The Role of Permutation Groups in the Search for a Logic for Polynomial Time.

Public Defense

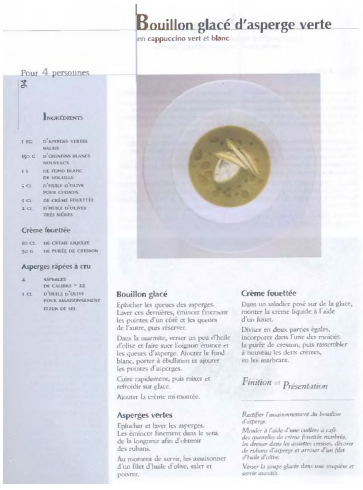
Anatole Dahan

IMJ-PRG
Université Paris-Cité

Supervisors: Arnaud Durand & Luc Segoufin

July 1, 2025

Procedural techniques



Bouillon glacé d'asperge verte
en cappuccino vert et blanc

Pour 4 personnes

Ingrédients

1 kg d'asperges vertes moyennes
150 g d'oignons blancs nouveaux
1 l de fond blanc de volaille
5 cl d'huile d'olive pour cuisson
5 cl de crème fouettée
2 cl d'huile d'olives très mûres

Crème fouettée
10 cl de crème liquide
50 g de purée de cresson

Asperges râpées à cru
4 asperges de calibre 28
1 cl d'huile d'olive pour assaisonnement
Fleur de sel

Bouillon glacé
Éplucher les queues des asperges. Laver ces dernières, émincer finement les pointes d'un côté et les queues de l'autre, puis réserver.
Dans la marmite, verser un peu d'huile d'olive et faire suer l'oignon émincé et les queues d'asperge. Ajouter le fond blanc, porter à ébullition et ajouter les pointes d'asperges.
Cuire rapidement, puis mixer et refroidir sur glace.
Ajouter la crème mi-montée.

Asperges vertes
Éplucher et livrer les asperges. Les émincer finement dans le sens de la longueur afin d'obtenir des rubans.
Au moment de servir, les assaisonner d'un filet d'huile d'olive, saler et poivrer.

Crème fouettée
Dans un saladier posé sur de la glace, monter la crème liquide à l'aide d'un fouet.
Diviser en deux parties égales, incorporer dans l'une des moitiés la purée de cresson, puis remonter à nouveau les deux crèmes, en les marbrant.

Finition et Présentation

Rectifier l'assaisonnement du bouillon d'asperge.
Monter à l'aide d'une cuillère à café des quenelles de crème fouettée marbrée, les disposer dans les assiettes creuses, décorer de rubans d'asperge et arroser d'un filet d'huile d'olive.
Verser le potage glacé dans une soupière et servir aussitôt.

Grand Livre de Cuisine, Alain Ducasse

The sum of length, width, and diagonal is 1 and 5 is the area.
Multiply length, width, and diagonal times length, width, and diagonal.
Multiply the area by 2.
Subtract the products and multiply what is left by one-half.
By what should the sum of length, width, and diagonal be multiplied to obtain this product?
The diagonal is the factor.

$$ax^2 + bx + c = 0$$

$$\iff \boxed{x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}}$$

Ancient Babylonian Algorithms, Donald Knuth

20^{th} century: computation models

Emergence of various equivalent computation models:

- Recursive functions
- Turing Machines
- λ-calculus
- ...

With various notions of resources:

- Time
- Space
- ...

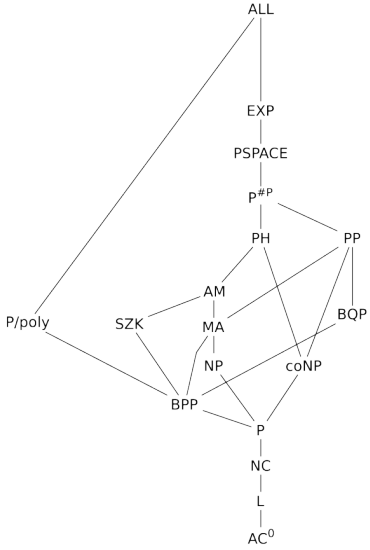
20^{th} century: computation models

Emergence of various equivalent computation models:

- Recursive functions
- Turing Machines
- λ-calculus
- ...

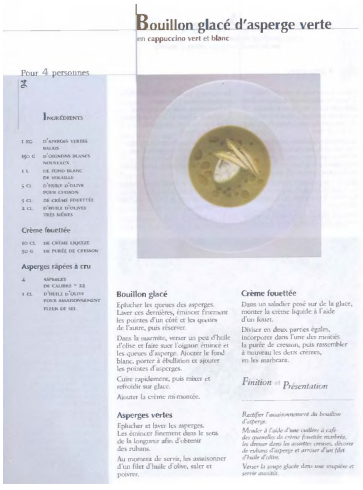
With various notions of resources:

- Time
- Space
- ...



Complexity Zoo

Symmetries in computations



Bouillon glacé d'asperge verte
en cappuccino vert et blanc

Pour 4 personnes

Ingrédients

1 kg d'asperges vertes locales
90 cl de grenouilles blanches nouvelles
1 l de fond blanc de volaille
5 cl d'huile d'olive pour cuisson
5 cl de crème fouettée
2 cl d'huile d'olives très mûres

Crème fouettée
60 cl de crème liquide
50 g de purée de cresson

Asperges râpées à cru
4 asperges de calibre 22
1 cl d'huile d'olive pour assaisonnement
Fleur de sel

Bouillon glacé
Éplucher les queues des asperges. Laver ces dernières, émincer finement les pointes d'un côté et les queues de l'autre, puis réserver.
Dans la marmite, verser un peu d'huile d'olive et faire suer l'oignon émincé et les queues d'asperge. Ajouter le fond blanc, porter à ébullition et ajouter les pointes d'asperges.
Cuire rapidement, puis mixer et refroidir sur glace.
Ajouter la crème montée.

Asperges vertes
Éplucher et laver les asperges. Les émincer finement dans le sens de la longueur afin d'obtenir des rubans.
Au moment de servir, les assaisonner d'un filet d'huile d'olive, saler et poivrer.

Crème fouettée
Dans un saladier posé sur de la glace, monter la crème liquide à l'aide d'un fouet.
Diviser en deux parties égales, incorporer dans l'une des moitiés la purée de cresson, puis remonter à nouveau les deux crèmes, en les marbrant.

Finition et Présentation

Rectifier l'assaisonnement du bouillon d'asperge.
Monter à l'aide d'une cuillère à café des quenelles de crème fouettée marbrée, les disposer dans les assiettes creuses, décorer de rubans d'asperge et arroser d'un filet d'huile d'olive.
Verser la soupe glacée dans une soupière et servir aussitôt.

Grand Livre de Cuisine, Alain Ducasse

Ancient Babylonian Algorithms, Donald Knuth

The sum of length, width, and diagonal is 1 and 5 is the area.
Multiply length, width, and diagonal times length, width, and diagonal.
Multiply the area by 2.
Subtract the products and multiply what is left by one-half.
By what should the sum of length, width, and diagonal be multiplied to obtain this product?
The diagonal is the factor.

$$ax^2 + bx + c = 0$$

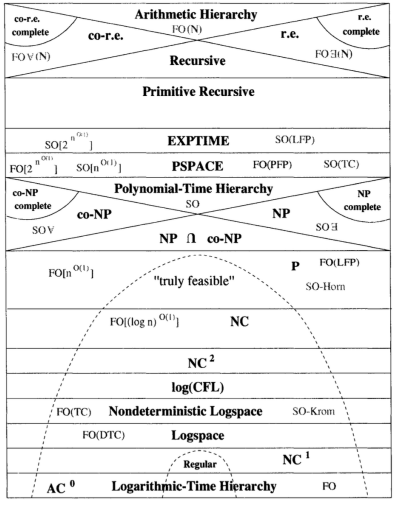
$$\iff \boxed{x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}}$$

Descriptive Complexity

- *Machine-free* models of computation *over structures*: logics
- Capture results
- Example: Fagin theorem NP $=$ SO$\exists$

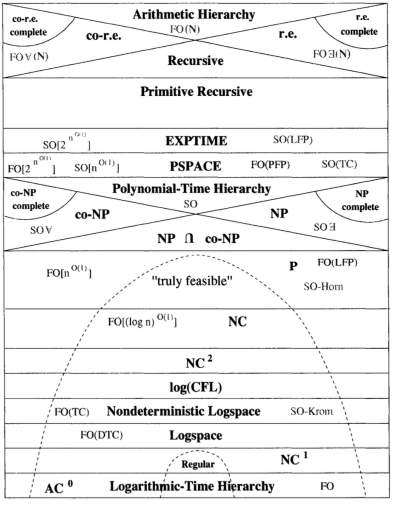
Descriptive Complexity

- *Machine-free* models of computation *over structures*: logics
- Capture results
- Example: Fagin theorem NP = SO∃



Descriptive Complexity

- *Machine-free* models of computation *over structures*: logics
- Capture results
- Example: Fagin theorem NP = SO∃
- Logic for P ?



The Immerman-Vardi theorem

Over ordered structures, $\mathsf{FO} + \mathrm{lfp} = \mathsf{P}$.

- Allows recursion.
- lfp: least fixed point. Given $\varphi(X, \vec{x})$,

$$(\mathrm{lfp}_{X,\vec{x}}\varphi)(\mathfrak{A}) = \underbrace{\varphi^{\mathfrak{A}}(\varphi^{\mathfrak{A}}(\dots\varphi^{\mathfrak{A}}}_{\text{until fixpoint reached}}(\emptyset)\dots))$$

- With $\leq$, arithmetic and definition of encoding $\rightsquigarrow$ simulation of poly-time TM.

The Immerman-Vardi theorem

Over ordered structures, FO + lfp = P.

- Allows recursion.
- lfp: least fixed point. Given $\varphi(X, \vec{x})$,

$$(\mathrm{lfp}_{X,\vec{x}}\varphi)(\mathfrak{A}) = \underbrace{\varphi^{\mathfrak{A}}(\varphi^{\mathfrak{A}}(\ldots\varphi^{\mathfrak{A}}}_{\text{until fixpoint reached}}(\emptyset)\ldots))$$

- With $\leq$, arithmetic and definition of encoding $\rightsquigarrow$ simulation of poly-time TM.

Back and forth

- Over unordered structures, FP does not express EVEN.
- Counting extension of FP: FPC.
  - Formulas evaluated over $\mathfrak{A} + (\{0, 1, \ldots, |A|\}, \leq)$
  - Add counting terms : $(\#x.\varphi(x)) > (\#y.\psi(y))$
  - All P-arithmetic operations are definable (Immerman-Vardi theorem).
- Cai-Fürer-Immerman (CFI): FPC $<$ P.

Back and forth

- [Daw08]: The CFI query reduces to the satisfiability of systems of equations over a field.

$\rightsquigarrow$ FP + rk : extend FP with a linear-algebraic operator.

- If $\varphi_M(\vec{x}, \vec{y}, \lambda)$ defines a matrix M over $\mathfrak{A}$:

$$M^{\mathfrak{A}} := \begin{pmatrix} & \vec{b} & \\ & \vdots & \\ & \varphi_M^{\mathfrak{A}}(\vec{a}, \vec{b}) \cdots\cdots & \end{pmatrix} \vec{a}$$

$(\mathsf{rk}_{\vec{x},\vec{y}}\varphi_M.p)(\mathfrak{A}) = \mathrm{rank}_{\mathbb{F}_p}(M)$.

Back and forth

- [Daw08]: The CFI query reduces to the satisfiability of systems of equations over a field.

$\rightsquigarrow$ FP + rk : extend FP with a linear-algebraic operator.

- If $\varphi_M(\vec{x}, \vec{y}, \lambda)$ defines a matrix M over $\mathfrak{A}$:

$$M^{\mathfrak{A}} := \begin{pmatrix} & \vec{b} & \\ & \vdots & \\ & \varphi_M^{\mathfrak{A}}(\vec{a}, \vec{b}) \cdots\cdots & \end{pmatrix} \vec{a}$$

$(\mathrm{rk}_{\vec{x},\vec{y}}\varphi_M.p)(\mathfrak{A}) = \mathrm{rank}_{\mathbb{F}_p}(M)$.

- Lichter '21: FP + rk $<$ P using a generalization of the CFI-construction.

Link to Graph Isomorphism and Canonisation

Graph Isomorphism (GI)	
Input:	$\mathfrak{G}, \mathfrak{H}$ two graphs
Question:	Whether $\mathfrak{G} \simeq \mathfrak{H}$

- CFI structures are hard instances of Graph Isomorphism: [Tor04]
- Over restricted classes of unordered structures:
  - FP + C = P for bounded tree-width [GM99]
  - FP + C = P when excluded minor [Gro17].
  - Canonisation : if $\mathcal{L} \geq$ FP canonizes $\mathcal{C}$, $\mathcal{L}$ captures P on $\mathcal{C}$.

Definition

A canonisation function: $f(\mathfrak{G}) \simeq \mathfrak{G}$ and if $\mathfrak{G} \simeq \mathfrak{H}$, $f(\mathfrak{G}) = f(\mathfrak{H})$.
In a logic: an interpretation $\mathcal{I} : \mathcal{C} \to \mathcal{C}^{<}$ s.t. $\mathcal{I}(\mathfrak{A})(\text{w.o.} \leq) \simeq \mathfrak{A}$.

A bit of (permutation) group theory

- Group axioms: binary operation,
  - associative
  - neutral
  - inverses
- $\mathrm{Sym}(D) := \{f : D \to D \text{ bijections } \}$ with composition.
- (Finite) Permutation group: $G \leq \mathrm{Sym}(D)$ for some (finite) D.
- Cayley theorem: any group is isomorphic to a permutation group.
- For $S \subseteq G$, $\langle S \rangle \leq G$ smallest subgroup of G containing S.

The role of permutation groups in Graph Isomorphism

Graph Automorphism problem ($\mathsf{GA}_{\mathcal{C}}$)	
Input:	$\mathcal{G} \in \mathcal{C}$
Output:	$S \subseteq \mathrm{Sym}(V_{\mathcal{G}})$ s.t. $\langle S \rangle = \mathrm{Aut}(\mathcal{G}) := \{\sigma \in \mathrm{Sym}(V_{\mathcal{G}}) \mid E_{\mathcal{G}}^{\sigma} = E_{\mathcal{G}}\}$

- If $\mathcal{C}$ union-closed, $\mathsf{GI}_{\mathcal{C}} \leq_{\mathsf{P}} \mathsf{GA}_{\mathcal{C}}$
- Many results for Graph Iso and/or Canonisation on restricted classes of graphs [Bab79, Luk82, BL83, Bab16]. Primitive operations:

Perm. Group Membership Problem	
Input:	$S \subseteq \mathrm{Sym}(D)$ and $\sigma \in \mathrm{Sym}(D)$
Question:	Does $\sigma \in \langle S \rangle$?

$\leq_{\mathsf{L}}^{T}$

Perm. Group Order Problem	
Input:	$S \subseteq \mathrm{Sym}(D)$
Output:	$\lvert\langle S \rangle\rvert$

Both are in P: Schreier-Sims algorithm.

The role of permutation groups in Graph Isomorphism

	$\mathrm{GA}_{\mathcal{C}}$)
Input:	$\mathcal{G} \in \mathcal{C}$
Output:	$S \subseteq \mathrm{Sym}(V_{\mathcal{G}})$ s.t. $\langle S \rangle = \mathrm{Aut}(\mathcal{G}) := \{\sigma \in \mathrm{Sym}(V_{\mathcal{G}}) \mid E_{\mathcal{G}}^{\sigma} = E_{\mathcal{G}}\}$

"Laplace theorem": Any group $G \leq \mathrm{Sym}(D)$ has a generating set S of poly-size (in $|D|$).

- If $\mathcal{C}$ union-closed, $\mathrm{GI}_{\mathcal{C}} \leq_{\mathsf{P}} \mathrm{GA}_{\mathcal{C}}$
- Many results for Graph Iso and/or Canonisation on restricted classes of graphs [Bab79, Luk82, BL83, Bab16]. Primitive operations:

Perm. Group Membership Problem	
Input:	$S \subseteq \mathrm{Sym}(D)$ and $\sigma \in \mathrm{Sym}(D)$
Question:	Does $\sigma \in \langle S \rangle$?

$\leq_{\mathsf{L}}^{T}$

Perm. Group Order Problem	
Input:	$S \subseteq \mathrm{Sym}(D)$
Output:	$\lvert\langle S \rangle\rvert$

Both are in P: Schreier-Sims algorithm.

The role of permutation groups in Graph Isomorphism

Graph Automorphism problem ($\mathsf{GA}_{\mathcal{C}}$)	
Input:	$\mathcal{G} \in \mathcal{C}$
Output:	$S \subseteq \mathrm{Sym}(V_{\mathcal{G}})$ s.t. $\langle S \rangle = \mathrm{Aut}(\mathcal{G}) := \{\sigma \in \mathrm{Sym}(V_{\mathcal{G}}) \mid E_{\mathcal{G}}^{\sigma} = E_{\mathcal{G}}\}$

- If $\mathcal{C}$ union-closed, $\mathsf{GI}_{\mathcal{C}} \leq_{\mathsf{P}} \mathsf{GA}_{\mathcal{C}}$
- Many results for Graph Iso and/or Canonisation on restricted classes of graphs [Bab79, Luk82, BL83, Bab16]. Primitive operations:

Perm. Group Membership Problem	
Input:	$S \subseteq \mathrm{Sym}(D)$ and $\sigma \in \mathrm{Sym}(D)$
Question:	Does $\sigma \in \langle S \rangle$?

$\leq_{\mathsf{L}}^{T}$

Perm. Group Order Problem	
Input:	$S \subseteq \mathrm{Sym}(D)$
Output:	$\lvert\langle S \rangle\rvert$

Both are in P: Schreier-Sims algorithm.

The Schreier-Sims algorithm

Idea: break a group into a tower of subgroups

$$G = H_0 \geq H_1 \geq \cdots \geq H_k = 1$$

For each $i < k$, build a *transversal* T_i of H_{i+1} in H_i. (T_i) is a *Strong Generating Set*.

The Schreier-Sims algorithm

Idea: break a group into a tower of subgroups

$$G = H_0 \geq H_1 \geq \cdots \geq H_k = 1$$

For each $i < k$, build a *transversal* T_i of H_{i+1} in H_i. (T_i) is a *Strong Generating Set*.

If $H \leq G, g \in G$, $gH := \{gh, h \in H\}$ is a coset of H in G
A transversal of H in G is a set of coset representaives.

The Schreier-Sims algorithm

Idea: break a group into a tower of subgroups

$$G = H_0 \geq H_1 \geq \cdots \geq H_k = 1$$

For each $i < k$, build a *transversal* T_i of H_{i+1} in H_i. (T_i) is a *Strong Generating Set*.

The Schreier-Sims algorithm

Idea: break a group into a tower of subgroups

$$G = H_0 \geq H_1 \geq \cdots \geq H_k = 1$$

For each $i < k$, build a *transversal* T_i of H_{i+1} in H_i. (T_i) is a *Strong Generating Set*.

- $\rightsquigarrow$ Sequential reduction of ($\sigma \in G$?) to ($\sigma \in H_i$?).
- $\rightsquigarrow$ Any $\sigma \in G$ admits a unique decomposition $t_1 \ldots, t_k$ with $t_i \in T_i$.
- $\rightsquigarrow$ $|G| = |T_0| \cdot |T_1| \ldots |T_{k-1}|$.
- $\rightsquigarrow$ If $K \leq G$ has polynomial index and decidable membership, can compute g.s. for K.

$$G \geq K \geq H_0 \cap K \geq H_1 \cap K \geq \cdots \geq H_k \cap K = 1$$

The Schreier-Sims algorithm

Idea: break a group into a tower of subgroups

$$G = H_0 \geq H_1 \geq \cdots \geq H_k = 1$$

For each $i < k$, build a *transversal* T_i of H_{i+1} in H_i. (T_i) is a *Strong Generating Set*.

- ⇝ Sequential reduction of ($\sigma \in G$?) to ($\sigma \in H_i$?).
- ⇝ Any $\sigma \in G$ admits a unique dec
- ⇝ $|G| = |T_0| \cdot |T_1| \ldots |T_{k-1}|$.
- ⇝ If $K \leq G$ has polynomial index and decidable membership, can compute g.s. for K.

$$|G : K| = |G|/|K| = |\{gK \mid g \in G\}|$$

$$G \geq K \geq H_0 \cap K \geq H_1 \cap K \geq \cdots \geq H_k \cap K = 1$$

The Schreier-Sims algorithm

Idea: break a group into a tower of subgroups

$$G = H_0 \geq H_1 \geq \cdots \geq H_k = 1$$

For each $i < k$, build a *transversal* T_i of H_{i+1} in H_i. (T_i) is a *Strong Generating Set*.

- $\rightsquigarrow$ Sequential reduction of ($\sigma \in G$?) to ($\sigma \in H_i$?).
- $\rightsquigarrow$ Any $\sigma \in G$ admits a unique decomposition $t_1 \ldots, t_k$ with $t_i \in T_i$.
- $\rightsquigarrow$ $|G| = |T_0| \cdot |T_1| \ldots |T_{k-1}|$.
- $\rightsquigarrow$ If $K \leq G$ has polynomial index and decidable membership, can compute g.s. for K.

$$G \geq K \geq H_0 \cap K \geq H_1 \cap K \geq \cdots \geq H_k \cap K = 1$$

The Schreier-Sims algorithm

Idea: break a group into a tower of subgroups

$$G = H_0 \geq H_1 \geq \cdots \geq H_k = 1$$

For each $i < k$, build a *transversal* T_i of H_{i+1} in H_i. (T_i) is a *Strong Generating Set*.

- $\rightsquigarrow$ Sequential reduction of ($\sigma \in G$?) to ($\sigma \in H_i$?).
- $\rightsquigarrow$ Any $\sigma \in G$ admits a unique decomposition $t_1 \ldots, t_k$ with $t_i \in T_i$.
- $\rightsquigarrow$ $|G| = |T_0| \cdot |T_1| \ldots |T_{k-1}|$.
- $\rightsquigarrow$ If $K \leq G$ has polynomial index and decidable membership, can compute g.s. for K.

$$G \geq K \geq H_0 \cap K \geq H_1 \cap K \geq \cdots \geq H_k \cap K = 1$$

- Notion of accessibility

The Schreier-Sims algorithm

Idea: break a group into a tower of subgroups

$$G = H_0 \geq H_1 \geq \cdots \geq H_k = 1$$

For each $i < k$, build a *transversal* T_i of H_{i+1} in H_i. (T_i) is a *Strong Generating Set*.

- For poly-time, we need $k < p(|D|)$, and $\forall i, |H_i : H_{i+1}| < p(|D|)$.
- If $D = \{a_1, \ldots, a_n\}$,

$$H_i := \mathrm{Stab}_{(a_1,\ldots,a_i)}(G)$$

- Without an ordering of D ?

The Schreier-Sims algorithm

Idea: break a group into a tower of subgroups

$$G = H_0 \geq H_1 \geq \cdots \geq H_k = 1$$

For each $i < k$, build a *transversal* T_i of H_{i+1} in H_i. (T_i) is a *Strong Generating Set*.

- For poly-time, we need $k < p(|D$
- If $D = \{a_1, \ldots, a_n\}$,

$\mathrm{Stab}_{(X)}(G) := \{g \in G \mid gx = x \forall x \in X\}$

$$H_i := \mathrm{Stab}_{(a_1,\ldots,a_i)}(G)$$

- Without an ordering of D ?

The Schreier-Sims algorithm

Idea: break a group into a tower of subgroups

$$G = H_0 \geq H_1 \geq \cdots \geq H_k = 1$$

For each $i < k$, build a *transversal* T_i of H_{i+1} in H_i. (T_i) is a *Strong Generating Set*.

- For poly-time, we need $k < p(|D|)$, and $\forall i, |H_i : H_{i+1}| < p(|D|)$.
- If $D = \{a_1, \ldots, a_n\}$,

$$H_i := \mathrm{Stab}_{(a_1,\ldots,a_i)}(G)$$

- Without an ordering of D ?

Contributions

- Study representations of permutation groups in extensions of FO.
  - Unordered sets of generators
  - Ordered sets of generators
  - Abelian groups
- Study of the expressive power of the operation ord : $S \mapsto |\langle S \rangle|$ as an extension of FP.
  - $\mathrm{FP} + \mathrm{ord} > \mathrm{FP} + \mathrm{rk}$

Outline

First approach: definable generating sets

$\mathsf{FP} + \mathsf{rk} \leq \mathsf{FP} + \mathsf{ord}$

$\mathsf{FP} + \mathsf{ord} \not\leq \mathsf{FP} + \mathsf{rk}$

Definable ordered sets of permutations

Plan

First approach: definable generating sets

FP + rk $\leq$ FP + ord

FP + ord $\not\leq$ FP + rk

Definable ordered sets of permutations

Definable sets of permutations

Definition

$\varphi(\vec{s}, \vec{t})$ (with $\mathrm{type}(\vec{s}) = \mathrm{type}(\vec{t}) = T$) *defines* $\sigma \in \mathrm{Sym}(A^T)$ if

$$\forall \vec{b}, \vec{c} \in A^T, \sigma(\vec{b}) = \vec{c} \iff (\mathfrak{A}, \vec{b}, \vec{c}) \models \varphi$$

- $\mathrm{graph}(\sigma) = \varphi(\mathfrak{A})$
- $\sigma = \mathrm{perm}(\varphi(\mathfrak{A}))$

Definition

A formula $\varphi(\vec{p}, \vec{s}, \vec{t})$ defines $S \subseteq \mathrm{Sym}(A^T)$ *binding* $\vec{p}$ if

$$\{\mathrm{perm}(\varphi(\mathfrak{A}, \vec{a})) \mid \vec{a} \in A^{\vec{p}}\} = S.$$

Low-hanging fruits

- FO + tc defines orbits.
- If G and H are $\mathcal{L}$-definable, with $\mathcal{L} \geq \mathrm{FO}$, $\langle G \cup H \rangle$ is $\mathcal{L}$-definable.

Low-hanging fruits

- FO + tc defines orbits.
- If G and H are $\mathcal{L}$-definable, with $\mathcal{L} \geq \mathrm{FO}$, $\langle G \cup H \rangle$ is $\mathcal{L}$-definable.
- Corollary: Membership $\leq^{T}_{\mathrm{FO}}$ Group Order.

Low-hanging fruits

- FO + tc defines orbits.
- If G and H are $\mathcal{L}$-definable, with $\mathcal{L} \geq$ FO, $\langle G \cup H \rangle$ is $\mathcal{L}$-definable.
- Corollary: Membership $\leq^T_{\text{FO}}$ Group Order.
- Turing FO reduction: Add an operator ord

Low-hanging fruits

- FO + tc defines orbits.
- If G and H are $\mathcal{L}$-definable, with $\mathcal{L} \geq \mathrm{FO}$, $\langle G \cup H \rangle$ is $\mathcal{L}$-definable.
- Corollary: Membership $\leq^T_{\mathrm{FO}}$ Group Order.
- Turing FO reduction: Add an operator ord

Definition

$(\mathrm{ord}_{\vec{p}.\vec{s}.\vec{t}}\ \varphi)$ is a *numerical predicate* (of arity $2 \cdot |\vec{s}|$) encoding $|\langle S \rangle|$.

Plan

First approach: definable generating sets

$\mathsf{FP} + \mathsf{rk} \leq \mathsf{FP} + \mathsf{ord}$

$\mathsf{FP} + \mathsf{ord} \not\leq \mathsf{FP} + \mathsf{rk}$

Definable ordered sets of permutations

Outline of the proof

- If $\varphi_M(\vec{x}, \vec{y}, \lambda)$ defines a matrix,

$$M : \mathbb{F}_p^{A^{\vec{x}}} \to \mathbb{F}_p^{A^{\vec{y}}}$$

$$\begin{aligned}(\mathrm{rk}_{\vec{x},\vec{y}}\varphi_M.p) &= \dim(\mathrm{Im}(M)) \\ &= \log_p |\mathrm{Im}(M)|\end{aligned}$$

  where $\mathrm{Im}(M) \leq \mathbb{F}_p^{A^{\vec{y}}}$ additive group.
- $\log_p$ is definable (Immerman-Vardi theorem)
- New goals:
  1. permutation representation of $\mathbb{F}_p^{A^{\vec{y}}}$
  2. define a generating set for $\mathrm{Im}(M) \leq \mathbb{F}_p^{A^{\vec{y}}}$.

Subgoal 1: Permutation representation of $\mathbb{F}_p^{A^{\vec{y}}}$.

- For $p \in \mathbb{N}^*$,

$$\begin{aligned} \mathsf{repr}_p : \mathbb{F}_p &\to S_p \\ k &\mapsto (i \mapsto i + k \mod p) \end{aligned}$$

  is FPC-definable.

- For T_1, T_2,

$$\begin{aligned} \iota_{T_1,T_2} : \mathrm{Sym}(A^{T_2})^{A^{T_1}} &\to \mathrm{Sym}(A^{T_1} \times A^{T_2}) \\ (g_{\vec{a}})_{\vec{a} \in A^{T_1}} &\mapsto ((\vec{a}, \vec{b}) \mapsto (\vec{a}, (g_{\vec{a}}(\vec{b})))) \end{aligned}$$

  is FPC-definable.

- Conclusion: $\iota_{\vec{y},\mu} \circ (\mathsf{repr}_p^{A^{\vec{y}}}) : \mathbb{F}_p^{A^{\vec{y}}} \hookrightarrow \mathrm{Sym}(A^{\vec{y}} \times A^{<})$ is FPC-definable.

Spoiler

Subgoal 2: Generating set for $\mathrm{Im}_{\mathbb{F}_p}(M)$

- $((\mathbb{F}_p)^{A^{\vec{x}}}, +)$ generated by the set B all scalar products of unit vectors.
- $\rightsquigarrow$ $\mathrm{Im}_{\mathbb{F}_p}(M)$ is generated by $\{M \cdot b \mid b \in B\}$.
- Given φ_M,

$$A^{\vec{x}} \times \mathbb{F}_p \to \mathrm{Sym}(A^{\vec{y}} \times A^{<})$$
$$(\vec{a}, \lambda) \mapsto \iota_{\vec{y},\mu} \circ (\mathrm{repr}_p^{A^{\vec{y}}})(M \cdot (\lambda \cdot \hat{e}_{\vec{a}}))$$

FPC-definable. □

Remark

Proof generalizes: FP + ord solves arbitrary systems of linear equations over any abelian group.

Plan

First approach: definable generating sets

$\mathsf{FP} + \mathsf{rk} \leq \mathsf{FP} + \mathsf{ord}$

$\mathsf{FP} + \mathsf{ord} \not\leq \mathsf{FP} + \mathsf{rk}$

Definable ordered sets of permutations

Preliminaries

- Lichter '21: $\mathsf{FP} + \mathsf{rk} < \mathsf{P}$
- Last remark: decision problem separation
- The separating query is defined on a class of structures with Abelian Colours.
- We show $\mathsf{FP} + \mathsf{ord}$ canonises (and thus captures P on) structures with Abelian Colours, $\rightsquigarrow$ $\mathsf{FP} + \mathsf{rk} < \mathsf{FP} + \mathsf{ord}$.

Structures with Abelian Colours

- Structure $\mathfrak{A}$ ordered partition
$$A_1 \preceq A_2 \preceq \cdots \preceq A_m$$
- For each $i \leq m$, abelian group Γ_i acting transitively on A_i

$\rightsquigarrow$ $|\Gamma_i| = |A_i|$: for all $a \in A_i$, $\gamma \mapsto \gamma(a)$ bijection.

- For each i, $\mathfrak{A}$ equipped with an enumeration $(\gamma^i_j)_{j<|A_i|}$ of Γ_i.

Structures with Abelian Colours

- Structure $\mathfrak{A}$ ordered partition

$$A_1 \preceq A_2$$

- For each $i \leq m$, abelian group Γ_i acting transitively on A_i.

$\rightsquigarrow$ $|\Gamma_i| = |A_i|$: for all $a \in A_i$, $\gamma \mapsto \gamma(a)$ bijection.

- For each i, $\mathfrak{A}$ equipped with an enumeration $(\gamma_j^i)_{j<|A_i|}$ of Γ_i.

If $G \leq \mathrm{Sym}(D)$ is abelian transitive, it is *regular*:
$\forall g \in G, (\exists x, gx = x) \implies g = 1$

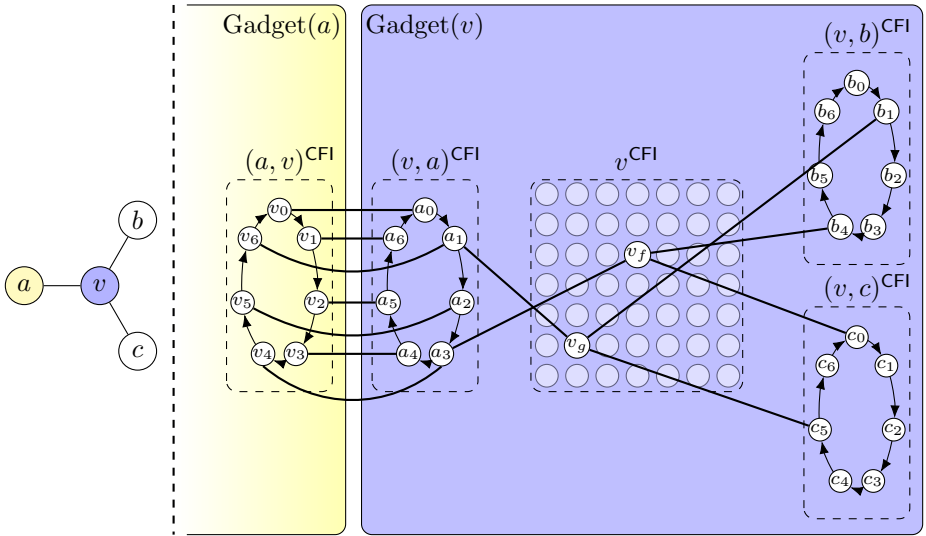
Structures with Abelian Colours

- Structure $\mathfrak{A}$ ordered partition
$$A_1 \preceq A_2 \preceq \cdots \preceq A_m$$
- For each $i \leq m$, abelian group Γ_i acting transitively on A_i

$\rightsquigarrow$ $|\Gamma_i| = |A_i|$: for all $a \in A_i$, $\gamma \mapsto \gamma(a)$ bijection.

- For each i, $\mathfrak{A}$ equipped with an enumeration $(\gamma^i_j)_{j<|A_i|}$ of Γ_i.

Archetypal abelian colours structures: CFI



Structures with Abelian Colours

- Structure $\mathfrak{A}$ ordered partition
$$A_1 \preceq A_2 \preceq \cdots \preceq A_m$$
- For each $i \leq m$, abelian group Γ_i acting transitively on A_i

$\rightsquigarrow$ $|\Gamma_i| = |A_i|$: for all $a \in A_i$, $\gamma \mapsto \gamma(a)$ bijection.

- For each i, $\mathfrak{A}$ equipped with an enumeration $(\gamma_j^i)_{j<|A_i|}$ of Γ_i.

$\rightsquigarrow$ $\mathcal{O}(A_i)$ (unordered) family of $|A_i|$ definable orderings of A_i.

- $\mathcal{O}(A) := \prod_{i=1}^{m} \mathcal{O}(A_i)$.

The canonisation algorithm

Algorithm 1: Canonisation procedure

Input : $\mathfrak{A} = (A, E, \prec, \Phi)$ a structure with Abelian colours
Output: A numerical relation $E^<$ isomorphic to E

$\mathcal{C} := \mathcal{O}(A)$;

for $(i, j) \in [m]^2$ **do**
    find $E^<_{i,j}$, the smallest lexicographical encoding of E on $A_i \cup A_j$ compatible with $\mathcal{C}$;
    $\mathcal{C} \leftarrow \{\sigma \in \mathcal{C}, \mathrm{enc}(E, \sigma)_{\restriction A_i \times A_j} = E^<_{i,j}\}$;

return $E^< := \bigcup_{i,j} E^<_{i,j}$

(same algorithm as [Pak15, chapter 6])

The canonisation algorithm

Algorithm 1: Canonisation procedure

Input : $\mathfrak{A} = (A, E, \prec, \Phi)$ a structure with Abelian colours
Output: A numerical relation $E^<$ isomorphic to E

$\mathcal{C} := \mathcal{O}(A)$;

for $(i, j) \in [m]^2$ **do**
    find $E^<_{i,j}$, the smallest lexicographical encoding of E on $A_i \cup A_j$ compatible with $\mathcal{C}$;
    $\mathcal{C} \leftarrow \{\sigma \in \mathcal{C}, \mathrm{enc}(E, \sigma)_{\restriction A_i \times A_j} = E^<_{i,j}\}$;

return $E^< := \bigcup_{i,j} E^<_{i,j}$

(same algorithm as [Pak15, chapter 6])

Representing labeling cosets: three challenges

- The usual representation ($\mathcal{C} = \sigma H$) of cosets is not isomorphism-invariant.
- Computation of accessible subgroups is not isomorphism-invariant.

Representing labeling cosets: three challenges

- The usual representation ($\mathcal{C} = \sigma H$) of cosets is not isomorphism-invariant.
- Computation of accessible subgroups is not isomorphism-invariant.

Aparté : Definable generating sets are limited

$$\mathfrak{A}_n := K_{n,n} \quad \text{and} \quad G_n := \mathrm{Aut}(\mathfrak{A}_n)$$

Aparté : Definable generating sets are limited

$$\mathfrak{A}_n := K_{n,n} \quad \text{and} \quad G_n := \mathrm{Aut}(\mathfrak{A}_n)$$

Worse, we can find a (sequence of) structure $\mathfrak{A}$ s.t.:

- A group $\mathbf{G}(\mathfrak{A})$ is (FP-)definable from $\mathfrak{A}$
- A subgroup $\mathbf{H}(\mathfrak{A})$ is accessible from $\mathbf{G}(\mathfrak{A})$
- FP defines a witness of the accessibility of $\mathbf{H}(\mathfrak{A})$ in $\mathbf{G}(\mathfrak{A})$.
- Any symmetric generating set of $\mathbf{H}(\mathfrak{A})$ has exponential size.

Representing labeling cosets: three challenges

- The usual representation ($\mathcal{C} = \sigma H$) of cosets is not isomorphism-invariant.
- Computation of accessible subgroups is not isomorphism-invariant.

Fix: Morphism-definability

Morphism-definability

Definition

R a relation symbol of arity $2k$.

$\varphi_m(R, \vec{x}, \vec{y})$ *defines a morphism* $m : G \to \mathrm{Sym}(A^{|\vec{x}|})$ on $\mathfrak{A}$, where $G \leq \mathrm{Sym}(A^k)$ if, for all $\sigma \in G$,

$$\varphi_m(\mathfrak{A}, R \leftarrow \mathrm{graph}(\sigma)) = \mathrm{graph}(m(\sigma))$$

$H \trianglelefteq G \leq \mathrm{Sym}(A^T)$ is morphism-definable from G in $\mathfrak{A}$ if:

- There is a definable generating set for G in $\mathfrak{A}$
- there is a $\mathcal{L}$-formula φ_m which defines a morphism $m : G \to \mathrm{Sym}(A^{T'})$ on $\mathfrak{A}$ such that $\ker(m) = H$.

Morphism-definability

$m : G \to H, m(g_1 g_2) = m(g_1)m(g_2)$

Definition

R a relation symbol of arity $2k$.

$\varphi_m(R, \vec{x}, \vec{y})$ *defines a morphism* $m : G \to \mathrm{Sym}(A^{|\vec{x}|})$ on $\mathfrak{A}$, where $G \leq \mathrm{Sym}(A^k)$ if, for all $\sigma \in G$,

$$\varphi_m(\mathfrak{A}, R \leftarrow \mathrm{graph}(\sigma)) = \mathrm{graph}(m(\sigma))$$

$H \trianglelefteq G \leq \mathrm{Sym}(A^T)$ is morphism-definable from G in $\mathfrak{A}$ if:

- There is a definable generating set for G in $\mathfrak{A}$
- there is a $\mathcal{L}$-formula φ_m which defines a morphism $m : G \to \mathrm{Sym}(A^{T'})$ on $\mathfrak{A}$ such that $\ker(m) = H$.

Morphism-definability

Definition

R a relation symbol of arity $2k$.

$\varphi_m(R, \vec{x}, \vec{y})$ *defines a morphism* $m : G \to \mathrm{Sym}(A^{|\vec{x}|})$ on $\mathfrak{A}$, where $G \leq \mathrm{Sym}(A^k)$ if, for all $\sigma \in G$,

$$\varphi_m(\mathfrak{A}, R \leftarrow \mathrm{graph}(\sigma)) = \mathrm{graph}(m(\sigma))$$

$H \trianglelefteq G \leq \mathrm{Sym}(A^T)$ is morphism-definable from G in $\mathfrak{A}$ if:

- There is a definable generating set for G in $\mathfrak{A}$
- there is a $\mathcal{L}$-formula φ_m which defines a morphism $m : G \to \mathrm{Sym}(A^{T'})$ on $\mathfrak{A}$ such that $\ker(m) = H$.

Properties of morphism-definability

- If H is morphism-definable from G in $\mathfrak{A}$, then FP $+$ ord defines $|H|$ and membership to H.
- Moreover, the morphism m defining H in G defines a bijection between cosets of H in G and $\mathrm{Im}(m)$.

- If H_1 and H_2 are morphism-definable from G in $\mathfrak{A}$, then so is $H_1 \cap H_2$.

Properties of morphism-definability

- If H is morph… defines $|H|$ and membership to H.
- Moreover, the morphism m defining H in G defines a bijection between cosets of H in G and $\mathrm{Im}(m)$.

First iso. theorem: $G/\ker(m) \simeq \mathrm{im}(m)$

- If H_1 and H_2 are morphism-definable from G in $\mathfrak{A}$, then so is $H_1 \cap H_2$.

Properties of morphism-definability

- If H is morphism-definable from G in $\mathfrak{A}$, then FP $+$ ord defines $|H|$ and membership to H.
- Moreover, the morphism m defining H in G defines a bijection between cosets of H in G and $\mathrm{Im}(m)$.

- If H_1 and H_2 are morphism-definable from G in $\mathfrak{A}$, then so is $H_1 \cap H_2$.

Properties of morphism-definability

- If H is morphism-definable from G in $\mathfrak{A}$, then FP $+$ ord defines $|H|$ and membership to H.
- Moreover, the morphism m defining H in G defines a bijection between cosets of H in G and $\mathrm{Im}(m)$.

$\rightsquigarrow$ solves the coset representation issue

- If H_1 and H_2 are morphism-definable from G in $\mathfrak{A}$, then so is $H_1 \cap H_2$.

Properties of morphism-definability

- If H is morphism-definable from G in $\mathfrak{A}$, then FP $+$ ord defines $|H|$ and membership to H.
- Moreover, the morphism m defining H in G defines a bijection between cosets of H in G and $\mathrm{Im}(m)$.

$\rightsquigarrow$ solves the coset representation issue

- If H_1 and H_2 are morphism-definable from G in $\mathfrak{A}$, then so is $H_1 \cap H_2$.

$\rightsquigarrow$ solves the subgroup computation issue

Representing labeling cosets: three challenges

- The usual representation ($\mathcal{C} = \sigma H$) of cosets is not isomorphism-invariant.
- Computation of accessible subgroups is not isomorphism-invariant.

Fix: Morphism-definability

- When $A \neq A^<$, labeling cosets are not group cosets (i.e. $\mathcal{C} \not\subseteq \mathrm{Sym}(A)$).

Representing labeling cosets: three challenges

- The usual representation ($\mathcal{C} = \sigma H$) of cosets is not isomorphism-invariant.
- Computation of accessible subgroups is not isomorphism-invariant.

Fix: Morphism-definability

- When $A \neq A^<$, labeling cosets are not group cosets (i.e. $\mathcal{C} \not\subseteq \mathrm{Sym}(A)$).

Fix: Labeling group

Labeling group

- Algorithmic perspective:
  - encoding of $\mathfrak{A} \approx$ one fixed $\sigma : A \to A^<$.
  - Represent labeling $\ell : A \to A^<$ by the unique $\tau \in \mathrm{Sym}(A)$ s.t. $\ell = \sigma\tau$.
  - Extends to cosets: $\ell G = \sigma\tau G$, and $\tau G \subseteq \mathrm{Sym}(A)$.
  - Choice of σ not isomorphism-invariant.

Labeling group

- Algorithmic perspective:
  - encoding of $\mathfrak{A} \approx$ one fixed $\sigma : A \to A^<$.
  - Represent labeling $\ell : A \to A^<$ by the unique $\tau \in \mathrm{Sym}(A)$ s.t. $\ell = \sigma\tau$.
  - Extends to cosets: $\ell G = \sigma\tau G$, and $\tau G \subseteq \mathrm{Sym}(A)$.
  - Choice of σ not isomorphism-invariant.
- Unordered setting with Abelian colouring:
  - families $\mathcal{O}(A_i)$ of labelings of A_i indexed by A_i ($a \mapsto \mathsf{map}_a^i$)
  - Represent labeling $\ell : A \to A^<$ by $(\tau_a)_{a \in A}$ s.t.

$$\forall i, \forall a \in A_i, \mathsf{map}_a^i \tau_a = \ell_{\restriction A_i}$$
$$\rightsquigarrow \tau_a = (\mathsf{map}_a^i)^{-1} \ell.$$

  - Yields $\varphi : \mathcal{C}^* \to \mathrm{Sym}(A)^A$. Thus, $\iota \circ \varphi : \mathcal{C}^* \to \mathrm{Sym}(A \times A)$. ($\iota := \iota_{A,A}$)

◂ Definition of ι

Labeling group

- $\exists \mathcal{G} \leq \mathrm{Sym}(A \times A)$ with definable generating set, s.t. $(\iota \circ \varphi)(\mathcal{O}(A)) \subseteq \mathcal{G}$
- $(\iota \circ \varphi)(\mathcal{O}(A))$ is a coset of a morphism-definable subgroup $\Gamma^* \leq \mathcal{G}$.
- Given an encoding of $E \cap (A_i \cap A_j)$, the corresponding labelings in $\mathcal{O}(A)$ form a coset of a morphism-definable subgroup $\mathrm{Aut}(E_{i,j})^*$ of Γ^*

▸ Skip technical details

Labeling group

- $\exists \mathcal{G} \leq \mathrm{Sym}(A \times A)$ with definable generating set, s.t. $(\iota \circ \varphi)(\mathcal{O}(A)) \subseteq \mathcal{G}$
- $(\iota \circ \varphi)(\mathcal{O}(A))$ is a coset of a morphism-definable subgroup $\Gamma^* \leq \mathcal{G}$.
- Given an encoding of $E \cap (A_i \cap A_j)$, the corresponding labelings in $\mathcal{O}(A)$ form a coset of a morphism-definable subgroup $\mathrm{Aut}(E_{i,j})^*$ of Γ^*

Requires Γ_i abelian for all i.

Skip technical details

Technical definitions: $\mathcal{G}$

$\mathcal{O}(A)$ elements out of reach. But, any $\pi \in \mathcal{O}(A)$ is a product of elements of $\mathcal{O}(A_i)$ which are definable.

$$\mathcal{G} := \left\langle \bigcup_i \iota \circ \varphi_i(\mathcal{O}(A_i)) \right\rangle$$

$$\text{where } \varphi_i(\sigma)_a := \begin{cases} \varphi(\sigma)_a & \text{if } a \in A_i \\ 1 & \text{otherwise} \end{cases}$$

Obviously, $\varphi(\sigma) = \prod \varphi_i(\sigma_{\restriction A_i})$.

Technical definitions: Γ^*, $\mathrm{Aut}(E_{i,j})^*$

Let $\Gamma := \prod_i \Gamma_i$

- $\varphi : \mathcal{O}(A) \to \mathrm{Sym}(A)^A$ is compatible with

$$\begin{aligned} \psi : \Gamma &\to \mathrm{Sym}(A)^A \\ \gamma &\mapsto (\gamma_{\restriction A_{i(a)}})_{a \in A} \end{aligned}$$

  (in the sense that $\varphi(\pi\gamma) = \varphi(\pi)\psi(\gamma)$).
- $\mathcal{O}(A) = \pi\Gamma$ for some (any) $\pi \in \mathcal{O}(A)$.

$\rightsquigarrow$ $\varphi(\mathcal{O}(A)) = \varphi(\pi)\psi(\Gamma)$.

$$\begin{aligned} \Gamma^* &:= \psi(\Gamma) \\ \mathrm{Aut}(E_{i,j}) &:= \{\sigma \in \Gamma_i\Gamma_j \mid \forall a \in A_i, b \in A_j, E(a,b) \iff E(\sigma(a), \sigma b)\} \\ \mathrm{Aut}(E_{i,j})^* &:= \psi(\mathrm{Aut}(E_{i,j})) \end{aligned}$$

Plan

First approach: definable generating sets

$\mathrm{FP} + \mathrm{rk} \leq \mathrm{FP} + \mathrm{ord}$

$\mathrm{FP} + \mathrm{ord} \not\leq \mathrm{FP} + \mathrm{rk}$

Definable ordered sets of permutations

Motivation

- Abelian colours: *abelian* and *ordered* groups.
- Γ_i ordered $\rightsquigarrow$ unordered orderings $\mathcal{O}(A_i)$
- Hence the need to leverage structural properties of Γ_i (commutativity).

Motivation

- Abelian colours: *abelian* and *ordered* groups.
- Γ_i ordered $\rightsquigarrow$ unordered orderings $\mathcal{O}(A_i)$
- Hence the need to leverage structural properties of Γ_i (commutativity).

$\rightsquigarrow$ If we have ordered generators but no structural property ?

Results

- If FP + ord defines an *ordered* generating set of permutations for G on $\mathfrak{A}$, and $H \leq G$ is FP + ord-definably accessible in G, then FP + ord defines a generating set for H.
- if G is abelian, FPC suffices.

How

Partial simulation of the Schreier-Sims:

$$G = H_0 \geq H_1 \geq \cdots \geq H_k = H \geq \mathrm{Stab}_{(a_1)}(H) \geq \mathrm{Stab}_{(a_1,a_2)}(H) \cdots \geq \mathrm{Stab}_{(a_1,\ldots,a_n)}(H) = 1$$

Algorithm 2: Sifting procedure

Input: $(T_i)_{i<k+n}$, $\sigma \in \mathrm{Sym}(D)$
Output: Does $\sigma \in G$?
for $i = 0$ *to* $k + n$ **do**
  if $\forall \tau \in T_i, \tau^{-1}\sigma \notin H_i$ **then**
    return (σ, i);
  else
    $\sigma \leftarrow \tau^{-1}\sigma$;
return $\top$;

Algorithm 3: Construction procedure

Input: $S \subseteq \mathrm{Sym}(D)$ s.t. $\langle S \rangle = G$
Output: $(T_i)_{i<k+n}$ a S.G.S. for G
$\ell := S$;
$T_i := \emptyset$ for all $i < k + n$;
for $\sigma \in \ell$ **do**
  if $\mathrm{sift}(\sigma) = (\tau, i)$ **then**
    T_i.add(τ)
    ℓ.add$(\tau \cdot \rho, \rho \cdot \tau)$ for $\rho \in \bigcup_j T_j$;

How

Partial simulation of the Schreier-Sims:

$$G = H_0 \geq H_1 \geq \cdots \geq H_k = H \geq \mathrm{Stab}_{(a_1)}(H) \geq \mathrm{Stab}_{(a_1,a_2)}(H) \cdots \geq \mathrm{Stab}_{(a_1,\ldots,a_n)}(H) = 1$$

Algorithm 2: Sifting procedure

Input: $(T_i)_{i<k}$, R,σ
Output: Does $\sigma \in G$?
for $i = 0$ *to* k **do**
  if $\forall \tau \in T_i, \tau^{-1}\sigma \notin H_i$ **then**
    return (σ, i);
  else
    $\sigma \leftarrow \tau^{-1}\sigma$;
return *Whether* $\sigma \in \langle R \rangle$;

Algorithm 3: Construction procedure

Input: $S \subseteq \mathrm{Sym}(D)$ s.t. $\langle S \rangle = G$
Output: $(T_i)_{i<k}$ and R
$\ell := S$;
$R := \emptyset$ and $T_i := \emptyset$ for all $i < k$;
for $\sigma \in \ell$ **do**
  if $\mathrm{sift}(\sigma) = (\tau, i)$ **then**
    if $i = k$ **then**
      R.add(τ);
    else
      T_i.add(τ);
    ℓ.add$(\tau \cdot \rho, \rho \cdot \tau)$ for $\rho \in \bigcup_j T_j \cup R$;

If S is ordered, these procedures are FP + ord definable

Results

- If FP + ord defines an *ordered* generating set of permutations for G on $\mathfrak{A}$, and $H \leq G$ is FP + ord-definably accessible in G, then FP + ord defines a generating set for H.
- if G is abelian, FPC suffices.

Results

- If FP + ord defines an *ordered* generating set of permutations for G on $\mathfrak{A}$, and $H \leq G$ is FP + ord-definably accessible in G, then FP + ord defines a generating set for H.
- if G is abelian, FPC suffices.
- FPC defines the automorphism groups of the structures (with abelian colours) which separate FP + rk from P.

Conclusion

- Leverage more general structural properties of groups.
- How far goes the canonisation power of FP $+$ ord?
- FO $+$ ord?
- CPT $+$ ord?
- Inexpressibility results?

References I

László Babai, *Monte-Carlo algorithms in graph isomorphism testing*, Université de Montréal Technical Report, DMS (1979), no. 79-10.

_____, *Graph Isomorphism in Quasipolynomial Time*, January 2016.

László Babai and Eugene M. Luks, *Canonical labeling of graphs*, Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing - STOC '83, ACM Press, 1983, pp. 171–183.

Anuj Dawar, *On the Descriptive Complexity of Linear Algebra*, Logic, Language, Information and Computation (Wilfrid Hodges and Ruy De Queiroz, eds.), vol. 5110, Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, pp. 17–25.

Martin Grohe and Julian Mariño, *Definability and Descriptive Complexity on Databases of Bounded Tree-Width*, Database Theory — ICDT'99 (Gerhard Goos, Juris Hartmanis, Jan Van Leeuwen, Catriel Beeri, and Peter Buneman, eds.), vol. 1540, Springer Berlin Heidelberg, Berlin, Heidelberg, 1999, pp. 70–82.

Martin Grohe, *Descriptive Complexity, Canonisation, and Definable Graph Structure Theory*, 1 ed., Cambridge University Press, August 2017.

References II


Eugene M. Luks, *Isomorphism of graphs of bounded valence can be tested in polynomial time*, Journal of Computer and System Sciences **25** (1982), no. 1, 42–65.


Wied Pakusa, *Linear equation systems and the search for a logical characterisation of polynomial time*, Ph.D. thesis, RWTH Aachen University, 2015.


Jacobo Torán, *On the Hardness of Graph Isomorphism*, SIAM Journal on Computing **33** (2004), no. 5, 1093–1108.