Successor-Invariant First-Order Logic on Classes of Bounded Degree

Julien Grange
Université de Rennes 1, Inria

02/25/2021

Table of contents

Introduction

Databases stored on disk come with an order

- Useful to scan the database
- Query results shouldn't depend upon it

Example of query using the successor relation

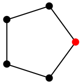
Example (Successor-dependent query)

$\mathcal{Q}_{\text{red}}$:= "The first node is red"

Example of query using the successor relation

Example (Successor-dependent query)

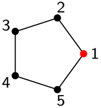
$\mathcal{Q}_{\text{red}} :=$ "The first node is red"



Example of query using the successor relation

Example (Successor-dependent query)

$\mathcal{Q}_{\text{red}} :=$ "The first node is red"

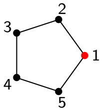


$\models \quad \mathcal{Q}_{\text{red}}$

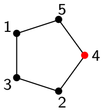
Example of query using the successor relation

Example (Successor-dependent query)

$\mathcal{Q}_{\mathsf{red}} :=$ "The first node is red"



$\models \quad \mathcal{Q}_{\mathsf{red}}$



$\models \quad \neg\mathcal{Q}_{\mathsf{red}}$

Example of query using the successor relation

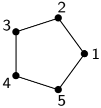
Example (Successor-invariant query)

$\mathcal{Q}_{\text{odd}} :=$"The last node belongs to P, where $P := \{1, 3, 5, \cdots\}$"

Example of query using the successor relation

Example (Successor-invariant query)

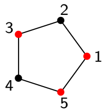
$\mathcal{Q}_{\text{odd}}$:="The last node belongs to P, where $P := \{1, 3, 5, \cdots\}$"



Example of query using the successor relation

Example (Successor-invariant query)

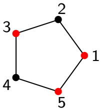
$\mathcal{Q}_{\text{odd}}$:=" The last node belongs to P, where $P := \{1, 3, 5, \cdots\}$"



Example of query using the successor relation

Example (Successor-invariant query)

$\mathcal{Q}_{\text{odd}}$:="The last node belongs to P, where $P := \{1, 3, 5, \cdots\}$"

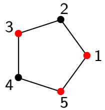


$\models \quad \mathcal{Q}_{\text{odd}}$

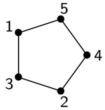
Example of query using the successor relation

Example (Successor-invariant query)

$\mathcal{Q}_{\mathsf{odd}}$:=" The last node belongs to P, where $P := \{1, 3, 5, \cdots\}$"



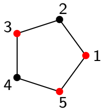
$\models \quad \mathcal{Q}_{\mathsf{odd}}$



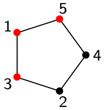
Example of query using the successor relation

Example (Successor-invariant query)

$\mathcal{Q}_{\text{odd}}$:=" The last node belongs to P, where $P := \{1, 3, 5, \cdots\}$"



$\models \quad \mathcal{Q}_{\text{odd}}$

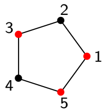


$\models \quad \mathcal{Q}_{\text{odd}}$

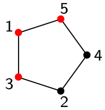
Example of query using the successor relation

Example (Successor-invariant query)

$\mathcal{Q}_{\text{odd}}$:=" The last node belongs to P, where $P := \{1, 3, 5, \cdots\}$"



$\models \quad \mathcal{Q}_{\text{odd}}$



$\models \quad \mathcal{Q}_{\text{odd}}$

$\mathcal{Q}_{\text{odd}}$ defines odd structures.

Definition of Succ-inv FO

$\varphi \in \mathrm{FO}(\Sigma, S)$ is **successor-invariant** over a finite Σ-structure $\mathcal{A}$ if

$$\forall S_1, S_2, \qquad (\mathcal{A}, S_1) \models \varphi \quad \leftrightarrow \quad (\mathcal{A}, S_2) \models \varphi$$

Definition of Succ-inv FO

$\varphi \in \mathrm{FO}(\Sigma, S)$ is **successor-invariant** over a finite Σ-structure $\mathcal{A}$ if

$$\forall S_1, S_2, \qquad (\mathcal{A}, S_1) \models \varphi \quad \leftrightarrow \quad (\mathcal{A}, S_2) \models \varphi$$

Definition (Successor-Invariant First-Order Logic)

Succ-inv $\mathrm{FO} := \{\varphi : \varphi$ is successor-invariant over every finite $\mathcal{A}\}$

Definition of Succ-inv FO

$\varphi \in \mathrm{FO}(\Sigma, S)$ is **successor-invariant** over a finite Σ-structure $\mathcal{A}$ if

$$\forall S_1, S_2, \qquad (\mathcal{A}, S_1) \models \varphi \quad \leftrightarrow \quad (\mathcal{A}, S_2) \models \varphi$$

Definition (Successor-Invariant First-Order Logic)

Succ-inv $\mathrm{FO} := \{\varphi : \varphi$ is successor-invariant over every finite $\mathcal{A}\}$

Succ-inv FO doesn't have a recursive syntax.

Known results

Immerman-Vardi (1982):

- $\mathrm{PTIME}\,(=\!<\!\text{-inv LFP}\,) = \text{Succ-inv LFP}$

Known results

Immerman-Vardi (1982):

- $\mathrm{PTIME}\,(=<\text{-inv LFP}) = \text{Succ-inv LFP}$

Rossman (2007):

- $\mathrm{FO} \subsetneq \text{Succ-inv FO}$

Known results

Immerman-Vardi (1982):

- $\mathrm{PTIME}\,(=<\text{-inv LFP}\,)\;=\;\text{Succ-inv LFP}$

Rossman (2007):

- $\mathrm{FO} \subsetneq \text{ Succ-inv FO}$

Benedikt, Segoufin (2009):

- $\text{Succ-inv FO} = \mathrm{FO}$ on trees

Known results

Immerman-Vardi (1982):

- $\mathrm{PTIME}\,(=<\text{-inv } \mathrm{LFP}\,) = \text{Succ-inv } \mathrm{LFP}$

Rossman (2007):

- $\mathrm{FO} \subsetneq \text{ Succ-inv } \mathrm{FO}$

Benedikt, Segoufin (2009):

- Succ-inv $\mathrm{FO} = \mathrm{FO}$ on trees
- Succ-inv $\mathrm{FO} \subseteq \mathrm{MSO}$ on
  - graphs of bounded degree
  - graphs of bounded treewidth

Succ-inv FO collapses to FO when the degree is bounded

Theorem

Let $\mathcal{C}_d$ be a class of degree at most d.

$$\text{Succ-inv FO} = \mathrm{FO} \textit{ on } \mathcal{C}_d$$

Succ-inv FO collapses to FO when the degree is bounded

Theorem

Let $\mathcal{C}_d$ be a class of degree at most d.

$$\text{Succ-inv } \mathrm{FO} = \mathrm{FO} \textit{ on } \mathcal{C}_d$$

$$\varphi \in \text{ Succ-inv } \mathrm{FO}$$

$$\downarrow$$

$$\exists \psi \in \mathrm{FO}, \quad \psi \leftrightarrow \varphi \text{ on } \mathcal{C}_d$$

Proof of the collapse

$$\begin{array}{ccc} \mathcal{G}_1 & \equiv_{f(k)} & \mathcal{G}_2 \\ & \downarrow & \\ (\mathcal{G}_1, S_1) & \equiv_k & (\mathcal{G}_2, S_2) \end{array}$$

Proof of the collapse

$$\begin{array}{ccc} \mathcal{G}_1 & \equiv_{f(k)} & \mathcal{G}_2 \\ & \downarrow & \\ (\mathcal{G}_1, S_1) & \equiv_k & (\mathcal{G}_2, S_2) \end{array}$$

$\varphi \in$ Succ-inv FO,
of quantifier rank k

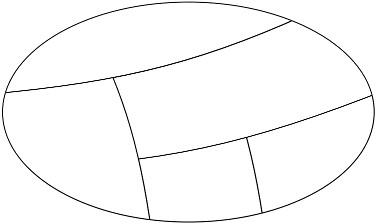
Proof of the collapse

$$\mathcal{G}_1 \quad \equiv_{f(k)} \quad \mathcal{G}_2$$

$$\downarrow$$

$$(\mathcal{G}_1, S_1) \quad \equiv_k \quad (\mathcal{G}_2, S_2)$$

$\varphi \in$ Succ-inv FO,
of quantifier rank k



Equivalence classes for $\equiv_{f(k)}$
over $\mathcal{C}_d$

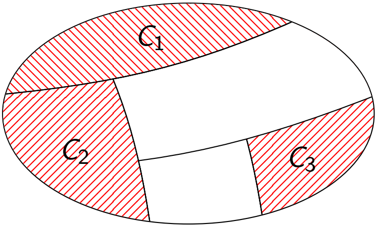
Proof of the collapse

$$\mathcal{G}_1 \qquad \equiv_{f(k)} \qquad \mathcal{G}_2$$

$$\downarrow$$

$$(\mathcal{G}_1, S_1) \qquad \equiv_k \qquad (\mathcal{G}_2, S_2)$$

$\varphi \in$ Succ-inv FO,
of quantifier rank k



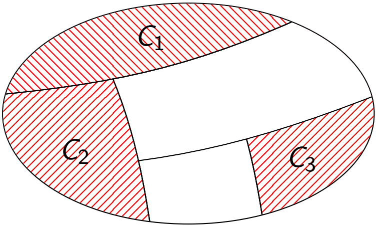
Equivalence classes for $\equiv_{f(k)}$ over $\mathcal{C}_d$

Proof of the collapse

$$\mathcal{G}_1 \quad \equiv_{f(k)} \quad \mathcal{G}_2$$
$$\downarrow$$
$$(\mathcal{G}_1, S_1) \quad \equiv_k \quad (\mathcal{G}_2, S_2)$$

$\varphi \in$ Succ-inv FO,
of quantifier rank k

$$\varphi \quad \leftrightarrow \quad \underbrace{\psi_{C_1} \vee \psi_{C_2} \vee \psi_{C_3}}_{\in \text{ FO}} \text{ on } \mathcal{C}_d$$



Equivalence classes for $\equiv_{f(k)}$ over $\mathcal{C}_d$

Proof of the collapse

$$\begin{array}{ccc} \mathcal{G}_1 & \equiv_{f(k)} & \mathcal{G}_2 \\ & \downarrow & \\ (\mathcal{G}_1, S_1) & \equiv_k & (\mathcal{G}_2, S_2) \end{array}$$

Proof of the collapse

$$\begin{array}{ccc} \mathcal{G}_1 & \equiv_{f(k)} & \mathcal{G}_2 \\ & \downarrow & \\ (\mathcal{G}_1, S_1) & \equiv_k & (\mathcal{G}_2, S_2) \end{array}$$

Because the degree is bounded, it amounts to

Proof of the collapse

$$\begin{array}{ccc} \mathcal{G}_1 & \equiv_{f(k)} & \mathcal{G}_2 \\ & \downarrow & \\ (\mathcal{G}_1, S_1) & \equiv_k & (\mathcal{G}_2, S_2) \end{array}$$

Because the degree is bounded, it amounts to

Hypothesis: $\mathcal{G}_1, \mathcal{G}_2$ have the same number of each neighborhood type (up to some threshold)

Proof of the collapse

$$\begin{array}{ccc} \mathcal{G}_1 & \equiv_{f(k)} & \mathcal{G}_2 \\ & \downarrow & \\ (\mathcal{G}_1, S_1) & \equiv_k & (\mathcal{G}_2, S_2) \end{array}$$

Because the degree is bounded, it amounts to

Hypothesis: $\mathcal{G}_1, \mathcal{G}_2$ have the same number of each neighborhood type (up to some threshold)

Goal: Construct S_1, S_2 such that $(\mathcal{G}_1, S_1), (\mathcal{G}_2, S_2)$ have the same number of each neighborhood type (up to some threshold)

Neighborhood types with many occurrences in $\mathcal{G}_1$

Neighborhood types with few occurrences in $\mathcal{G}_1$

Neighborhood types with many occurrences in $\mathcal{G}_1$

- have many occurrences in $\mathcal{G}_2$

Neighborhood types with few occurrences in $\mathcal{G}_1$

Neighborhood types with many occurrences in $\mathcal{G}_1$

- have many occurrences in $\mathcal{G}_2$
- can translate to their **fractal** version in $(\mathcal{G}_1, S_1)$ and $(\mathcal{G}_2, S_2)$

Neighborhood types with few occurrences in $\mathcal{G}_1$

Neighborhood types with many occurrences in $\mathcal{G}_1$

- have many occurrences in $\mathcal{G}_2$
- can translate to their **fractal** version in $(\mathcal{G}_1, S_1)$ and $(\mathcal{G}_2, S_2)$

Neighborhood types with few occurrences in $\mathcal{G}_1$

- have the same number of occurrences in $\mathcal{G}_2$

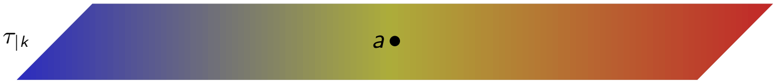
Neighborhood types with many occurrences in $\mathcal{G}_1$

- have many occurrences in $\mathcal{G}_2$
- can translate to their **fractal** version in $(\mathcal{G}_1, S_1)$ and $(\mathcal{G}_2, S_2)$

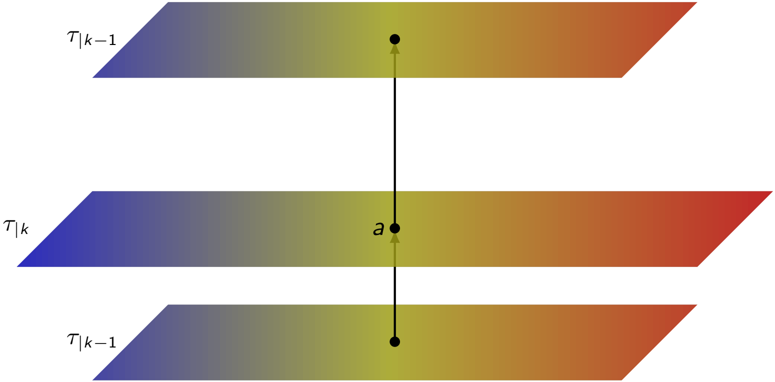
Neighborhood types with few occurrences in $\mathcal{G}_1$

- have the same number of occurrences in $\mathcal{G}_2$
- can be embedded among frequent neighborhood types in $(\mathcal{G}_1, S_1)$ and $(\mathcal{G}_2, S_2)$

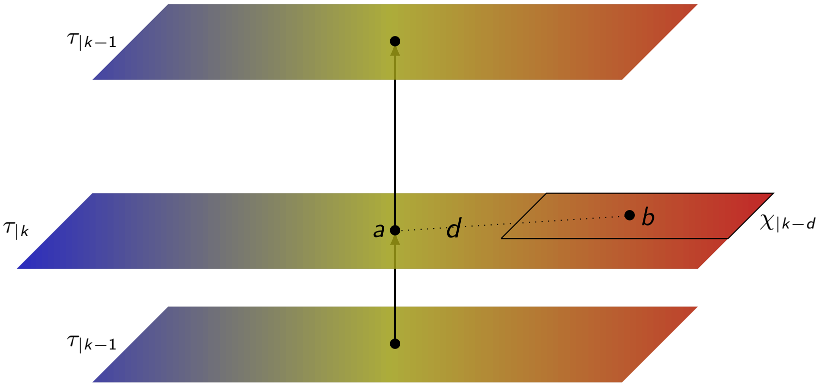
Fractal types



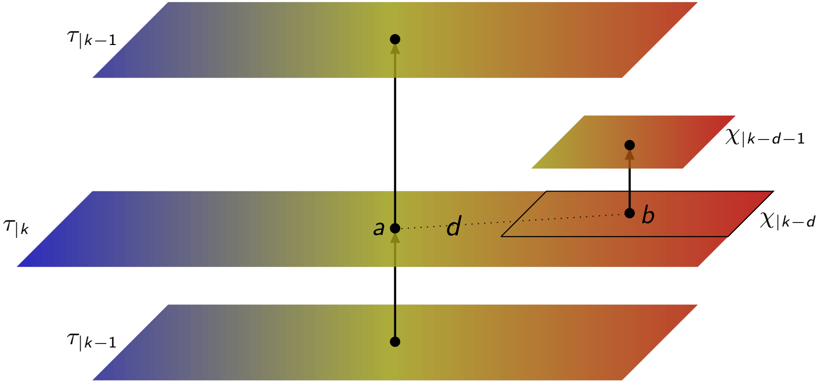
Fractal types



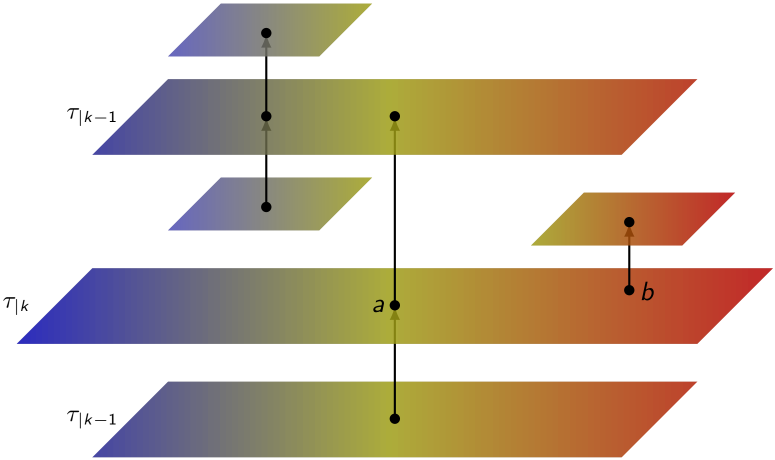
Fractal types

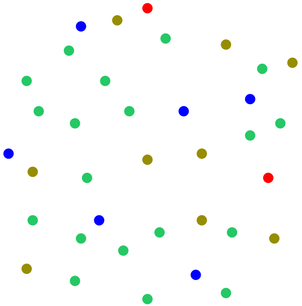


Fractal types



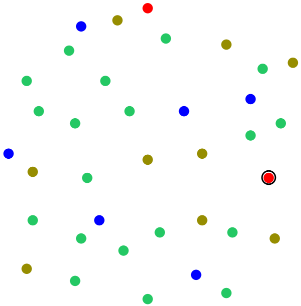
Fractal types

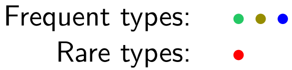


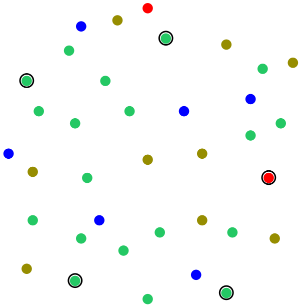


Frequent types:

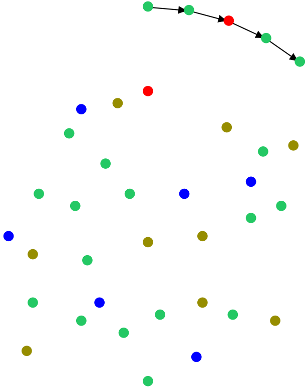
Rare types:




Frequent types:
Rare types:

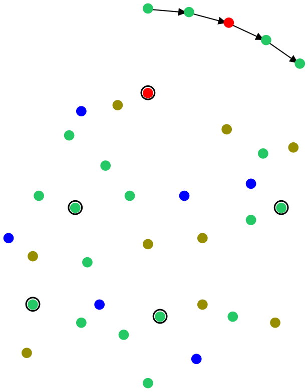


Frequent types:
Rare types:



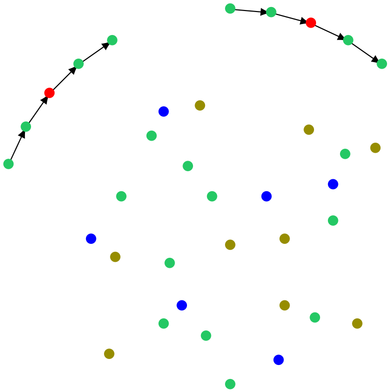
Frequent types:

Rare types:

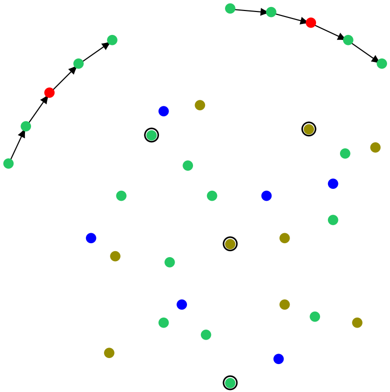


Frequent types:

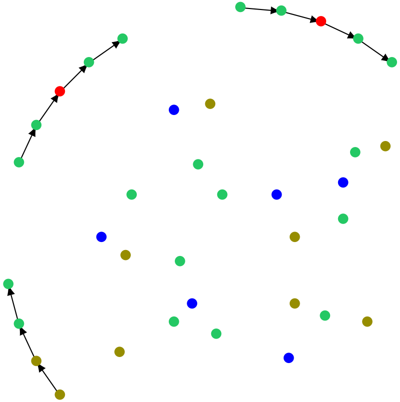
Rare types:



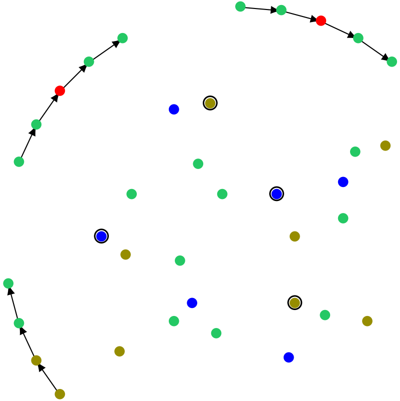
Frequent types:
Rare types:



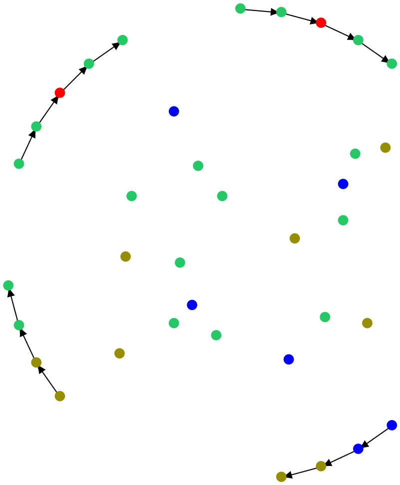
Frequent types:
Rare types:



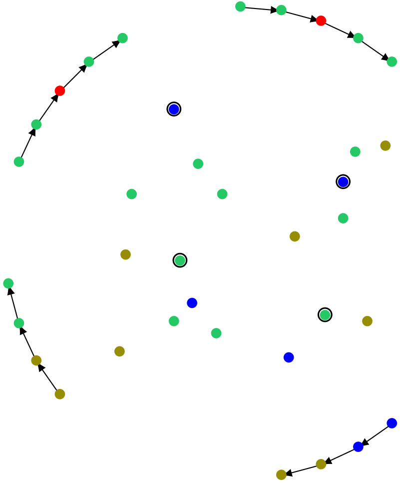
Frequent types:
Rare types:



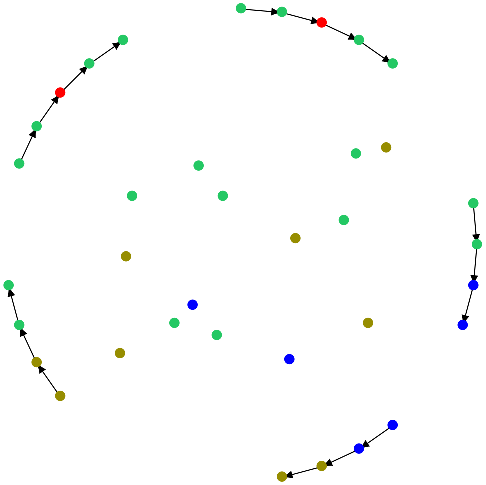
Frequent types:
Rare types:



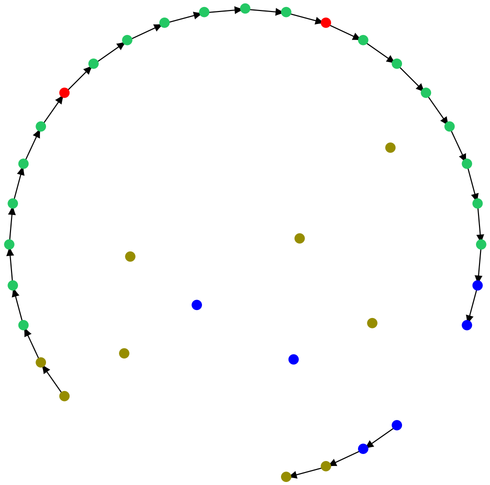
Frequent types:
Rare types:



Frequent types:
Rare types:

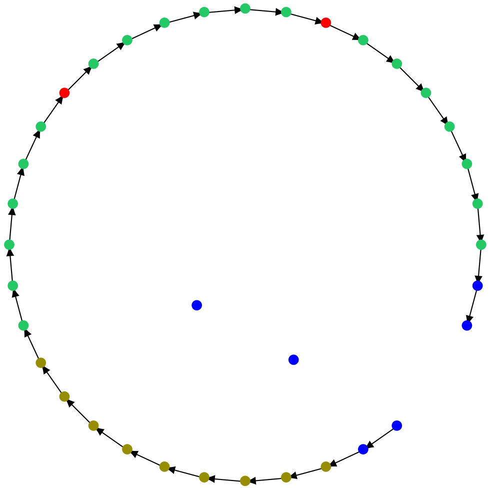


Frequent types:
Rare types:



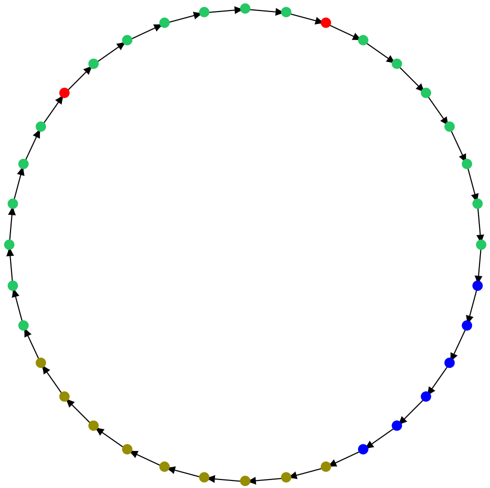
Frequent types:

Rare types:

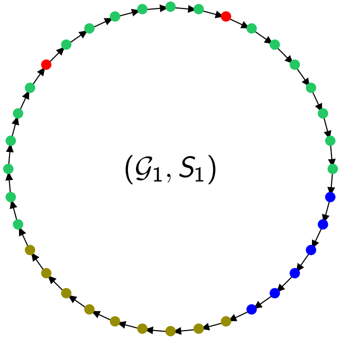


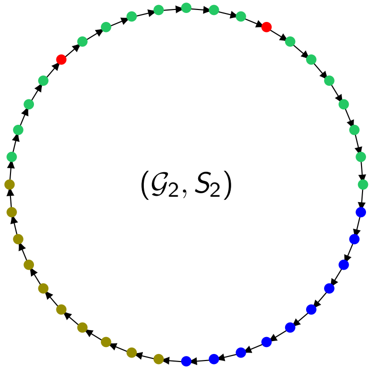
Frequent types:

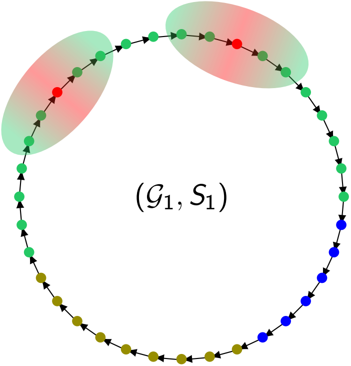
Rare types:

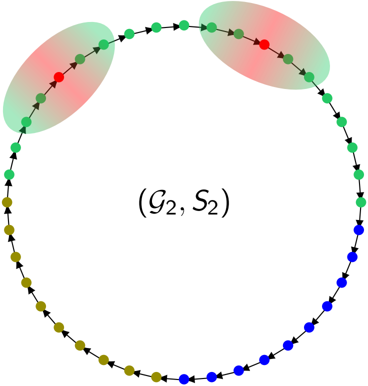


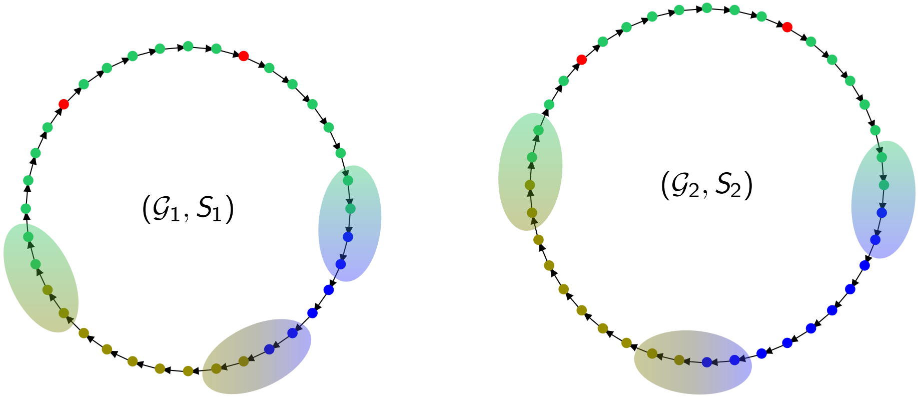
Frequent types:
Rare types:

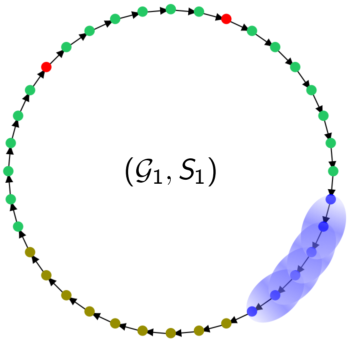

$(\mathcal{G}_1, S_1)$

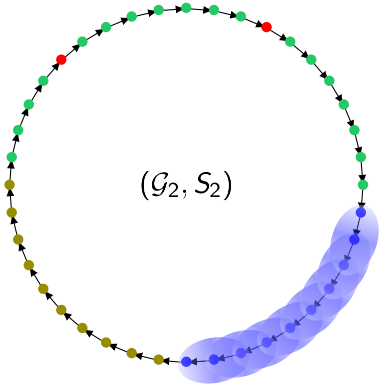

$(\mathcal{G}_2, S_2)$

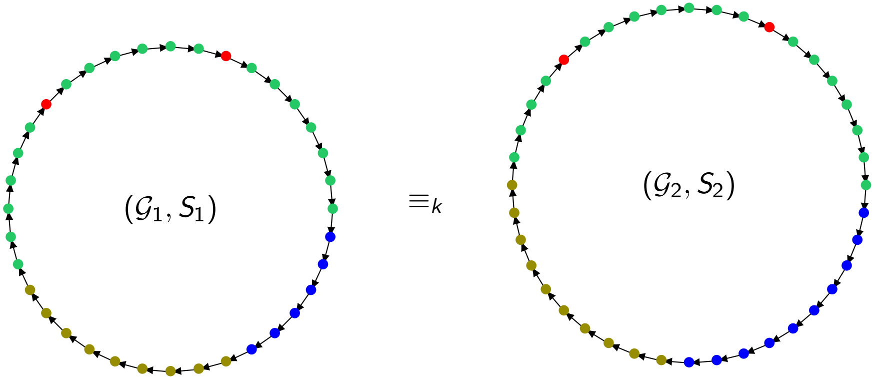

$(\mathcal{G}_1, \mathcal{S}_1)$


$(\mathcal{G}_2, \mathcal{S}_2)$


$(\mathcal{G}_1, S_1)$
$(\mathcal{G}_2, S_2)$


$(\mathcal{G}_1, S_1)$


$(\mathcal{G}_2, S_2)$



$\square$

Conclusion

Theorem

Succ-inv $\mathrm{FO} = \mathrm{FO}$ *on classes of bounded degree*

Follow-up questions

Succ-inv $\mathrm{FO} = \mathrm{FO}$ on other sparse classes?
$<$-inv $\mathrm{FO} = \mathrm{FO}$ on classes of bounded degree?
What about Succ-inv $\mathcal{L}$, where $\mathcal{L}$ is a fragment of FO (e.g. CQ)?