Choiceless Polynomial Time

Benedikt Pago [1]

ESSLLI 2025, Bochum

[1]University of Cambridge

UNIVERSITY OF CAMBRIDGE
Department of Computer Science and Technology

Landscape of polynomial time logics

Ptime

$\rotatebox{-60}{$\lneqq$}$

Group order logic

$\rotatebox{90}{$\lneq$}$

Rank logic

$\rotatebox{60}{$\lneq$}$

FPC

$\rotatebox{90}{$\lneq$}$

IFP

$\rotatebox{90}{$\lneq$}$

First-order logic (FO)

PTIME

$\leq$ $\qquad\qquad$ $\geq$ **(=?)**

Group order logic

$\lneq$

Rank logic $\xleftrightarrow[\neq]{}$ **Choiceless Polynomial Time** (CPT)

$\gneq$ $\qquad\qquad$ $\lneq$

FPC

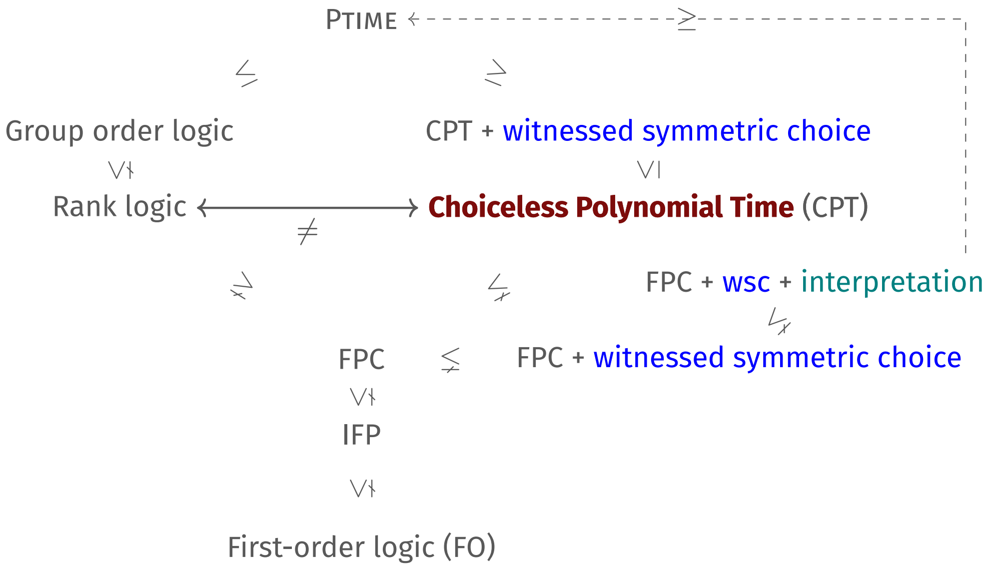
$\lneq$

IFP

$\lneq$

First-order logic (FO)

Landscape of polynomial time logics



Landscape of polynomial time logics

PTIME

Group order logic

Rank logic

CPT + witnessed symmetric choice

Choiceless Polynomial Time (CPT)

FPC + wsc + interpretation

FPC

FPC + witnessed symmetric choice

IFP

First-order logic (FO)

Candidates for capturing PTIME

Up to now, all logics were *extensions* of FO/fixed-point logics.

Up to now, all logics were *extensions* of FO/fixed-point logics.

CPT can be seen as a *restriction*:

It is obtained by enforcing **polynomial-time Turing machines** to be isomorphism-invariant [Blass, Gurevich, Shelah, 1999].

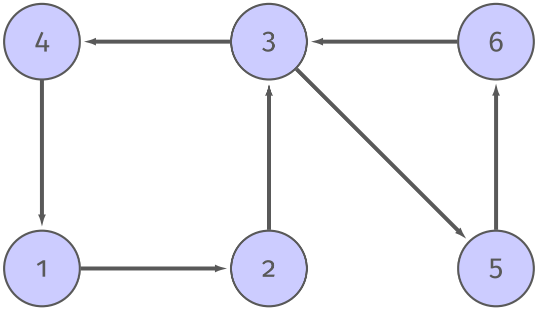
```c
void DFS(int graph[MAX_NODES][MAX_NODES], bool visited[MAX_NODES], int current_node, int num_nodes)
{
  visited[current_node] = true;

  for (int i = 0; i < num_nodes; i++) {
    if (graph[current_node][i] == 1 && !visited[i]) {
      DFS(graph, visited, i, num_nodes);
    }
  }
}
```

A C-program for depth-first search

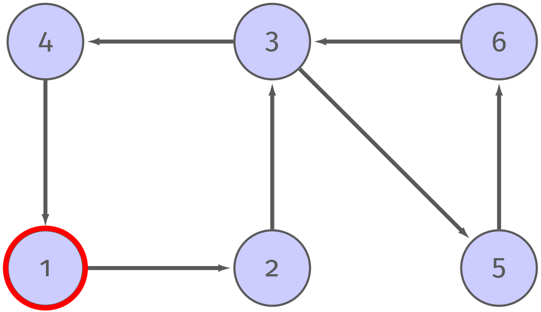
```
void G = (V, E), current_node ∈ V DES])
{
  visited[current_node] = true;

  all i ∈ V do in parallel: i++) {
    if (graph[current_node][i] == 1 && !visited[i]) {
      G, i graph);
    }
  }
}
```

A choiceless program for DFS

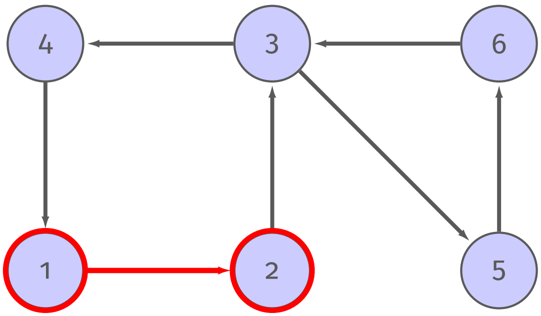
DFS with choices



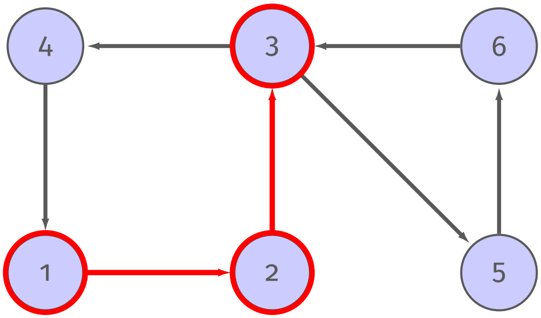
DFS with choices



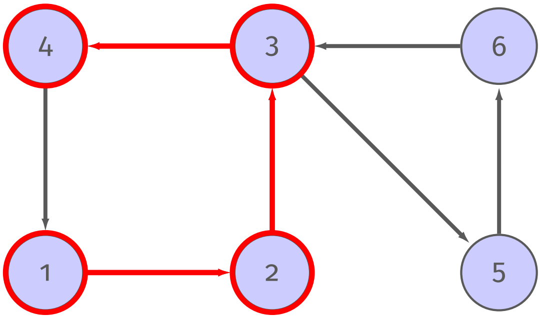
DFS with choices



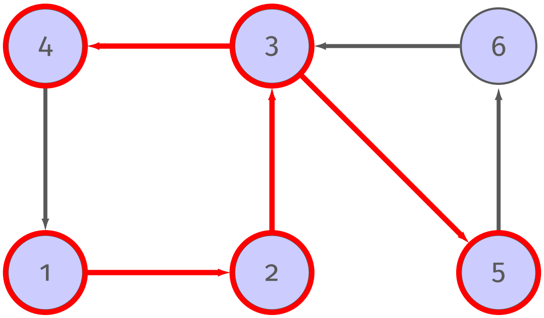
DFS with choices



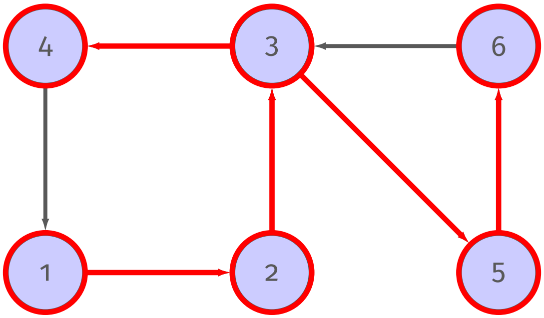
DFS with choices



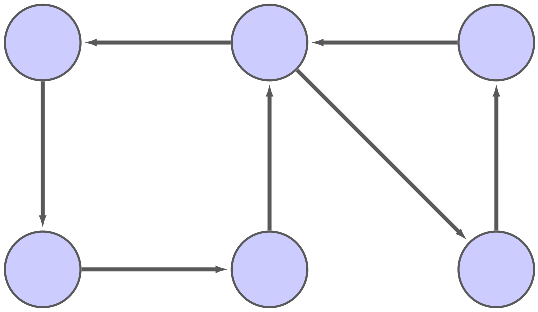
DFS with choices



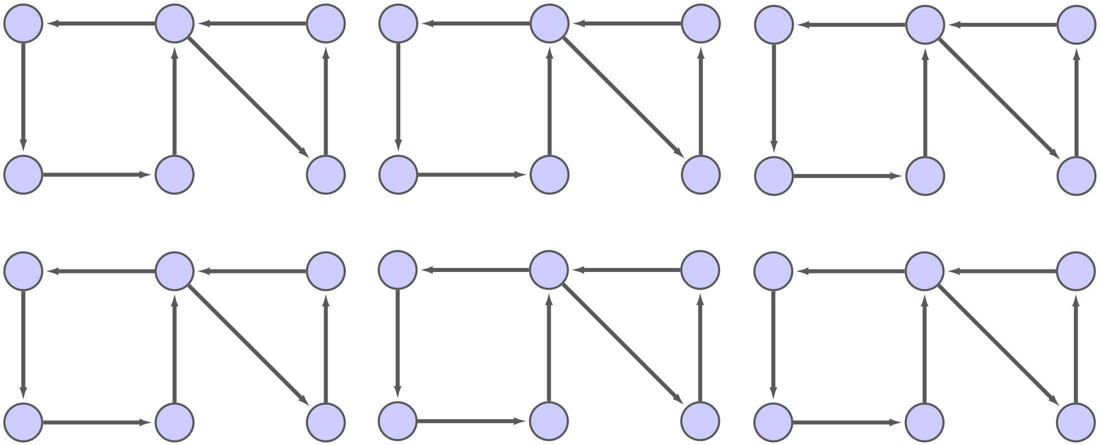
DFS with choices



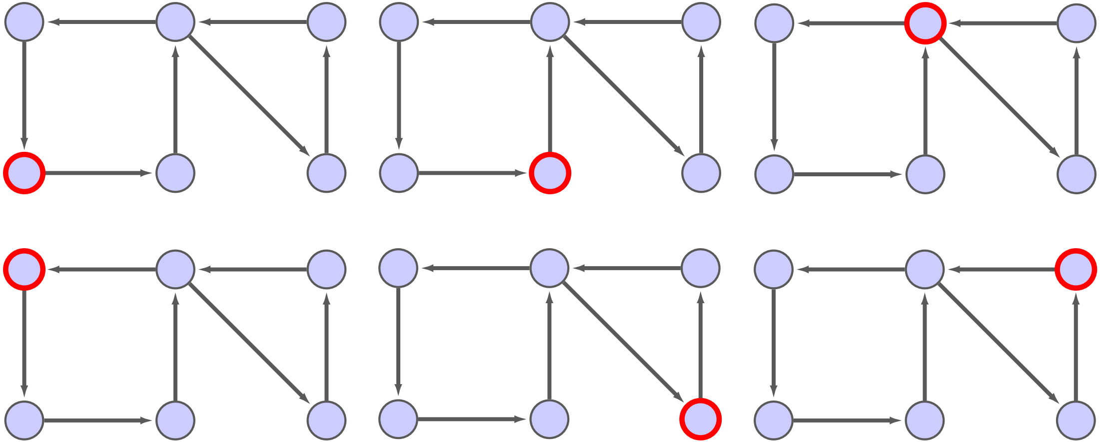
DFS without choices



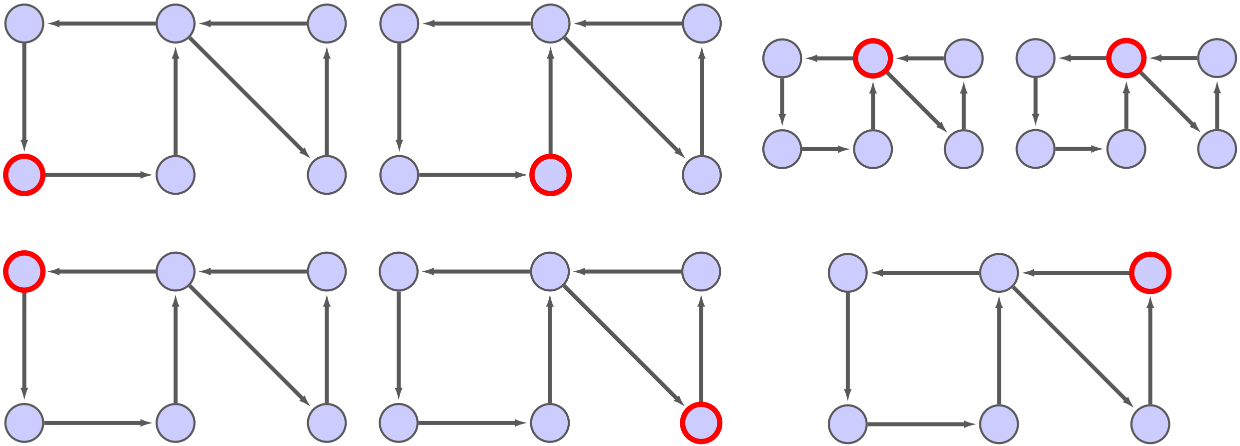
DFS without choices



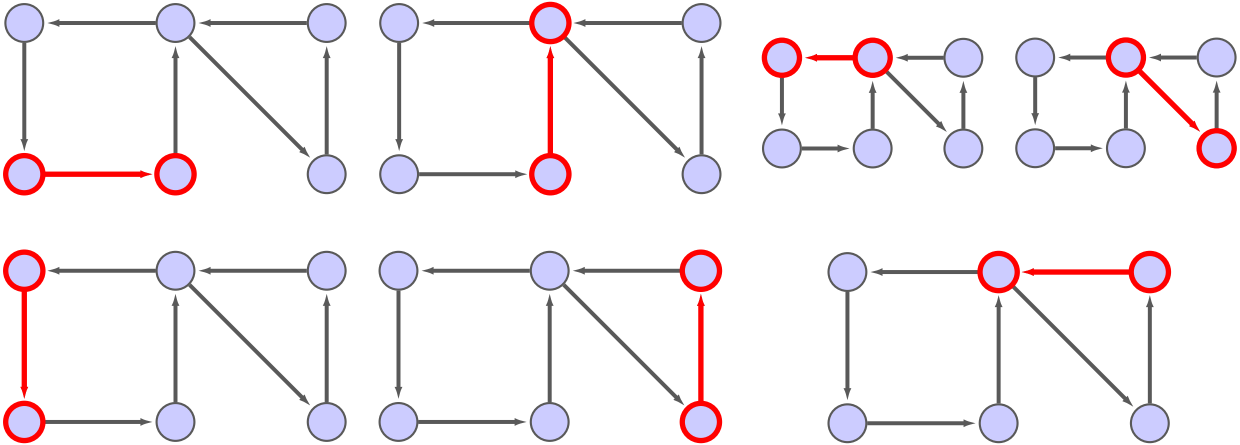
DFS without choices



DFS without choices



DFS without choices



1. Original definition: abstract state **machine model** [Blass, Gurevich, Shelah, 1999].
2. **BGS-logic** [Rossman, 2010].
3. **Polynomial Interpretation Logic** [Grädel, Pakusa, Schalthöfer, Kaiser, 2015].

Definition (FO-interpretation)

A σ-structure $\mathfrak{B}$ is **FO-interpretable** in a τ-structure $\mathfrak{A}$ if there exist formulas $\varphi_\delta(\bar{x}), \varphi(\bar{x}, \bar{y})_\approx, (\varphi_R)_{R \in \sigma}$ and a $k \in \mathbb{N}$ such that

- $B = \{[\bar{a}]_\approx \mid \bar{a} \in A^k, \mathfrak{A} \models \varphi_\delta(\bar{a})\}$
- For each r-ary $R \in \sigma$, $R^\mathfrak{B} = \{(\bar{a}_1, \ldots, \bar{a}_r) \mid \mathfrak{A} \models \varphi_R(\bar{a}_1, \ldots, \bar{a}_r)\}$.

Polynomial Interpretation Logic

Definition (PIL, simplified)

Sentences of PIL are of the form $(\mathcal{I}_{\text{step}}, \psi_{\text{end}}, \psi_{\text{out}}, p(n))$, where

- $\mathcal{I}_{\text{step}}$ is an FO-interpretation,
- $\psi_{\text{end}}, \psi_{\text{out}}$ are FO-sentences,
- $p(n)$ is a polynomial serving as a time and space bound.

Definition (PIL, simplified)

Sentences of PIL are of the form $(\mathcal{I}_{\text{step}}, \psi_{\text{end}}, \psi_{\text{out}}, p(n))$, where

- $\mathcal{I}_{\text{step}}$ is an FO-interpretation,
- $\psi_{\text{end}}, \psi_{\text{out}}$ are FO-sentences,
- $p(n)$ is a polynomial serving as a time and space bound.

$(\mathcal{I}_{\text{step}}, \psi_{\text{end}}, \psi_{\text{out}}, p(n))$ defines a **run** in any structure $\mathfrak{A}$:

$$\mathfrak{A}, \mathcal{I}_{\text{step}}(\mathfrak{A}), \mathcal{I}_{\text{step}}(\mathcal{I}_{\text{step}}(\mathfrak{A})), \dots, \mathfrak{B}.$$

where $\mathfrak{B}$ is the first structure in the run with $\mathfrak{B} \models \psi_{\text{end}}$.

Definition (PIL, simplified)

Sentences of PIL are of the form $(\mathcal{I}_{\text{step}}, \psi_{\text{end}}, \psi_{\text{out}}, p(n))$, where

- $\mathcal{I}_{\text{step}}$ is an FO-interpretation,
- $\psi_{\text{end}}, \psi_{\text{out}}$ are FO-sentences,
- $p(n)$ is a polynomial serving as a time and space bound.

$(\mathcal{I}_{\text{step}}, \psi_{\text{end}}, \psi_{\text{out}}, p(n))$ defines a **run** in any structure $\mathfrak{A}$:

$$\mathfrak{A}, \mathcal{I}_{\text{step}}(\mathfrak{A}), \mathcal{I}_{\text{step}}(\mathcal{I}_{\text{step}}(\mathfrak{A})), \dots, \mathfrak{B}.$$

where $\mathfrak{B}$ is the first structure in the run with $\mathfrak{B} \models \psi_{\text{end}}$.
If the number of steps or size of the structures *exceeds $p(|\mathfrak{A}|)$*, it is *aborted*.

$$\mathfrak{A} \models (\mathcal{I}_{\text{step}}, \psi_{\text{end}}, \psi_{\text{out}}, p(n)) \iff \text{the run terminates with } \mathfrak{B}, \text{ s.t. } \mathfrak{B} \models \psi_{\text{out}}.$$

Power of PIL: Using higher-dimensional interpretations, a PIL sentence can *grow the input structure* arbitrarily.

Computing linear orders

$$\mathcal{I}_{\mathsf{step}} := (\varphi_\delta(x_1, x_2) := \mathsf{true},$$
$$\varphi_<(x_1, x_2, y_1, y_2) := (x_1 = x_2 \wedge y_1 = y_2 \wedge x_1 < y_1) \vee (x_1 = x_2 \wedge y_1 \neq y_2 \wedge y_1 = x_1)).$$

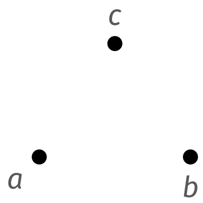
Ignore $\psi_{\mathsf{end}}, \psi_{\mathsf{out}}, p(n)$ for now.

Example: Defining a linear order in PIL

Computing linear orders

$$\begin{aligned}\mathcal{I}_{\text{step}} := ((\varphi_\delta(x_1, x_2) &:= \text{true},\\ \varphi_<(x_1, x_2, y_1, y_2) &:= (x_1 = x_2 \wedge y_1 = y_2 \wedge x_1 < y_1) \vee (x_1 = x_2 \wedge y_1 \neq y_2 \wedge y_1 = x_1)).\end{aligned}$$

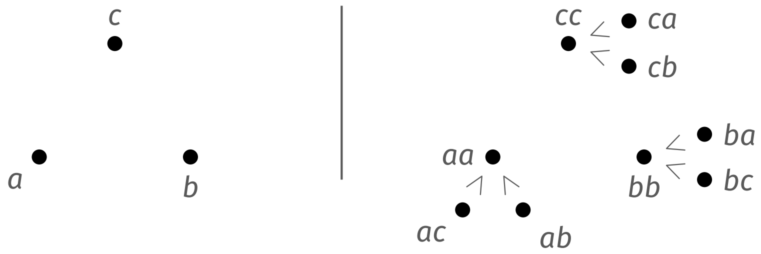
Ignore $\psi_{\text{end}}, \psi_{\text{out}}, p(n)$ for now.



Computing linear orders

$$\mathcal{I}_{\text{step}} := ((\varphi_\delta(x_1, x_2) := \text{true},$$
$$\varphi_<(x_1, x_2, y_1, y_2) := (x_1 = x_2 \wedge y_1 = y_2 \wedge x_1 < y_1) \vee (x_1 = x_2 \wedge y_1 \neq y_2 \wedge y_1 = x_1)).$$

Ignore $\psi_{\text{end}}, \psi_{\text{out}}, p(n)$ for now.

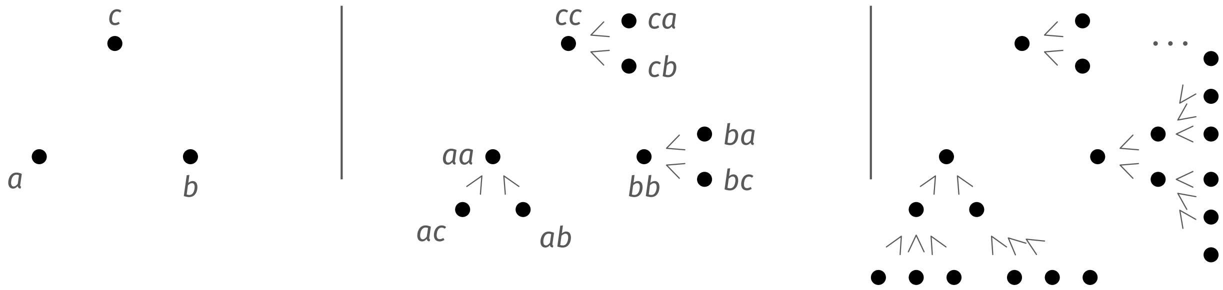


Example: Defining a linear order in PIL

Computing linear orders

$$\mathcal{I}_{\text{step}} := ((\varphi_\delta(x_1, x_2) := \text{true}, \\ \varphi_<(x_1, x_2, y_1, y_2) := (x_1 = x_2 \land y_1 = y_2 \land x_1 < y_1) \lor (x_1 = x_2 \land y_1 \neq y_2 \land y_1 = x_1)).$$

Ignore $\psi_{\text{end}}, \psi_{\text{out}}, p(n)$ for now.



Weakness of CPT

The stages $\mathfrak{A}, \mathfrak{A}_1, \mathfrak{A}_2, \dots$ of a PIL computation are closed under the group $\mathbf{Aut}(\mathfrak{A})$.

If $\mathfrak{A}$ is very symmetric, then intuitively, the structures $\mathfrak{A}_i$ quickly become super-polynomially large, and the computation cannot be carried out in PIL.

Presentations of CPT

1. Original definition: abstract state machine model.
2. **BGS-logic**: Useful for lower bounds.
3. Polynomial Interpretation Logic: Most understandable.

BGS-logic is FO augmented with terms for the creation of *hereditarily finite sets*:

Syntax and semantics of set-terms:

Let $\mathfrak{A} = (A, \tau)$ be a structure.

- Universe of structure: $[\![\texttt{Atoms}]\!]^{\mathfrak{A}} = A$.

BGS-logic is FO augmented with terms for the creation of *hereditarily finite sets*:

Syntax and semantics of set-terms:

Let $\mathfrak{A} = (A, \tau)$ be a structure.

- Universe of structure: $[\![\texttt{Atoms}]\!]^{\mathfrak{A}} = A$.
- **Set constructor:** $[\![\mathrm{Pair}(r, s)]\!]^{\mathfrak{A}} = \{[\![r]\!]^{\mathfrak{A}}, [\![s]\!]^{\mathfrak{A}}\}$.

BGS-logic is FO augmented with terms for the creation of *hereditarily finite sets*:

Syntax and semantics of set-terms:

Let $\mathfrak{A} = (A, \tau)$ be a structure.

- Universe of structure: $[\![\texttt{Atoms}]\!]^{\mathfrak{A}} = A$.
- **Set constructor:** $[\![\mathrm{Pair}(r, s)]\!]^{\mathfrak{A}} = \{[\![r]\!]^{\mathfrak{A}}, [\![s]\!]^{\mathfrak{A}}\}$.
- **Comprehension term:** $[\![\{s : x \in r : \varphi\}]\!]^{\mathfrak{A}} = \{[\![s(x)]\!]^{\mathfrak{A}} \mid x \in [\![r]\!]^{\mathfrak{A}}, \mathfrak{A} \models \varphi(x)\}$.

BGS-logic is FO augmented with terms for the creation of *hereditarily finite sets*:

Syntax and semantics of set-terms:
Let $\mathfrak{A} = (A, \tau)$ be a structure.

- Universe of structure: $[\![\mathtt{Atoms}]\!]^{\mathfrak{A}} = A$.
- **Set constructor:** $[\![\mathrm{Pair}(r, s)]\!]^{\mathfrak{A}} = \{[\![r]\!]^{\mathfrak{A}}, [\![s]\!]^{\mathfrak{A}}\}$.
- **Comprehension term:** $[\![\{s : x \in r : \varphi\}]\!]^{\mathfrak{A}} = \{[\![s(x)]\!]^{\mathfrak{A}} \mid x \in [\![r]\!]^{\mathfrak{A}}, \mathfrak{A} \models \varphi(x)\}$.
- **Iteration term:** $[\![s(x)^*]\!]^{\mathfrak{A}}$ is the least fixed-point of the sequence $[\![s(\emptyset)]\!]^{\mathfrak{A}}, [\![s(s(\emptyset))]\!]^{\mathfrak{A}}, [\![s(s(s(\emptyset)))]\!]^{\mathfrak{A}}, \dots$.

Choiceless Polynomial Time as BGS-logic

BGS-logic is FO augmented with terms for the creation of *hereditarily finite sets*:

Syntax and semantics of set-terms:
Let $\mathfrak{A} = (A, \tau)$ be a structure.

- Universe of structure: $[\![\mathtt{Atoms}]\!]^{\mathfrak{A}} = A$.
- **Set constructor:** $[\![\mathrm{Pair}(r, s)]\!]^{\mathfrak{A}} = \{[\![r]\!]^{\mathfrak{A}}, [\![s]\!]^{\mathfrak{A}}\}$.
- **Comprehension term:** $[\![\{s : x \in r : \varphi\}]\!]^{\mathfrak{A}} = \{[\![s(x)]\!]^{\mathfrak{A}} \mid x \in [\![r]\!]^{\mathfrak{A}}, \mathfrak{A} \models \varphi(x)\}$.
- **Iteration term:** $[\![s(x)^*]\!]^{\mathfrak{A}}$ is the least fixed-point of the sequence $[\![s(\emptyset)]\!]^{\mathfrak{A}}, [\![s(s(\emptyset))]\!]^{\mathfrak{A}}, [\![s(s(s(\emptyset)))]\!]^{\mathfrak{A}}, \ldots$.
- Iteration terms are accompanied by a polynomial resource bound for time and space: (s^*, p).

CPT is strictly stronger than fixed-points

Theorem

$$\text{FPC} \lneq \text{CPT}.$$

Proof.

- FPC cannot distinguish CFI graphs.

CPT is strictly stronger than fixed-points

Theorem

$$\mathrm{FPC} \lneq \mathrm{CPT}.$$

Proof.

- FPC cannot distinguish CFI graphs.
- This is also true if they are augmented with exponentially many isolated vertices (*padding*).

CPT is strictly stronger than fixed-points

Theorem

$$\text{FPC} \lneq \text{CPT}.$$

Proof.

- FPC cannot distinguish CFI graphs.
- This is also true if they are augmented with exponentially many isolated vertices (*padding*).
- CPT can distinguish the padded CFI structures by *computing all linear orders* on the non-padding part and running the PTIME-algorithm that distinguishes them.

Inexpressibility results for CPT?

- It is known that CPT cannot define the set of all hyperplanes in a given finite vector space [Rossman, 2010].

Inexpressibility results for CPT?

- It is known that CPT cannot define the set of all hyperplanes in a given finite vector space [Rossman, 2010].
- But there is no known decision problem in PTIME that is not in CPT.

Inexpressibility results for CPT?

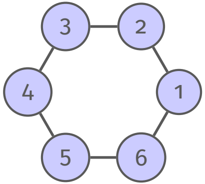
- It is known that CPT cannot define the set of all hyperplanes in a given finite vector space [Rossman, 2010].
- But there is no known decision problem in PTIME that is not in CPT.
- Obvious idea: Try CFI graphs...

Definability on different classes of base graphs:

Linearly ordered base graphs [1]	✓
Preordered base graphs with log-size colour classes [2]	✓
Base graphs with linear degree [2]	✓
General unordered base graphs	?

[1] [Dawar, Richerby, Rossman, 2008]
[2] [Pakusa, Schalthöfer, Selman, 2018]

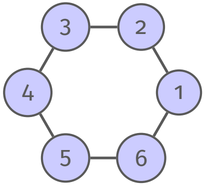
Is the CFI-query definable in Choiceless Polynomial Time?



Definability on different classes of base graphs:

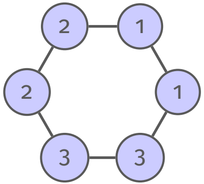
Linearly ordered base graphs [1]	✔
Preordered base graphs with log-size colour classes [2]	✔
Base graphs with linear degree [2]	✔
General unordered base graphs	?

[1][Dawar, Richerby, Rossman, 2008]
[2][Pakusa, Schalthöfer, Selman, 2018]



Definability on different classes of base graphs:

Linearly ordered base graphs [1]	✔
Preordered base graphs with log-size colour classes [2]	✔
Base graphs with linear degree [2]	✔
General unordered base graphs	?

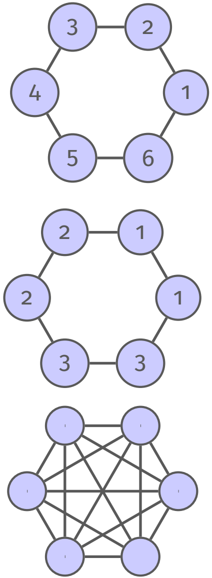


[1][Dawar, Richerby, Rossman, 2008]
[2][Pakusa, Schalthöfer, Selman, 2018]

Is the CFI-query definable in Choiceless Polynomial Time?

Definability on different classes of base graphs:

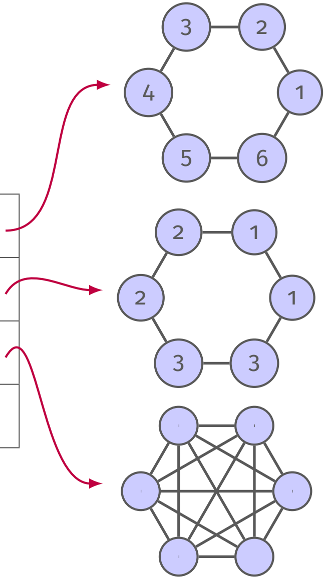
Linearly ordered base graphs [1]	✓
Preordered base graphs with log-size colour classes [2]	✓
Base graphs with linear degree [2]	✓
General unordered base graphs	?



[1] [Dawar, Richerby, Rossman, 2008]
[2] [Pakusa, Schalthöfer, Selman, 2018]

Definability on different classes of base graphs:

Linearly ordered base graphs [1]	✓
Preordered base graphs with log-size colour classes [2]	✓
Base graphs with linear degree [2]	✓
General unordered base graphs	?



[1] [Dawar, Richerby, Rossman, 2008]
[2] [Pakusa, Schalthöfer, Selman, 2018]

Non-definability of preorders

Theorem (P., CSL 2021)

There exists a family of unordered base graphs (with sub-linear degree) where no preorder with log-size colour classes is CPT-definable.

$\Rightarrow$ There are CFI-instances that cannot be handled with any of the known CPT-algorithms.

Non-definability of preorders

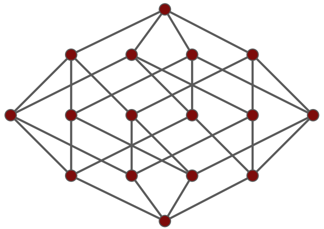
Theorem (P., CSL 2021)

There exists a family of unordered base graphs (with sub-linear degree) where no preorder with log-size colour classes is CPT-definable.

$\Rightarrow$ There are CFI-instances that cannot be handled with any of the known CPT-algorithms.

Theorem (technical version)

Any preorder with log-size colour classes on the n-dimensional hypercube has an orbit of super-polynomial size.



4-dimensional hypercube

Problem: The expressive power of a CPT sentence is *not limited* by its *number of variables*.
The creation of h.f. sets can "simulate" extra variables.

Problem: The expressive power of a CPT sentence is *not limited* by its *number of variables*.
The creation of h.f. sets can “simulate” extra variables.

$\Longrightarrow$ There is *no pebble game* that characterises CPT.

Problem: The expressive power of a CPT sentence is *not limited* by its *number of variables*.
The creation of h.f. sets can "simulate" extra variables.

$\Longrightarrow$ There is *no pebble game* that characterises CPT.

But: A property of the created h.f. sets controls the expressivity like the pebble number.

Definition (Support)

Let x a h.f. set over $\mathfrak{A}$. A **support** for x is a tuple α over $\mathfrak{A}$ such that “fixing α also fixes x”.

Formally: For every $\pi \in \mathbf{Aut}(\mathfrak{A})$ such that $\pi(\alpha) = \alpha$, it holds $\pi(x) = x$.

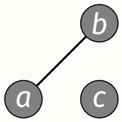
The support of a hereditarily finite set

Definition (Support)

Let x a h.f. set over $\mathfrak{A}$. A **support** for x is a tuple α over $\mathfrak{A}$ such that "fixing α also fixes x".

Formally: For every $\pi \in \mathbf{Aut}(\mathfrak{A})$ such that $\pi(\alpha) = \alpha$, it holds $\pi(x) = x$.

Example:



The structure $\mathfrak{A}$

- $\mathbf{Aut}(\mathfrak{A}) = \{\mathrm{id}, (a\ b)\}$.
- Let $x = \{a, \{c\}\}$.
- A trivial support for x: ac.
- A **smallest support**: a (or b).

An approach to prove indistinguishability in CPT

- Let $\mathfrak{A}, \mathfrak{B}$ satisfy $\mathfrak{A} \equiv_{\mathcal{C}^k} \mathfrak{B}$.

An approach to prove indistinguishability in CPT

- Let $\mathfrak{A}, \mathfrak{B}$ satisfy $\mathfrak{A} \equiv_{\mathcal{C}^k} \mathfrak{B}$.
- $\implies$ Any CPT-program distinguishing $\mathfrak{A}$ and $\mathfrak{B}$ must construct (on input $\mathfrak{A}$) a h.f. set x whose smallest support has size $\geq k$ [Dawar, Richerby, Rossman, 2008].

An approach to prove indistinguishability in CPT

- Let $\mathfrak{A}, \mathfrak{B}$ satisfy $\mathfrak{A} \equiv_{\mathcal{C}^k} \mathfrak{B}$.
- $\implies$ Any CPT-program distinguishing $\mathfrak{A}$ and $\mathfrak{B}$ must construct (on input $\mathfrak{A}$) a h.f. set x whose smallest support has size $\geq k$ [Dawar, Richerby, Rossman, 2008].
- Since CPT is *isomorphism-invariant*, it must construct x together with all its automorphic images $\mathbf{Orb}_{\mathfrak{A}}(x) := \{\pi(x) \mid \pi \in \mathbf{Aut}(\mathfrak{A})\}$.

An approach to prove indistinguishability in CPT

- Let $\mathfrak{A}, \mathfrak{B}$ satisfy $\mathfrak{A} \equiv_{\mathcal{C}^k} \mathfrak{B}$.
- $\implies$ Any CPT-program distinguishing $\mathfrak{A}$ and $\mathfrak{B}$ must construct (on input $\mathfrak{A}$) a h.f. set x whose smallest support has size $\geq k$ [Dawar, Richerby, Rossman, 2008].
- Since CPT is *isomorphism-invariant*, it must construct x together with all its automorphic images $\mathbf{Orb}_{\mathfrak{A}}(x) := \{\pi(x) \mid \pi \in \mathbf{Aut}(\mathfrak{A})\}$.
- Since CPT is *polynomially bounded*, $|\mathbf{Orb}_{\mathfrak{A}}(x)|$ must be polynomial in $|\mathfrak{A}|$.

An approach to prove indistinguishability in CPT

- Let $\mathfrak{A}, \mathfrak{B}$ satisfy $\mathfrak{A} \equiv_{\mathcal{C}^k} \mathfrak{B}$.
- $\implies$ Any CPT-program distinguishing $\mathfrak{A}$ and $\mathfrak{B}$ must construct (on input $\mathfrak{A}$) a h.f. set x whose smallest support has size $\geq k$ [Dawar, Richerby, Rossman, 2008].
- Since CPT is *isomorphism-invariant*, it must construct x together with all its automorphic images $\mathbf{Orb}_{\mathfrak{A}}(x) := \{\pi(x) \mid \pi \in \mathbf{Aut}(\mathfrak{A})\}$.
- Since CPT is *polynomially bounded*, $|\mathbf{Orb}_{\mathfrak{A}}(x)|$ must be polynomial in $|\mathfrak{A}|$.
- $\implies$ One way to show CPT-indistinguishability is to prove that **large support** implies a **large orbit**.

Estimating orbit-sizes

Goal: Establish *lower bounds on orbit size* depending on support size.

Estimating orbit-sizes

Goal: Establish *lower bounds on orbit size* depending on support size.

Example:

- Let $\mathfrak{A} = \{1, 2, \dots, n\}$, and x some subset, e.g. $x = \{1, 2, ..., k\}$.

Estimating orbit-sizes

Goal: Establish *lower bounds on orbit size* depending on support size.

Example:

- Let $\mathfrak{A} = \{1, 2, \dots, n\}$, and x some subset, e.g. $x = \{1, 2, ..., k\}$.
- We have $\mathbf{Aut}(\mathfrak{A}) = \mathbf{Sym}(n)$.

Estimating orbit-sizes

Goal: Establish *lower bounds on orbit size* depending on support size.

Example:

- Let $\mathfrak{A} = \{1, 2, \ldots, n\}$, and x some subset, e.g. $x = \{1, 2, ..., k\}$.
- We have $\textbf{Aut}(\mathfrak{A}) = \textbf{Sym}(n)$.
- The smallest support S of x is $S = x$.

Estimating orbit-sizes

Goal: Establish *lower bounds on orbit size* depending on support size.

Example:

- Let $\mathfrak{A} = \{1, 2, \ldots, n\}$, and x some subset, e.g. $x = \{1, 2, ..., k\}$.
- We have $\mathbf{Aut}(\mathfrak{A}) = \mathbf{Sym}(n)$.
- The smallest support S of x is $S = x$.
- $\mathbf{Orb}_{\mathfrak{A}}(x)$ is the set of all k-element subsets of $[n]$.

Goal: Establish *lower bounds on orbit size* depending on support size.

Example:

- Let $\mathfrak{A} = \{1, 2, \ldots, n\}$, and x some subset, e.g. $x = \{1, 2, ..., k\}$.
- We have $\mathbf{Aut}(\mathfrak{A}) = \mathbf{Sym}(n)$.
- The smallest support S of x is $S = x$.
- $\mathbf{Orb}_{\mathfrak{A}}(x)$ is the set of all k-element subsets of $[n]$.
- $|\mathbf{Orb}_{\mathfrak{A}}(x)| = \binom{n}{k} \approx n^k$ is polynomial in n iff $k = |S|$ is constant.

Estimating orbit-sizes

Goal: Establish *lower bounds on orbit size* depending on support size.

Example:

- Let $\mathfrak{A} = \{1, 2, \ldots, n\}$, and x some subset, e.g. $x = \{1, 2, ..., k\}$.
- We have $\mathbf{Aut}(\mathfrak{A}) = \mathbf{Sym}(n)$.
- The smallest support S of x is $S = x$.
- $\mathbf{Orb}_{\mathfrak{A}}(x)$ is the set of all k-element subsets of $[n]$.
- $|\mathbf{Orb}_{\mathfrak{A}}(x)| = \binom{n}{k} \approx n^k$ is polynomial in n iff $k = |S|$ is constant.

Goal: Establish *lower bounds on orbit size* depending on support size.

Example:

- Let $\mathfrak{A} = \{1, 2, \ldots, n\}$, and x some subset, e.g. $x = \{1, 2, ..., k\}$.
- We have $\mathbf{Aut}(\mathfrak{A}) = \mathbf{Sym}(n)$.
- The smallest support S of x is $S = x$.
- $\mathbf{Orb}_{\mathfrak{A}}(x)$ is the set of all k-element subsets of $[n]$.
- $|\mathbf{Orb}_{\mathfrak{A}}(x)| = \binom{n}{k} \approx n^k$ is polynomial in n iff $k = |S|$ is constant.

Generally: On any structure $\mathfrak{A}$ with $\mathbf{Aut}(\mathfrak{A}) = \mathbf{Sym}(A)$,
$|\mathbf{Orb}(x)|$ is *polynomial* in $|\mathfrak{A}|$ if and only if x has a support of constant size.

Goal: Establish *lower bounds on orbit size* depending on support size.

Example:

- Let $\mathfrak{A} = \{1, 2, \dots, n\}$, and x some subset, e.g. $x = \{1, 2, ..., k\}$.
- We have $\mathbf{Aut}(\mathfrak{A}) = \mathbf{Sym}(n)$.
- The smallest support S of x is $S = x$.
- $\mathbf{Orb}_{\mathfrak{A}}(x)$ is the set of all k-element subsets of $[n]$.
- $|\mathbf{Orb}_{\mathfrak{A}}(x)| = \binom{n}{k} \approx n^k$ is polynomial in n iff $k = |S|$ is constant.

Generally: On any structure $\mathfrak{A}$ with $\mathbf{Aut}(\mathfrak{A}) = \mathbf{Sym}(A)$,
$|\mathbf{Orb}(x)|$ is *polynomial* in $|\mathfrak{A}|$ if and only if x has a support of constant size.

$\implies$ Informally, on structures with $\mathbf{Aut}(\mathfrak{A}) = \mathbf{Sym}(A)$, CPT is not stronger than FPC.

CPT - wrap up

- CPT is a candidate logic for **capturing PTIME**.

CPT - wrap up

- CPT is a candidate logic for **capturing PTIME**.
- CPT can distinguish CFI-graphs if they come with a certain preorder relation; in fact, it distinguishes also the CFI graphs over $\mathbb{Z}_{2^i}$ that are indistinguishable in **rank logic**.

CPT - wrap up

- CPT is a candidate logic for **capturing PTIME**.
- CPT can distinguish CFI-graphs if they come with a certain preorder relation; in fact, it distinguishes also the CFI graphs over $\mathbb{Z}_{2^i}$ that are indistinguishable in **rank logic**.
- **But:** It seems plausible that CPT cannot distinguish unordered CFI graphs, e.g. over n-dimensional hypercubes.

CPT - wrap up

- CPT is a candidate logic for **capturing PTIME**.
- CPT can distinguish CFI-graphs if they come with a certain preorder relation; in fact, it distinguishes also the CFI graphs over $\mathbb{Z}_{2^i}$ that are indistinguishable in **rank logic**.
- **But:** It seems plausible that CPT cannot distinguish unordered CFI graphs, e.g. over n-dimensional hypercubes.
- In particular, the **rank operator** is probably not expressible in CPT (?)

CPT - wrap up

- CPT is a candidate logic for **capturing PTIME**.
- CPT can distinguish CFI-graphs if they come with a certain preorder relation; in fact, it distinguishes also the CFI graphs over $\mathbb{Z}_{2^i}$ that are indistinguishable in **rank logic**.
- **But:** It seems plausible that CPT cannot distinguish unordered CFI graphs, e.g. over n-dimensional hypercubes.
- In particular, the **rank operator** is probably not expressible in CPT (?)
- **Open problem:** Separate CPT from PTIME.

Proof complexity and finite model theory

Problem

Input: *A set* $\mathcal{F} = \{\varphi_1(\vec{x}), ...\varphi_m(\vec{x})\}$ *of propositional formulas.*

Question: *Is there a* $\{0, 1\}$*-assignment to the variables that satisfies all formulas?*

Problem

Input: *A set* $\mathcal{F} = \{\varphi_1(\vec{x}), ...\varphi_m(\vec{x})\}$ *of propositional formulas.*

Question: *Is there a* $\{0, 1\}$*-assignment to the variables that satisfies all formulas?*

- **Propositional proof systems** provide efficiently *verifiable certificates* for the *non-existence of solutions*.

Problem

Input: *A set* $\mathcal{F} = \{\varphi_1(\vec{x}), ...\varphi_m(\vec{x})\}$ *of propositional formulas.*

Question: *Is there a* $\{0, 1\}$*-assignment to the variables that satisfies all formulas?*

- **Propositional proof systems** provide efficiently *verifiable certificates* for the *non-existence of solutions*.
- If every certificate has polynomial size, then $\mathrm{co}\text{-}\mathrm{NP} = \mathrm{NP}$.

Problem

Input: *A set* $\mathcal{F} = \{\varphi_1(\vec{x}), ...\varphi_m(\vec{x})\}$ *of propositional formulas.*

Question: *Is there a* $\{0,1\}$*-assignment to the variables that satisfies all formulas?*

- **Propositional proof systems** provide efficiently *verifiable certificates* for the *non-existence of solutions*.
- If every certificate has polynomial size, then $\mathrm{co\text{-}NP} = \mathrm{NP}$.
- **Proof complexity** seeks to establish proof size lower bounds against stronger and stronger proof systems, towards $\mathrm{co\text{-}NP} \neq \mathrm{NP}$.

Motivation: Transfer lower-bound techniques from finite model theory to proof complexity.

Problem

Input: *A set* $\mathcal{F} = \{f_1(\vec{x}), ...f_m(\vec{x})\}$ *of polynomials in* $\mathbb{F}[X]$.

Question: *Is there a solution, i.e. an assignment* $s \colon X \to \mathbb{F}$ *that is a common zero?*

Problem

Input: *A set* $\mathcal{F} = \{f_1(\vec{x}), ...f_m(\vec{x})\}$ *of polynomials in* $\mathbb{F}[X]$.

Question: *Is there a solution, i.e. an assignment* $s\colon X \to \mathbb{F}$ *that is a common zero?*

Typical certificate via **Hilbert's Nullstellensatz**: $\mathcal{F}$ is unsat iff there exist $g_1, ...g_m \in \mathbb{F}[X]$ such that $\sum_{i\in[m]} g_i \cdot f_i = 1$.

Definition (Grochow, Pitassi; 2016)

An **IPS certificate** of unsatisfiability of $\mathcal{F} = \{f_1(\vec{x}), \dots, f_m(\vec{x})\} \subseteq \mathbb{F}[X]$ is a polynomial $C(\vec{x}, y_1, \dots, y_m)$ such that:

1. $C(\vec{x}, \vec{0}) = 0$.
2. $C(\vec{x}, \vec{f}) = 1$.

An **IPS refutation** of $\mathcal{F}$ is an *algebraic circuit* that represents $C(\vec{x}, \vec{y})$.
The *size* of a refutation is the number of gates in the circuit.

Definition (Grochow, Pitassi; 2016)

An **IPS certificate** of unsatisfiability of $\mathcal{F} = \{f_1(\vec{x}), ..., f_m(\vec{x})\} \subseteq \mathbb{F}[X]$ is a polynomial $C(\vec{x}, y_1, ..., y_m)$ such that:

1. $C(\vec{x}, \vec{0}) = 0$.
2. $C(\vec{x}, \vec{f}) = 1$.

An **IPS refutation** of $\mathcal{F}$ is an *algebraic circuit* that represents $C(\vec{x}, \vec{y})$.
The *size* of a refutation is the number of gates in the circuit.

To make IPS "*isomorphism-invariant*", we only allow circuits that are symmetric under the symmetries of $\mathcal{F}$.

Theorem (Dawar, Grädel, Kullmann, P., 2025)

Let $G \not\cong H$, $k \in \mathbb{N}$.

- *G and H k-**WL-distinguishable** $\Leftrightarrow$ there is a poly-size proof of non-isomorphism in deg_k sym-IPS.*
- *G and H **CPT-distinguishable** $\Rightarrow$ there is a poly-size proof of non-isomorphism in sym-IPS (possibly of unbounded degree).*

Logics with witnessed choice operators

- **Observation:** Suppose $C \subseteq A^k$ is an orbit of $\mathfrak{A}$, i.e. there is a tuple $\bar{a} \in A^k$ such that

$$C = \{\pi(\bar{a}) \mid \pi \in \mathbf{Aut}(\mathfrak{A})\}.$$

Then the computation outcome will be the same for every possible choice from C.

How to allow choices without breaking isomorphism-invariance

- **Observation:** Suppose $C \subseteq A^k$ is an orbit of $\mathfrak{A}$, i.e. there is a tuple $\bar{a} \in A^k$ such that

$$C = \{\pi(\bar{a}) \mid \pi \in \textbf{Aut}(\mathfrak{A})\}.$$

  Then the computation outcome will be the same for every possible choice from C.
- $\implies$ we can allow choices from choice sets that are orbits.

How to allow choices without breaking isomorphism-invariance

- **Observation:** Suppose $C \subseteq A^k$ is an orbit of $\mathfrak{A}$, i.e. there is a tuple $\bar{a} \in A^k$ such that

$$C = \{\pi(\bar{a}) \mid \pi \in \mathbf{Aut}(\mathfrak{A})\}.$$

  Then the computation outcome will be the same for every possible choice from C.
- $\implies$ we can allow choices from choice sets that are orbits.
- Choice sets can be defined via formulas.

How to allow choices without breaking isomorphism-invariance

- **Observation:** Suppose $C \subseteq A^k$ is an orbit of $\mathfrak{A}$, i.e. there is a tuple $\bar{a} \in A^k$ such that

$$C = \{\pi(\bar{a}) \mid \pi \in \mathbf{Aut}(\mathfrak{A})\}.$$

  Then the computation outcome will be the same for every possible choice from C.
- $\implies$ we can allow choices from choice sets that are orbits.
- Choice sets can be defined via formulas.
- **Problem:** If C is not an orbit, then the computation should be aborted because the choice would break isomorphism-invariance. How does the program know if C is an orbit?

How to allow choices without breaking isomorphism-invariance

- **Observation:** Suppose $C \subseteq A^k$ is an orbit of $\mathfrak{A}$, i.e. there is a tuple $\bar{a} \in A^k$ such that

$$C = \{\pi(\bar{a}) \mid \pi \in \mathbf{Aut}(\mathfrak{A})\}.$$

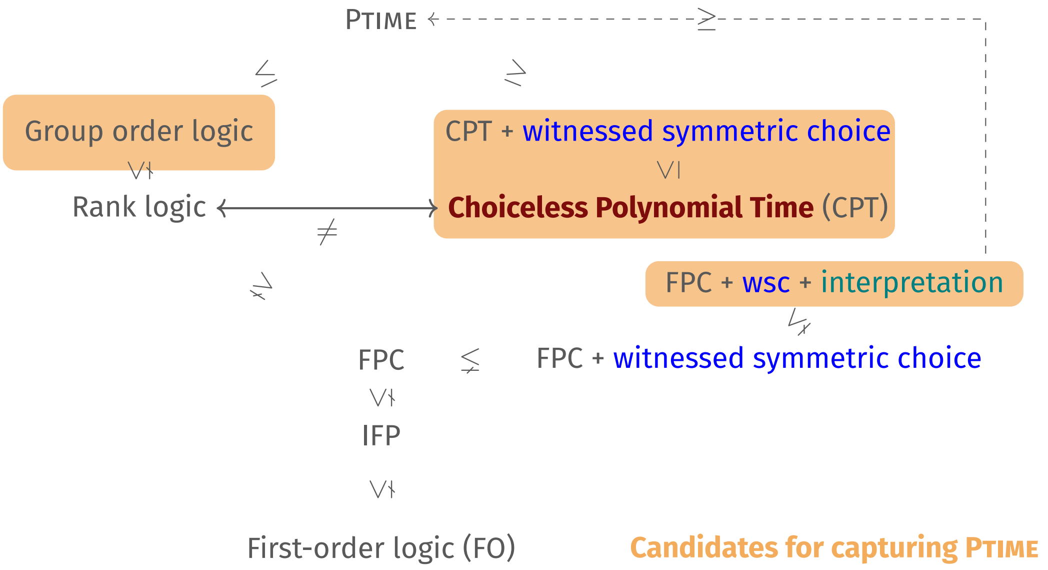
  Then the computation outcome will be the same for every possible choice from C.
- $\implies$ we can allow choices from choice sets that are orbits.
- Choice sets can be defined via formulas.
- **Problem:** If C is not an orbit, then the computation should be aborted because the choice would break isomorphism-invariance. How does the program know if C is an orbit?
- **Solution:** Not only C must be defined by a formula, but also the automorphisms witnessing the fact that C is an orbit.

CPT with witnessed symmetric choice

Theorem (Lichter, Schweitzer, 2024)

CPT with witnessed symmetric choice captures PTIME on every class of structures on which it defines isomorphism.

Landscape of polynomial time logics



[1] A. Blass, Y. Gurevich, and S. Shelah. **"On polynomial time computation over unordered structures".** In: The Journal of Symbolic Logic 67.3 (2002), pp. 1093–1125.

[2] Andreas Blass, Yuri Gurevich, and Saharon Shelah. **"Choiceless polynomial time".** In: Annals of Pure and Applied Logic 100.1-3 (1999), pp. 141–187.

[3] Anuj Dawar and David Richerby. **"A fixed-point logic with symmetric choice".** In: International Workshop on Computer Science Logic. Springer. 2003, pp. 169–182.

[4] Anuj Dawar, David Richerby, and Benjamin Rossman. **"Choiceless Polynomial Time, Counting and the Cai–Fürer–Immerman graphs".** In: Annals of Pure and Applied Logic 152.1-3 (2008), pp. 31–50.

[5] Anuj Dawar et al. **Symmetric Proofs in the Ideal Proof System.** 2025. arXiv: `2504.16820 [cs.LO]`. https://arxiv.org/abs/2504.16820.

[6] F. Gire and H.K. Hoang. **“An Extension of Fixpoint Logic with a Symmetry-Based Choice Construct”.** In: Information and Computation 144.1 (1998), pp. 40–65. ISSN: 0890-5401. https://doi.org/10.1006/inco.1998.2712.

[7] Erich Grädel and Martin Grohe. **“Is Polynomial Time Choiceless?”** In: Fields of Logic and Computation II. Springer, 2015, pp. 193–209.

[8] Erich Grädel et al. **“Characterising Choiceless Polynomial Time with First-Order Interpretations”.** In: 30th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2015, Kyoto, Japan, July 6-10, 2015. IEEE Computer Society, 2015, pp. 677–688. 10.1109/LICS.2015.68. https://doi.org/10.1109/LICS.2015.68.

[9] Joshua A. Grochow and Toniann Pitassi. **“Circuit Complexity, Proof Complexity, and Polynomial Identity Testing: The Ideal Proof System”.** In: J. ACM 65.6 (2018), 37:1–37:59. 10.1145/3230742. https://doi.org/10.1145/3230742.

[10] Moritz Lichter. **“Witnessed Symmetric Choice and Interpretations in Fixed-Point Logic with Counting”.** In: 50th International Colloquium on Automata, Languages, and Programming (ICALP 2023). Ed. by Kousha Etessami, Uriel Feige, and Gabriele Puppis. 261. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023, 133:1–133:20. ISBN: 978-3-95977-278-5. 10.4230/LIPIcs.ICALP.2023.133. https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ICALP.2023.133.

[11] Moritz Lichter and Pascal Schweitzer. **“Choiceless Polynomial Time with Witnessed Symmetric Choice”.** In: J. ACM 71.2 (Apr. 2024). ISSN: 0004-5411. 10.1145/3648104. https://doi.org/10.1145/3648104.

[12] Benedikt Pago. **“Choiceless Computation and Symmetry: Limitations of Definability”.** In: 29th EACSL Annual Conference on Computer Science Logic (CSL 2021). 183. Leibniz International Proceedings in Informatics (LIPIcs). 2021, 33:1–33:21. 10.4230/LIPIcs.CSL.2021.33. https://drops.dagstuhl.de/opus/volltexte/2021/13467.

[13] Wied Pakusa, Svenja Schalthöfer, and Erkal Selman. **“Definability of Cai-Fürer-Immerman problems in Choiceless Polynomial Time”.** In: ACM Transactions on Computational Logic (TOCL) 19.2 (2018), pp. 1–27. 10.1145/3154456.

[14] Benjamin Rossman. **“Choiceless Computation and Symmetry”.** In: Fields of Logic and Computation. Springer, 2010, pp. 565–580.

[15] Faried Abu Zaid et al. **“Choiceless Polynomial Time on structures with small Abelian colour classes”.** In: International Symposium on Mathematical Foundations of Computer Science. Springer. 2014, pp. 50–62.