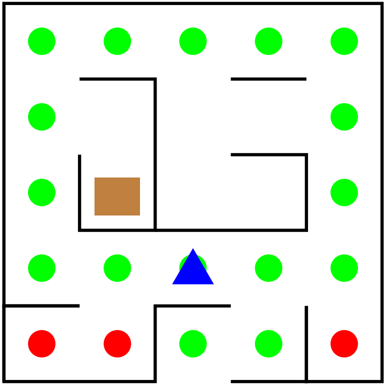
What is computation?

Second example: finding a treasure



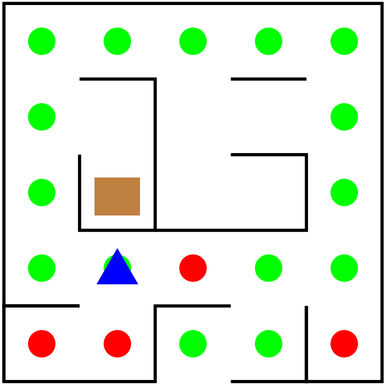
What is computation?

Second example: finding a treasure



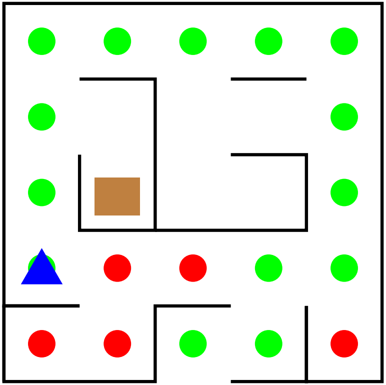
What is computation?

Second example: finding a treasure



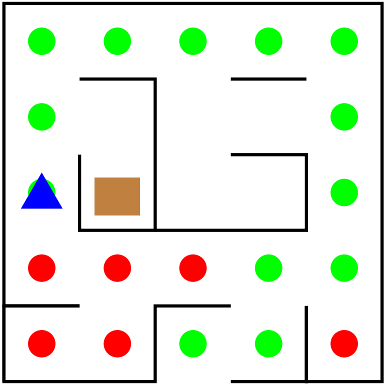
What is computation?

Second example: finding a treasure



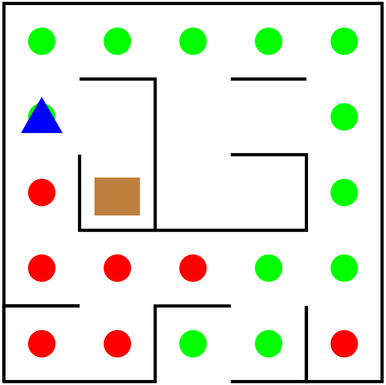
What is computation?

Second example: finding a treasure



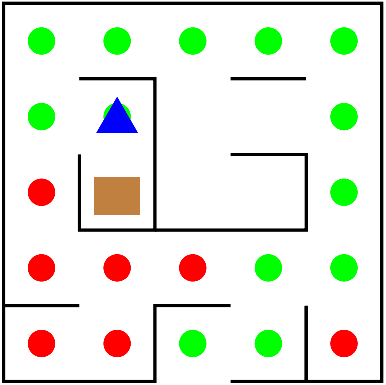
What is computation?

Second example: finding a treasure



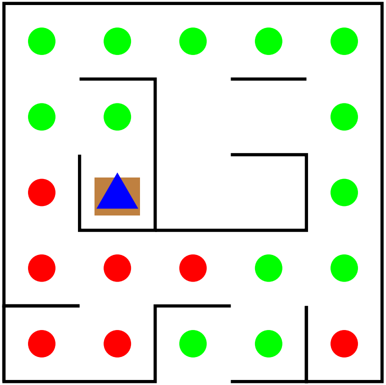
What is computation?

Second example: finding a treasure



What is computation?

Second example: finding a treasure



What is computation?

Models of computation and computability

Computation

Sequence of elementary computational steps transforming an input into an output, execution of an algorithm.

Model of computation

Class of objects implementing a certain type of algorithms.

Computability

What can be computed in a given model of computation?

How efficient can computation be?

First example: addition vs multiplication

$$\begin{array}{rr} & 537 \\ + & 71 \\ \hline \end{array}$$

How efficient can computation be?

First example: addition vs multiplication

$$\begin{array}{rr} & \overset{1}{5}37 \\ + & 71 \\ \hline & 608 \end{array}$$

How efficient can computation be?

First example: addition vs multiplication

$$\begin{array}{rr} & \overset{1}{5}37 \\ + & 71 \\ \hline & 608 \end{array}$$

$$\begin{array}{rr} & 537 \\ \times & 71 \\ \hline \end{array}$$

How efficient can computation be?

First example: addition vs multiplication

$$
\begin{array}{rr}
 & {}^{1} \\
 & 537 \\
+ & 71 \\
\hline
 & 608
\end{array}
\qquad\qquad
\begin{array}{rr}
 & 537 \\
\times & 71 \\
\hline
 & {}^{1\,1} \\
 & 537 \\
 & {}^{3\,2\,4} \\
+ & 5190 \\
\hline
 & 38127
\end{array}
$$

How efficient can computation be?

First example: addition vs multiplication

$$\begin{array}{rr} & \overset{1}{5}37 \\ + & 71 \\ \hline & 608 \end{array}$$

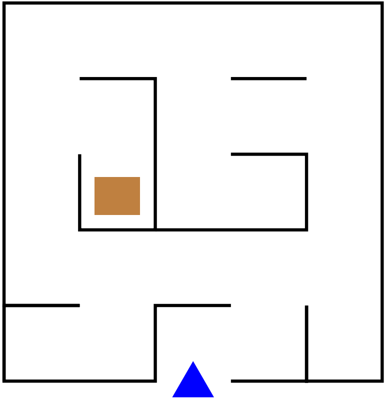
$$\begin{array}{rr} & 537 \\ \times & 71 \\ \hline & \overset{1\,1}{537} \\ + & \overset{3\,2\,4}{5190} \\ \hline & 38127 \end{array}$$

How long does it take to add/multiply two numbers?

One does count the elementary computational steps.

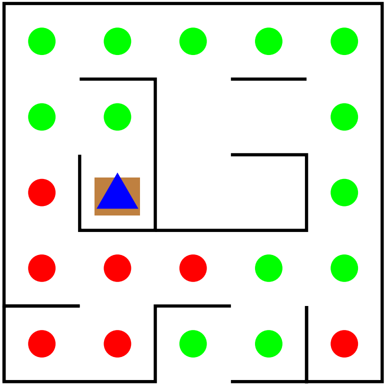
How efficient can computation be?

Second example: finding a treasure vs checking cycle-freeness



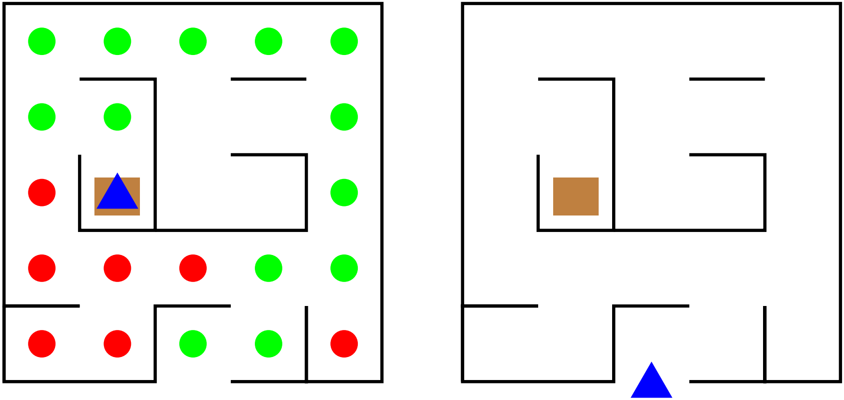
How efficient can computation be?

Second example: finding a treasure vs checking cycle-freeness



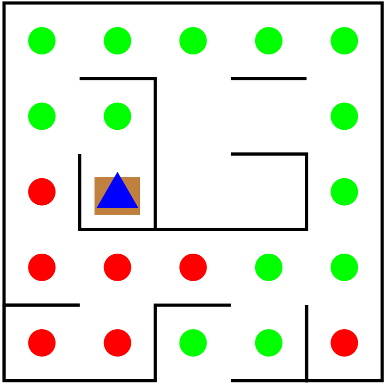
How efficient can computation be?

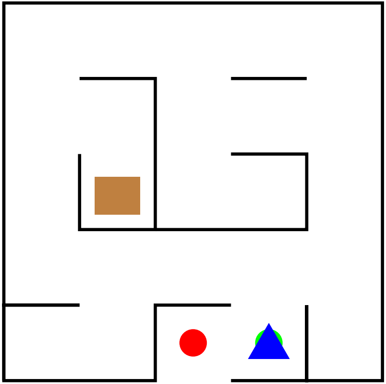
Second example: finding a treasure vs checking cycle-freeness



How efficient can computation be?

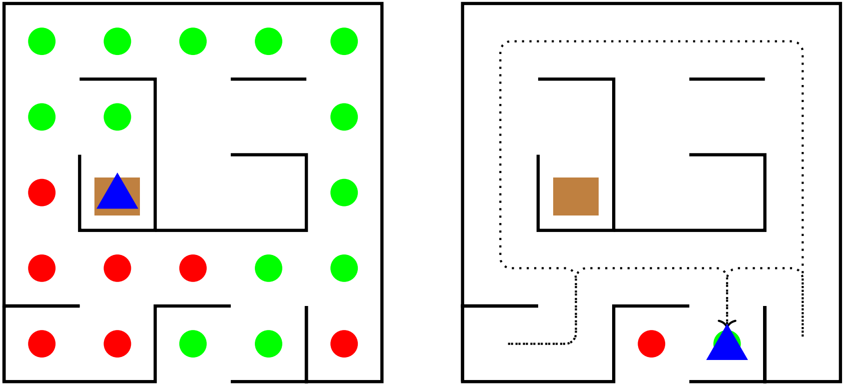
Second example: finding a treasure vs checking cycle-freeness





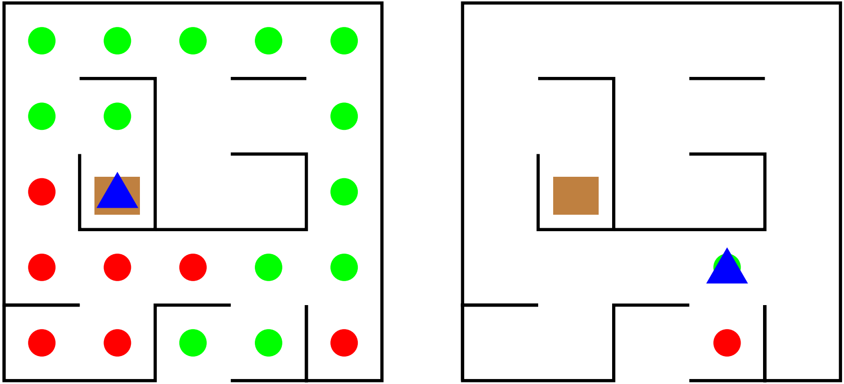
How efficient can computation be?

Second example: finding a treasure vs checking cycle-freeness



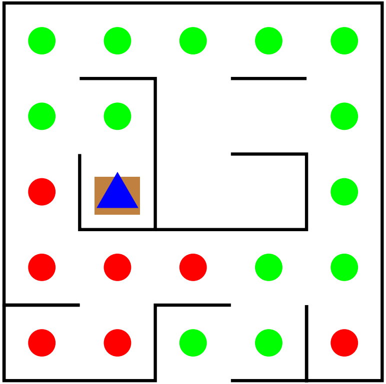
How efficient can computation be?

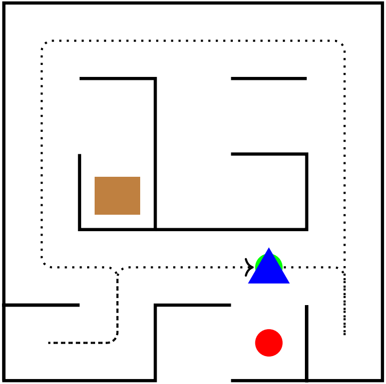
Second example: finding a treasure vs checking cycle-freeness



How efficient can computation be?

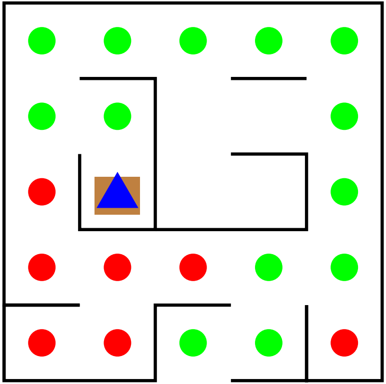
Second example: finding a treasure vs checking cycle-freeness

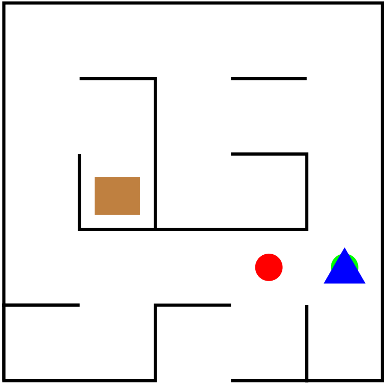




How efficient can computation be?

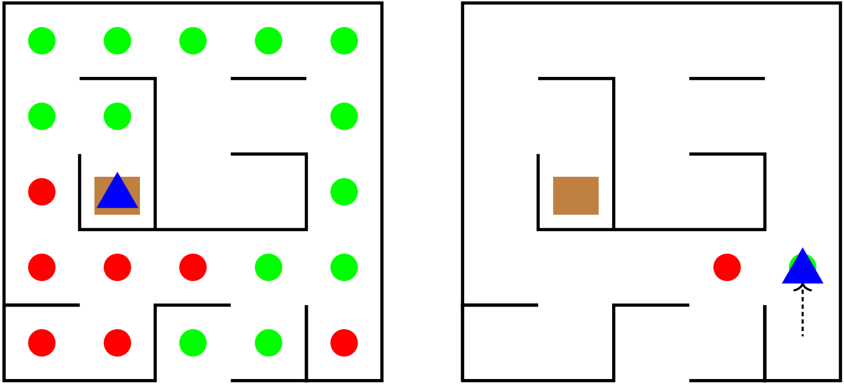
Second example: finding a treasure vs checking cycle-freeness





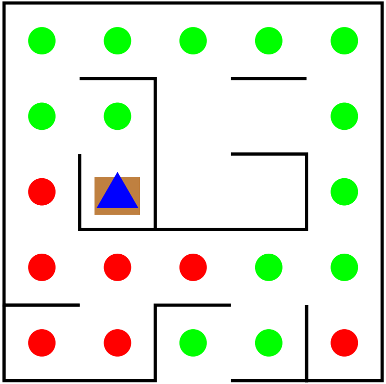
How efficient can computation be?

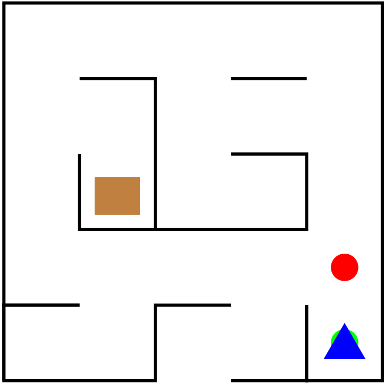
Second example: finding a treasure vs checking cycle-freeness



How efficient can computation be?

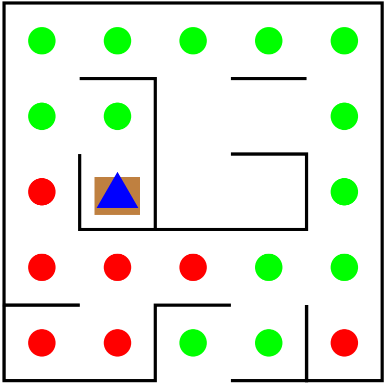
Second example: finding a treasure vs checking cycle-freeness

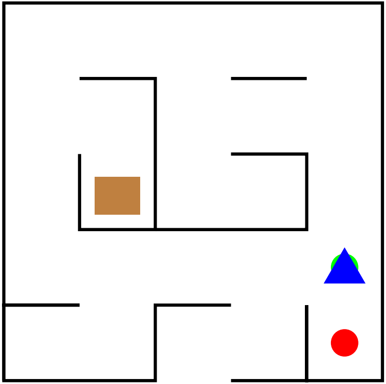




How efficient can computation be?

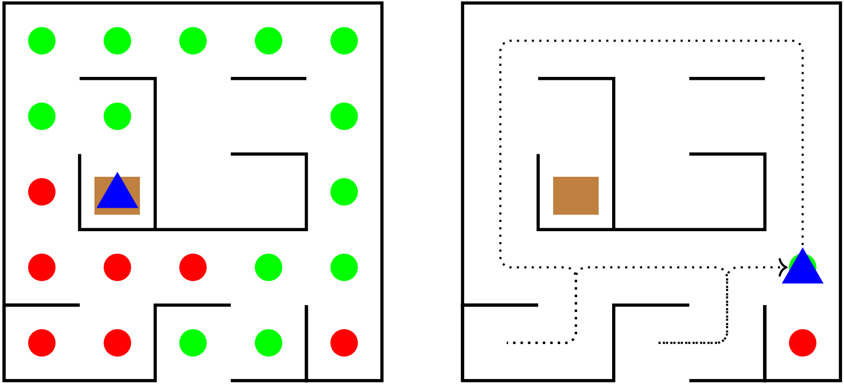
Second example: finding a treasure vs checking cycle-freeness





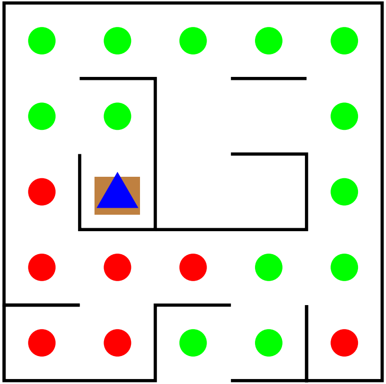
How efficient can computation be?

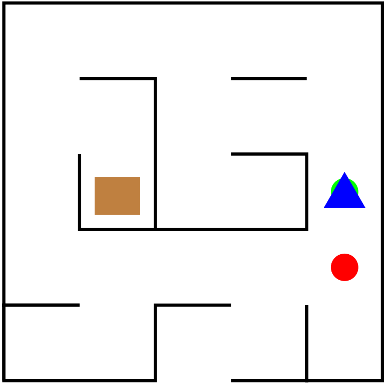
Second example: finding a treasure vs checking cycle-freeness



How efficient can computation be?

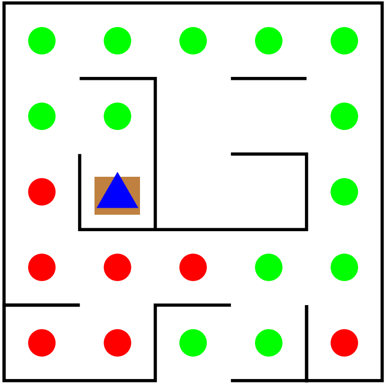
Second example: finding a treasure vs checking cycle-freeness

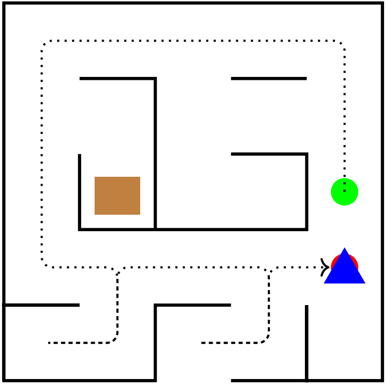




How efficient can computation be?

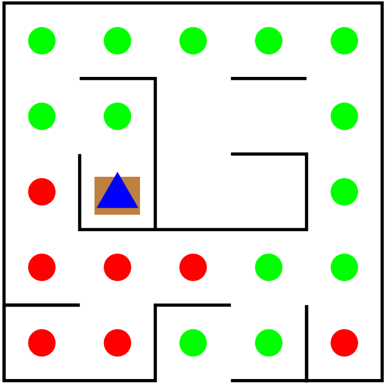
Second example: finding a treasure vs checking cycle-freeness

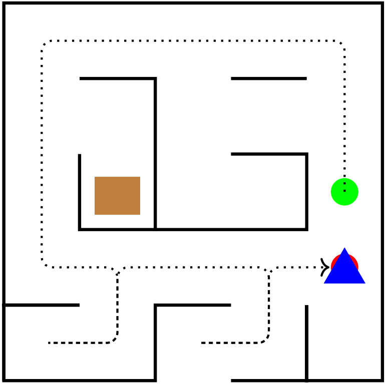




How efficient can computation be?

Second example: finding a treasure vs checking cycle-freeness





How many stones does it take to find a treasure/test cycle-freeness?

One does count the maximum number of stones used simultaneously.

How efficient can computation be?

Computational cost and complexity

Computational cost

Some quantity associated to each algorithm, measuring the level of some kind of resource consumption.

Complexity

How efficiently (in terms of some computational cost for some model of computation) can something be computed?
$\rightarrow$ Complexity measure.

This Ph.D. thesis

For several models of computation and associated complexity measures, look for lower bounds.

On the relationship between time and space

Important complexity classes

- L: languages decidable in logarithmic space on a Turing machine.
- P: languages decidable in polynomial time on a Turing machine.

Result

$\mathrm{L} \subseteq \mathrm{P}$.

Conjecture (widely believed)

$\mathrm{P} \not\subseteq \mathrm{L}$.

On the relationship between time and space

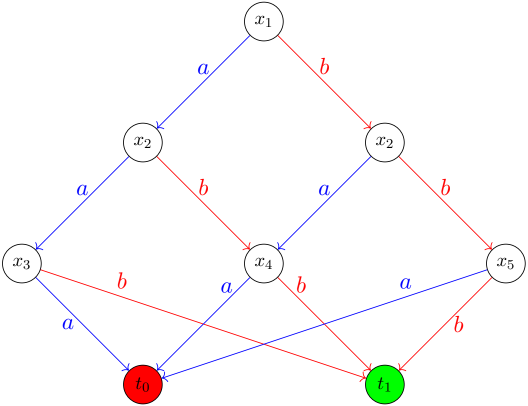
Problem

Turing machine: combinatorially hard to handle.

Possible approach

Branching programs (BPs): combinatorially simpler.

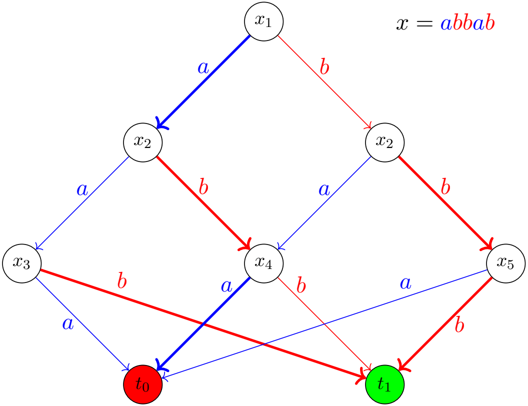
The branching program approach



- x_i: letter at position i.
- Computes function $f\colon \{a, b\}^5 \to \{0, 1\}$.
- Size of P: number of vertices.

Non-uniform model: language decided by sequence $P_0, P_1, P_2, \ldots, P_n, \ldots$ of BPs, where P_n is for inputs of length n.

The branching program approach



- x_i: letter at position i.
- Computes function $f \colon \{a, b\}^5 \to \{0, 1\}$.
- Size of P: number of vertices.

Non-uniform model: language decided by sequence $P_0, P_1, P_2, \ldots, P_n, \ldots$ of BPs, where P_n is for inputs of length n.

The branching program approach

Result (Masek, 1976)

Any language in L is decided by a sequence of polynomial size BPs.

"Simpler" task?

Separate L from P.

$\Downarrow$

Show a super-logarithmic lower bound on space for Turing machines deciding a language of P.

$\Downarrow$

Show a super-polynomial lower bound on the size of BPs deciding a language of P.

The branching program approach

Goal

Show a super-polynomial lower bound on the size of BPs deciding a language of P.

Problem

- It's difficult!
- Best lower bound: $\Theta(n^2/\log^2 n)$ (Nečiporuk, 1966).

The branching program approach

Goal

Show a super-polynomial lower bound on the size of BPs deciding a language of P.

Problem

- It's difficult!
- Best lower bound: $\Theta(n^2/\log^2 n)$ (Nečiporuk, 1966).

Restricted variants

- Various restrictions studied, for example:
  - bounded-width BPs;
  - oblivious BPs;
  - read-once BPs.
- With significant lower bounds (reported by Razborov, 1991).

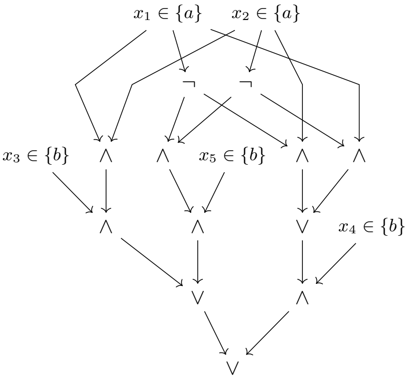
First contribution: the limits of Nečiporuk's method

The contribution

- First formulation of Nečiporuk's method for any complexity measure and Boolean function.
- For several complexity measures: upper bound on the best lower bound obtainable by Nečiporuk's method.

Complexity measure	**Best lower bound obtainable** **(and obtained)**
Non-det. BP size	$\Theta\left(\frac{n^{3/2}}{\log n}\right)$
BP size	$\Theta\left(\frac{n^2}{\log^2 n}\right)$
Formulæ size	$\Theta\left(\frac{n^2}{\log n}\right)$

Second contribution: programs over monoids



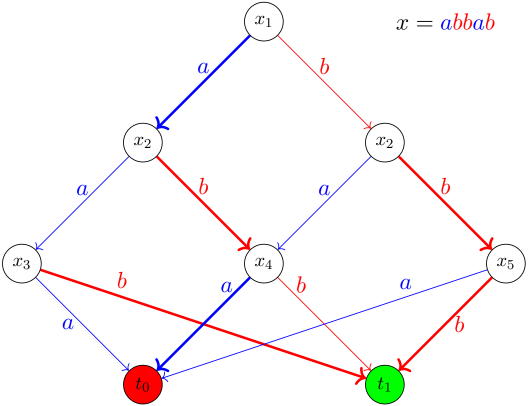
The class NC^1

- Such circuits of poly. size, log. depth and fan-in 2.
- Probably weaker than poly. size BPs.
- It's still not excluded that $\mathsf{P} \subseteq \mathsf{NC}^1$...

Programs over monoids: a restricted variant of BPs capturing NC^1.

Programs over monoids

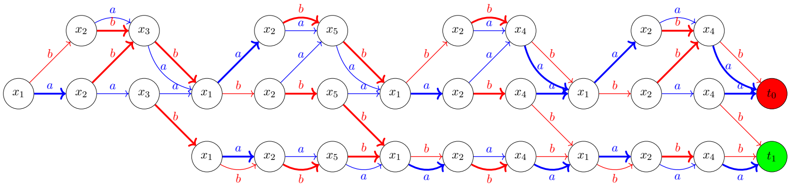
From BPs to programs over monoids



Programs over monoids

From BPs to programs over monoids

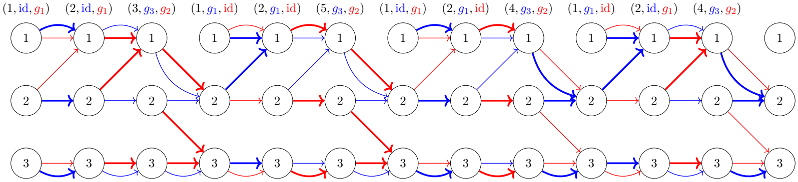
$$x = abbab$$



Programs over monoids

From BPs to programs over monoids

$$x = abbab$$



Where we consider these functions $[3] \to [3]$:

$$\mathrm{id} = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \end{pmatrix},\ g_1 = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 1 & 3 \end{pmatrix},$$

$$g_2 = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 3 \end{pmatrix},\ g_3 = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 2 & 3 \end{pmatrix}.$$

Definition

Pair $(M, \star)$ where M set and $\star \colon M \times M \to M$ such that:

- $\star$ is associative;
- $\star$ has an identity.

Remark

Each monoid $(M, \star)$ has a unique identity, denoted by $1_{(M,\star)}$.

Examples

- $(\mathbb{N}, +)$ with identity 0.
- $(\mathbb{Z}/2\,\mathbb{Z}, +)$ with identity 0.
- $(\mathbb{Z}/3\,\mathbb{Z}, \times)$ with identity 1.
- $(\Sigma^*, \cdot)$ with identity ε (Σ finite alphabet).

Program over $(M, \star)$ (finite) on Σ^n: finite sequence of instructions

$$P = (i_1, f_1)(i_2, f_2) \cdots (i_l, f_l)$$

such that $i_j \in [n]$ and $f_j \colon \Sigma \to M$. We set

$$P(w) = f_1(w_{i_1}) \star f_2(w_{i_2}) \star \cdots \star f_l(w_{i_l}) \ .$$

P recognises $L \subseteq \Sigma^n$ iff there exists $F \subseteq M$ such that

$$L = P^{-1}(F) \ .$$

$L \subseteq \Sigma^*$ is recognised by $(P_n)_{n \in \mathbb{N}}$ iff P_n (poly. length) recognises $L \cap \Sigma^n$. (Non-uniform model.)

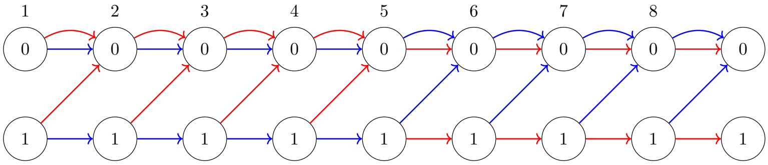
Programs over monoids

Example

Let $f_a : \{a, b\} \to \mathbb{Z}/2\,\mathbb{Z}$ such that $f_a(a) = 1$ and $f_a(b) = 0$.
Let $f_b : \{a, b\} \to \mathbb{Z}/2\,\mathbb{Z}$ such that $f_b(a) = 0$ and $f_b(b) = 1$.

A $(\mathbb{Z}/2\,\mathbb{Z}, \times)$-program P on $\{a, b\}^8$

$$P = (1, f_a)(2, f_a)(3, f_a)(4, f_a)(5, f_b)(6, f_b)(7, f_b)(8, f_b)$$



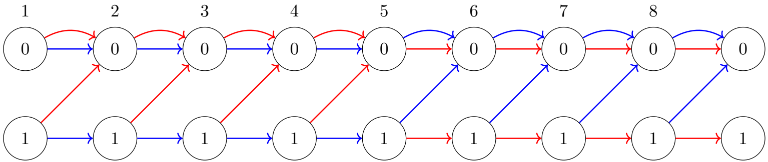
Programs over monoids

Example

Let $f_a\colon \{a,b\} \to \mathbb{Z}/2\,\mathbb{Z}$ such that $f_a(a) = 1$ and $f_a(b) = 0$.
Let $f_b\colon \{a,b\} \to \mathbb{Z}/2\,\mathbb{Z}$ such that $f_b(a) = 0$ and $f_b(b) = 1$.

A $(\mathbb{Z}/2\,\mathbb{Z}, \times)$-program P on $\{a,b\}^8$

$$P = (1, f_a)(2, f_a)(3, f_a)(4, f_a)(5, f_b)(6, f_b)(7, f_b)(8, f_b)$$



Recognises $a^4 b^4 = P^{-1}(\{1\})$.

Programs over monoids

Probable non-example

$$L = \{w \in \{a, b\}^* \mid w \text{ contains as many } a\text{'s as } b\text{'s}\}.$$

Can we recognise this language with programs over monoids?

Programs over monoids

Probable non-example

$L = \{w \in \{a, b\}^* \mid w \text{ contains as many } a\text{'s as } b\text{'s}\}.$

Can we recognise this language with programs over monoids?

- Let $f\colon \{a, b\} \to \mathbb{Z}$ such that $f(a) = 1$ and $f(b) = -1$.
  The $(\mathbb{Z}, +)$-program P on Σ^n

$$P = (1, f)(2, f) \cdots (n, f)$$

recognises $L \cap \Sigma^n = P^{-1}(\{0\})$.

Programs over monoids

Probable non-example

$$L = \{w \in \{a, b\}^* \mid w \text{ contains as many } a\text{'s as } b\text{'s}\}.$$

Can we recognise this language with programs over monoids?

- Let $f\colon \{a, b\} \to \mathbb{Z}$ such that $f(a) = 1$ and $f(b) = -1$.
  The $(\mathbb{Z}, +)$-program P on Σ^n

  $$P = (1, f)(2, f) \cdots (n, f)$$

  recognises $L \cap \Sigma^n = P^{-1}(\{0\})$.
- Exponential length: can do it with finite monoid.

Programs over monoids

Probable non-example

$$L = \{w \in \{a, b\}^* \mid w \text{ contains as many } a\text{'s as } b\text{'s}\}.$$

Can we recognise this language with programs over monoids?

- Let $f\colon \{a, b\} \to \mathbb{Z}$ such that $f(a) = 1$ and $f(b) = -1$.
  The $(\mathbb{Z}, +)$-program P on Σ^n

  $$P = (1, f)(2, f) \cdots (n, f)$$

  recognises $L \cap \Sigma^n = P^{-1}(\{0\})$.
- Exponential length: can do it with finite monoid.
- What if the monoid is finite and the length polynomial?

Morphisms and recognition

Definition

- $\varphi\colon M \to N$ is a morphism from $(M, \star)$ to $(N, \perp)$ iff
  - for all $m_1, m_2 \in M$, $\varphi(m_1) \perp \varphi(m_2) = \varphi(m_1 \star m_2)$;
  - $\varphi(1_{(M,\star)}) = 1_{(N,\perp)}$.
- Morphism $\varphi\colon \Sigma^* \to M$ from $(\Sigma^*, \cdot)$ to $(M, \star)$ recognises $L \subseteq \Sigma^*$ iff there exists $F \subseteq M$ such that $L = \varphi^{-1}(F)$. $(M, \star)$ recognises L.

Example

$b^*(ab^*ab^*)^* = \varphi^{-1}(\{0\})$ where φ from $(\{a,b\}^*, \cdot)$ to $(\mathbb{Z}/2\,\mathbb{Z}, +)$ with

$$\begin{aligned} \varphi\colon \{a,b\}^* &\to \mathbb{Z}/2\,\mathbb{Z} \\ a &\mapsto 1 \\ b &\mapsto 0\ . \end{aligned}$$

Morphisms and recognition

Fundamental theorem

Theorem

A language is recognised by a finite monoid iff it is regular.

Importance

- Basis of algebraic automata theory.
- Eilenberg's theorem: bijective correspondence between varieties of regular languages (closed under natural operations on regular languages) and varieties of finite monoids (closed under natural operations on finite monoids).
- Lots of explicit algebraic classifications of subclasses of regular languages obtained in last 50 years.

p-recognition

Definition

$(M, \star)$ p-recognises $L \subseteq \Sigma^*$ iff there exists $(P_n)_{n \in \mathbb{N}}$ sequence of poly. length $(M, \star)$-programs recognising L.

Examples

- $(M, \star)$ recognises $L \subseteq \Sigma^* \Rightarrow (M, \star)$ p-recognises L.
- $\{a^n b^n \mid n \in \mathbb{N}\}$ is p-recognised by $(\mathbb{Z}/2\,\mathbb{Z}, \times)$.
- $\{w \in \{a, b\}^* \mid w$ contains as many a's as b's$\}$ is probably not p-recognised by any finite monoid.

Definition

For any variety of finite monoids $\mathbf{V}$, $\mathcal{P}(\mathbf{V})$ is the class of languages p-recognised by monoids of $\mathbf{V}$.

Theorem (Barrington)

A language belongs to NC^1 *iff it is p-recognised by a finite monoid.*

Importance

- Shows unexpected power of programs over monoids.
- Gives a semigroup-theoretic point of view on "small" complexity classes.

Some subclasses of NC^1

- AC^0: polynomial size, constant depth circuits with $\neg$ and unbounded fan-in $\wedge$ and $\vee$ gates.
- ACC^0: polynomial size, constant depth circuits with $\neg$ and unbounded fan-in $\wedge$, $\vee$ and $\equiv$ gates.

$$\mathsf{AC}^0 \subset \mathsf{ACC}^0 \subseteq \mathsf{NC}^1$$

Theorem (Barrington-Thérien)

$\mathsf{AC}^0 = \mathcal{P}(\mathbf{A})$ — $\mathbf{A}$: *finite aperiodic monoids*

$\mathsf{ACC}^0 = \mathcal{P}(\mathbf{M}_{sol})$ — $\mathbf{M}_{sol}$: *finite solvable monoids*

$\mathsf{NC}^1 = \mathcal{P}(\mathbf{M})$ — $\mathbf{M}$: *finite monoids*

Hopes and contribution of this thesis

Hopes since late 1980s

- Prove new circuit lower bounds using techniques from algebraic automata theory.
- Give new semigroup-theoretic proofs of things like $\mathrm{MOD_m} \notin \mathsf{AC}^0 = \mathcal{P}(\mathbf{A})$ for all $m \in \mathbb{N}, m \geq 2$.
- None of this materialised yet.

Hopes and contribution of this thesis

Hopes since late 1980s

- Prove new circuit lower bounds using techniques from algebraic automata theory.
- Give new semigroup-theoretic proofs of things like $\mathrm{MOD_m} \notin \mathsf{AC}^0 = \mathcal{P}(\mathbf{A})$ for all $m \in \mathbb{N}, m \geq 2$.
- None of this materialised yet.

General goal

Better understand $\mathcal{P}(\mathbf{V})$ for $\mathbf{V} \subseteq \mathbf{A}$, knowing that understanding $\mathcal{P}(\mathbf{V}) \cap \mathcal{R}eg$ "suffices" for lower bounds.

Contribution of this thesis

- Investigate general property of $\mathcal{P}(\mathbf{V}) \cap \mathcal{R}eg$ for any $\mathbf{V}$.
- Study the cases of $\mathbf{DA}$ and $\mathbf{J}$.

Regular languages and tameness

Observation

- $\mathcal{P}(\mathbf{V}) = \mathcal{P}(\mathbf{W})$ iff $\mathcal{P}(\mathbf{V}) \cap \mathcal{R}\text{eg} = \mathcal{P}(\mathbf{W}) \cap \mathcal{R}\text{eg}$ (McKenzie-Péladeau-Thérien).
- Characterising the regular languages in $\mathcal{P}(\mathbf{V})$ is fundamental: would resolve much of the structure of NC^1.

Regular languages and tameness

Observation

- $\mathcal{P}(\mathbf{V}) = \mathcal{P}(\mathbf{W})$ iff $\mathcal{P}(\mathbf{V}) \cap \mathcal{R}\text{eg} = \mathcal{P}(\mathbf{W}) \cap \mathcal{R}\text{eg}$ (McKenzie-Péladeau-Thérien).
- Characterising the regular languages in $\mathcal{P}(\mathbf{V})$ is fundamental: would resolve much of the structure of NC^1.

Subcontribution 1 New tameness condition for $\mathbf{V}$ to "behave well" with respect to p-recognition of regular languages (does not give much more power than classical recognition over $\mathbf{V}$).

- Strengthens Péladeau's p-varieties.
- Inspired by similar results for semigroups (Péladeau-Straubing-Thérien).

Proposition

Let $\mathbf{V}$ be a tame variety of finite monoids. Then

$$\mathcal{P}(\mathbf{V}) \cap \mathcal{R}\mathrm{eg} \subseteq \mathcal{L}(\mathbf{QV}) \ .$$

When $\mathbf{V}$ is local, equality holds.

Proposition

Let $\mathbf{V}$ *be a tame variety of finite monoids. Then*

$$\mathcal{P}(\mathbf{V}) \cap \mathcal{R}\mathrm{eg} \subseteq \mathcal{L}(\mathbf{QV}) \ .$$

When $\mathbf{V}$ *is local, equality holds.*

Examples of tame varieties of finite monoids

- $\mathbf{A}$, *follows from* $\mathrm{MOD_m} \notin \mathsf{AC}^0$ *for all* $m \in \mathbb{N}, m \geq 2$ *(Furst-Saxe-Sipser, Ajtai)*.
- $\mathbf{DA}$, subcontribution 2.

Example of a non-tame variety of finite monoids

$\mathbf{J}$, subcontribution 3.

Proving tameness is a way to prove lower bounds

- $\mathbf{M}_{\mathsf{sol}}$ tame $\Rightarrow \mathsf{ACC}^0 \subsetneq \mathsf{NC}^1$.
- $\mathbf{A}$ tame $\Rightarrow \mathsf{AC}^0 \subsetneq \mathsf{ACC}^0$.
- $\mathbf{DA}$ tame $\Rightarrow \mathcal{P}(\mathbf{DA}) \subsetneq \mathsf{AC}^0$.

Consequence

Proving tameness is hard!

Non-tameness of $\mathbf{J}$

The proof

- $(a+b)^*ac(a+c)^*$ can be *p*-recognised by the syntactic monoid $(M,\star)$ of $(b+c)^*c(b+c)^*b(b+c)^*$.
- $(a+b)^*ac(a+c)^* \notin \mathcal{L}(\mathbf{QJ})$.
- So $\mathbf{J}$ isn't tame (otherwise $\mathcal{P}(\mathbf{J}) \cap \mathcal{R}\mathrm{eg} \subseteq \mathcal{L}(\mathbf{QJ})$).

Non-tameness of $\mathbf{J}$

The proof

- $(a+b)^*ac(a+c)^*$ can be *p*-recognised by the syntactic monoid $(M,\star)$ of $(b+c)^*c(b+c)^*b(b+c)^*$.
- $(a+b)^*ac(a+c)^* \notin \mathcal{L}(\mathbf{QJ})$.
- So $\mathbf{J}$ isn't tame (otherwise $\mathcal{P}(\mathbf{J}) \cap \mathcal{R}\text{eg} \subseteq \mathcal{L}(\mathbf{QJ})$).

The trick

$\psi\colon \{a,b,c\}^* \to M$ recognising $(b+c)^*c(b+c)^*b(b+c)^*$.

$$P = (1,f)(2,\psi)(1,\psi)(3,\psi)(2,\psi)(4,\psi)(3,\psi)\cdots(n,\psi)(n-1,\psi)$$

Non-tameness of $\mathbf{J}$

The proof

- $(a+b)^*ac(a+c)^*$ can be *p*-recognised by the syntactic monoid $(M,\star)$ of $(b+c)^*c(b+c)^*b(b+c)^*$.
- $(a+b)^*ac(a+c)^* \notin \mathcal{L}(\mathbf{QJ})$.
- So $\mathbf{J}$ isn't tame (otherwise $\mathcal{P}(\mathbf{J}) \cap \mathcal{R}eg \subseteq \mathcal{L}(\mathbf{QJ})$).

The trick

$\psi\colon \{a,b,c\}^* \to M$ recognising $(b+c)^*c(b+c)^*b(b+c)^*$.

$$P = (1,f)(2,\psi)(1,\psi)(3,\psi)(2,\psi)(4,\psi)(3,\psi)\cdots(n,\psi)(n-1,\psi)$$

$$P(abca\cdots ac) =$$

Non-tameness of $\mathbf{J}$

The proof

- $(a+b)^*ac(a+c)^*$ can be *p*-recognised by the syntactic monoid $(M,\star)$ of $(b+c)^*c(b+c)^*b(b+c)^*$.
- $(a+b)^*ac(a+c)^* \notin \mathcal{L}(\mathbf{QJ})$.
- So $\mathbf{J}$ isn't tame (otherwise $\mathcal{P}(\mathbf{J}) \cap \mathcal{R}\text{eg} \subseteq \mathcal{L}(\mathbf{QJ})$).

The trick

$\psi\colon \{a,b,c\}^* \to M$ recognising $(b+c)^*c(b+c)^*b(b+c)^*$.

$$P = (1,f)(2,\psi)(1,\psi)(3,\psi)(2,\psi)(4,\psi)(3,\psi)\cdots(n,\psi)(n-1,\psi)$$

$$P(abca\cdots ac) = \psi(b$$

Non-tameness of $\mathbf{J}$

The proof

- $(a+b)^*ac(a+c)^*$ can be *p*-recognised by the syntactic monoid $(M,\star)$ of $(b+c)^*c(b+c)^*b(b+c)^*$.
- $(a+b)^*ac(a+c)^* \notin \mathcal{L}(\mathbf{QJ})$.
- So $\mathbf{J}$ isn't tame (otherwise $\mathcal{P}(\mathbf{J}) \cap \mathcal{R}\mathrm{eg} \subseteq \mathcal{L}(\mathbf{QJ})$).

The trick

$\psi\colon \{a,b,c\}^* \to M$ recognising $(b+c)^*c(b+c)^*b(b+c)^*$.

$$P = (1,f)(2,\psi)(1,\psi)(3,\psi)(2,\psi)(4,\psi)(3,\psi)\cdots(n,\psi)(n-1,\psi)$$

$$P(abca\cdots ac) = \psi(ba$$

Non-tameness of $\mathbf{J}$

The proof

- $(a+b)^*ac(a+c)^*$ can be *p*-recognised by the syntactic monoid $(M,\star)$ of $(b+c)^*c(b+c)^*b(b+c)^*$.
- $(a+b)^*ac(a+c)^* \notin \mathcal{L}(\mathbf{QJ})$.
- So $\mathbf{J}$ isn't tame (otherwise $\mathcal{P}(\mathbf{J}) \cap \mathcal{R}\mathrm{eg} \subseteq \mathcal{L}(\mathbf{QJ})$).

The trick

$\psi\colon \{a,b,c\}^* \to M$ recognising $(b+c)^*c(b+c)^*b(b+c)^*$.

$$P = (1,f)(2,\psi)(1,\psi)(3,\psi)(2,\psi)(4,\psi)(3,\psi)\cdots(n,\psi)(n-1,\psi)$$

$$P(abca\cdots ac) = \psi(bac$$

Non-tameness of $\mathbf{J}$

The proof

- $(a+b)^*ac(a+c)^*$ can be p-recognised by the syntactic monoid $(M,\star)$ of $(b+c)^*c(b+c)^*b(b+c)^*$.
- $(a+b)^*ac(a+c)^* \notin \mathcal{L}(\mathbf{QJ})$.
- So $\mathbf{J}$ isn't tame (otherwise $\mathcal{P}(\mathbf{J}) \cap \mathcal{R}eg \subseteq \mathcal{L}(\mathbf{QJ})$).

The trick

$\psi\colon \{a,b,c\}^* \to M$ recognising $(b+c)^*c(b+c)^*b(b+c)^*$.

$$P = (1,f)(2,\psi)(1,\psi)(3,\psi)(2,\psi)(4,\psi)(3,\psi)\cdots(n,\psi)(n-1,\psi)$$

$$P(abca\cdots ac) = \psi(bacb$$

Non-tameness of $\mathbf{J}$

The proof

- $(a+b)^*ac(a+c)^*$ can be p-recognised by the syntactic monoid $(M,\star)$ of $(b+c)^*c(b+c)^*b(b+c)^*$.
- $(a+b)^*ac(a+c)^* \notin \mathcal{L}(\mathbf{QJ})$.
- So $\mathbf{J}$ isn't tame (otherwise $\mathcal{P}(\mathbf{J}) \cap \mathcal{R}eg \subseteq \mathcal{L}(\mathbf{QJ})$).

The trick

$\psi\colon \{a,b,c\}^* \to M$ recognising $(b+c)^*c(b+c)^*b(b+c)^*$.

$$P = (1,f)(2,\psi)(1,\psi)(3,\psi)(2,\psi)(4,\psi)(3,\psi)\cdots(n,\psi)(n-1,\psi)$$

$$P(abca\cdots ac) = \psi(bacba$$

Non-tameness of $\mathbf{J}$

The proof

- $(a+b)^*ac(a+c)^*$ can be p-recognised by the syntactic monoid $(M,\star)$ of $(b+c)^*c(b+c)^*b(b+c)^*$.
- $(a+b)^*ac(a+c)^* \notin \mathcal{L}(\mathbf{QJ})$.
- So $\mathbf{J}$ isn't tame (otherwise $\mathcal{P}(\mathbf{J}) \cap \mathcal{R}\text{eg} \subseteq \mathcal{L}(\mathbf{QJ})$).

The trick

$\psi\colon \{a,b,c\}^* \to M$ recognising $(b+c)^*c(b+c)^*b(b+c)^*$.

$$P = (1,f)(2,\psi)(1,\psi)(3,\psi)(2,\psi)(4,\psi)(3,\psi)\cdots(n,\psi)(n-1,\psi)$$

$$P(abca\cdots ac) = \psi(bacbac$$

Non-tameness of $\mathbf{J}$

The proof

- $(a+b)^*ac(a+c)^*$ can be p-recognised by the syntactic monoid $(M,\star)$ of $(b+c)^*c(b+c)^*b(b+c)^*$.
- $(a+b)^*ac(a+c)^* \notin \mathcal{L}(\mathbf{QJ})$.
- So $\mathbf{J}$ isn't tame (otherwise $\mathcal{P}(\mathbf{J}) \cap \mathcal{R}\mathrm{eg} \subseteq \mathcal{L}(\mathbf{QJ})$).

The trick

$\psi\colon \{a,b,c\}^* \to M$ recognising $(b+c)^*c(b+c)^*b(b+c)^*$.

$$P = (1,f)(2,\psi)(1,\psi)(3,\psi)(2,\psi)(4,\psi)(3,\psi)\cdots(n,\psi)(n-1,\psi)$$

$$P(abca\cdots ac) = \psi(bacbac\cdots ca)$$

Non-tameness of $\mathbf{J}$

The proof

- $(a+b)^*ac(a+c)^*$ can be p-recognised by the syntactic monoid $(M,\star)$ of $(b+c)^*c(b+c)^*b(b+c)^*$.
- $(a+b)^*ac(a+c)^* \notin \mathcal{L}(\mathbf{QJ})$.
- So $\mathbf{J}$ isn't tame (otherwise $\mathcal{P}(\mathbf{J}) \cap \mathcal{R}\mathrm{eg} \subseteq \mathcal{L}(\mathbf{QJ})$).

The trick

$\psi\colon \{a,b,c\}^* \to M$ recognising $(b+c)^*c(b+c)^*b(b+c)^*$.

$$P = (1,f)(2,\psi)(1,\psi)(3,\psi)(2,\psi)(4,\psi)(3,\psi)\cdots(n,\psi)(n-1,\psi)$$

$$P(abca\cdots ac) = \psi(bacbac\cdots ca)$$
$$P(baca\cdots ac) =$$

Non-tameness of $\mathbf{J}$

The proof

- $(a+b)^*ac(a+c)^*$ can be p-recognised by the syntactic monoid $(M,\star)$ of $(b+c)^*c(b+c)^*b(b+c)^*$.
- $(a+b)^*ac(a+c)^* \notin \mathcal{L}(\mathbf{QJ})$.
- So $\mathbf{J}$ isn't tame (otherwise $\mathcal{P}(\mathbf{J}) \cap \mathcal{R}eg \subseteq \mathcal{L}(\mathbf{QJ})$).

The trick

$\psi\colon \{a,b,c\}^* \to M$ recognising $(b+c)^*c(b+c)^*b(b+c)^*$.

$$P = (1,f)(2,\psi)(1,\psi)(3,\psi)(2,\psi)(4,\psi)(3,\psi)\cdots(n,\psi)(n-1,\psi)$$

$$P(abca\cdots ac) = \psi(bacbac\cdots ca)$$
$$P(baca\cdots ac) = \psi(abcaac\cdots ca)$$

Non-tameness of $\mathbf{J}$

Conjecture

$\mathcal{P}(\mathbf{J}) \cap \mathcal{R}\mathrm{eg} = \mathcal{L}(\mathbf{Q}(\mathbf{J} * \mathbf{D} \cap \langle \mathbf{DA} \rangle_{\mathbf{S}}))$.

What we know

- $\subseteq$: proved (using tameness of $\mathbf{DA}$ and Maciel-Péladeau-Thérien).
- $\supseteq$: proved in a particular case.

Tameness of $\mathbf{DA}$

Overview

Proof of tameness through semigroup-theoretic “lower bound” proof for $\mathcal{P}(\mathbf{DA})$; implies characterisation $\mathcal{P}(\mathbf{DA}) \cap \mathcal{R}\mathrm{eg} = \mathcal{L}(\mathbf{QDA})$.

Tameness of $\mathbf{DA}$

Overview

Proof of tameness through semigroup-theoretic "lower bound" proof for $\mathcal{P}(\mathbf{DA})$; implies characterisation $\mathcal{P}(\mathbf{DA}) \cap \mathcal{R}\mathrm{eg} = \mathcal{L}(\mathbf{QDA})$.

The proof

- Boils down to proving that $(c + ab)^*$, $(b + ab)^*$ and $b^*((ab^*)^k)^*$ for any $k \in \mathbb{N}_{\geq 2}$ are not in $\mathcal{P}(\mathbf{DA})$.

Tameness of $\mathbf{DA}$

Overview

Proof of tameness through semigroup-theoretic “lower bound” proof for $\mathcal{P}(\mathbf{DA})$; implies characterisation $\mathcal{P}(\mathbf{DA}) \cap \mathcal{R}\mathrm{eg} = \mathcal{L}(\mathbf{QDA})$.

The proof

- Boils down to proving that $(c + ab)^*$, $(b + ab)^*$ and $b^*\left((ab^*)^k\right)^*$ for any $k \in \mathbb{N}_{\geq 2}$ are not in $\mathcal{P}(\mathbf{DA})$.
- Very briefly, for $L = (c + ab)^*$, given P over $(M, \star) \in \mathbf{DA}$ that should recognise $L \cap \Sigma^n$, fix a constant number of letters in the input
  - so that it can still be completed into words inside and outside $L \cap \Sigma^n$;
  - but such that the output of P is the same for any input word.

Tameness of $\mathbf{DA}$

Some proof ideas

Fundamental property

For all $u, v, r \in M$, we have:

- if $u \,\mathfrak{R}\, v$ and $u \,\mathfrak{R}\, ur$, then $v \,\mathfrak{R}\, vr$;
- if $u \,\mathfrak{L}\, v$ and $u \,\mathfrak{L}\, ru$, then $v \,\mathfrak{L}\, rv$.

$$w = {*}{*}{*}c{*}{*}{*}\cdots{*}{*}a{*}{*}\cdots{*}{*}b{*}$$

$$u \; \overbrace{(i_1, f_1)(i_2, f_2)\cdots(i_j, f_j)\cdots(i_l, f_l)}^{P} \; v$$

Tameness of $\mathbf{DA}$

Some proof ideas

Fundamental property

For all $u, v, r \in M$, we have:

- if $u \,\mathfrak{R}\, v$ and $u \,\mathfrak{R}\, ur$, then $v \,\mathfrak{R}\, vr$;
- if $u \,\mathfrak{L}\, v$ and $u \,\mathfrak{L}\, ru$, then $v \,\mathfrak{L}\, rv$.

$$w = *{*}{*}c{*}{*}{*} \cdots {*}{*}a{*}{*} \cdots {*}{*}b*$$

$$u \; \overbrace{\underbrace{(i_1, f_1)(i_2, f_2) \cdots}_{P'_j} \color{red}{(i_j, f_j)} \underbrace{\cdots (i_l, f_l)}_{P''_j}}^{P} \; v$$

Fundamental property

For all $u, v, r \in M$, we have:

- if $u \mathfrak{R} v$ and $u \mathfrak{R} ur$, then $v \mathfrak{R} vr$;
- if $u \mathfrak{L} v$ and $u \mathfrak{L} ru$, then $v \mathfrak{L} rv$.

$$w = ***c*** \cdots **a{\color{red}b}* \cdots **b*$$

$$u \ \underbrace{(i_1, f_1)(i_2, f_2) \cdots}_{P'_j} f_j(b) \underbrace{\cdots (i_l, f_l)}_{P''_j} \ v$$

Tameness of $\mathbf{DA}$

Some proof ideas

Fundamental property

For all $u, v, r \in M$, we have:

- if $u \,\mathfrak{R}\, v$ and $u \,\mathfrak{R}\, ur$, then $v \,\mathfrak{R}\, vr$;
- if $u \,\mathfrak{L}\, v$ and $u \,\mathfrak{L}\, ru$, then $v \,\mathfrak{L}\, rv$.

$$w = {*}{*}{*}c{*}{*}{*}\cdots{*}{*}ab{*}\cdots{*}{*}b{*}$$

$$u \;\underbrace{(i_1, f_1)(i_2, f_2)\cdots}_{P'_j} f_j(b) \underbrace{\cdots(i_l, f_l)}_{P''_j}\; v$$

Tameness of $\mathbf{DA}$

Some proof ideas

Fundamental property

For all $u, v, r \in M$, we have:

- if $u \mathfrak{R} v$ and $u \mathfrak{R} ur$, then $v \mathfrak{R} vr$;
- if $u \mathfrak{L} v$ and $u \mathfrak{L} ru$, then $v \mathfrak{L} rv$.

$$w = *b*cc** \cdots **ab* \cdots *ab*$$

$$t' \qquad f_j(b) \ \underbrace{\cdots (i_l, f_l)}_{P''_j} \ v$$

Tameness of $\mathbf{DA}$

Some proof ideas

Fundamental property

For all $u, v, r \in M$, we have:

- if $u \,\mathfrak{R}\, v$ and $u \,\mathfrak{R}\, ur$, then $v \,\mathfrak{R}\, vr$;
- if $u \,\mathfrak{L}\, v$ and $u \,\mathfrak{L}\, ru$, then $v \,\mathfrak{L}\, rv$.

$$w = {*}b{*}cc{*}{*}\cdots{*}{*}ab{*}\cdots{*}ab{*}$$

$$t' \qquad\qquad f_j(b) \; \underbrace{\cdots(i_l, f_l)}_{P''_j} \; v$$

Tameness of $\mathbf{DA}$

Some proof ideas

Fundamental property

For all $u, v, r \in M$, we have:

- if $u \,\mathfrak{R}\, v$ and $u \,\mathfrak{R}\, ur$, then $v \,\mathfrak{R}\, vr$;
- if $u \,\mathfrak{L}\, v$ and $u \,\mathfrak{L}\, ru$, then $v \,\mathfrak{L}\, rv$.

$$w = ab{*}cc{*}{*}\cdots c{*}ab{*}\cdots{*}ab{*}$$

$$t''$$

Fundamental property

For all $u, v, r \in M$, we have:

- if $u \,\mathfrak{R}\, v$ and $u \,\mathfrak{R}\, ur$, then $v \,\mathfrak{R}\, vr$;
- if $u \,\mathfrak{L}\, v$ and $u \,\mathfrak{L}\, ru$, then $v \,\mathfrak{L}\, rv$.

$$w = ab{*}cc{*}{*}\cdots c{*}ab{*}\cdots{*}ab{*}$$

$$t''$$

Conclusion and perspectives

Contributions

1. Formal, measure-independent, treatment and study of Nečiporuk's method.

2. Better understanding of computational power of programs over monoids taken from small varieties of finite monoids.
   - New tameness notion; for $\mathbf{V}$ implies that $\mathcal{P}(\mathbf{V}) \cap \mathcal{R}eg \subseteq \mathcal{L}(\mathbf{QV})$ (equality when $\mathbf{V}$ local).
   - Proof of tameness of $\mathbf{DA}$ and characterisation of $\mathcal{P}(\mathbf{DA}) \cap \mathcal{R}eg$.
   - Proof of non-tameness of $\mathbf{J}$ and conjectural characterisation of $\mathcal{P}(\mathbf{J}) \cap \mathcal{R}eg$, partially shown.

Conclusion and perspectives

Some future directions

1. In straightforward continuation
   - Fully characterise $\mathcal{P}(\mathbf{J}) \cap \mathcal{R}\mathrm{eg}$.
   - Progressively study tameness for hierarchy inside $\mathbf{A}$ (in view of reproving its tameness).

2. More adventurous
   - Explore more general versions of Nečiporuk's method.
   - Study tameness for varieties of finite non-aperiodic monoids.
   - Understand better properties of tameness.

Thank you for listening.