Grant Agreement no. 681988, CSP-Infinity

DFG Research Training Group 1763

Grant Agreement no. 771779, DeciGUT

Datalog-Expressibility for Monadic and Guarded Second-Order Logic

Manuel Bodirsky[1], Simon Knäuer[1], and Sebastian Rudolph[2]

[1]TU Dresden, Institut für Algebra
[2]TU Dresden, Computational Logic Group, Germany

48th International Colloquium on Automata, Languages and Programming
July 12-16, 2021

Introduction 1

Monadic Second-Order Logic:

Introduction 1

Monadic Second-Order Logic:

- Büchi's theorem, 1960: MSO on words = regular languages.

Introduction 1

Monadic Second-Order Logic:

- Büchi's theorem, 1960: MSO on words = regular languages.
- Courcelle's theorem, 1990: MSO properties can be decided in linear time on graphs of bounded treewidth.

Introduction 1

Monadic Second-Order Logic:

- Büchi's theorem, 1960: MSO on words = regular languages.
- Courcelle's theorem, 1990: MSO properties can be decided in linear time on graphs of bounded treewidth.

Datalog:

Introduction 1

Monadic Second-Order Logic:

- Büchi's theorem, 1960: MSO on words = regular languages.
- Courcelle's theorem, 1990: MSO properties can be decided in linear time on graphs of bounded treewidth.

Datalog:

- "Prolog without function symbols."

Introduction 1

Monadic Second-Order Logic:

- Büchi's theorem, 1960: MSO on words = regular languages.
- Courcelle's theorem, 1990: MSO properties can be decided in linear time on graphs of bounded treewidth.

Datalog:

- "Prolog without function symbols."
- "Local consistency algorithms."

Introduction 1

Monadic Second-Order Logic:

- Büchi's theorem, 1960: MSO on words = regular languages.
- Courcelle's theorem, 1990: MSO properties can be decided in linear time on graphs of bounded treewidth.

Datalog:

- "Prolog without function symbols."
- "Local consistency algorithms."
- Query answering in database theory.

Introduction 1

Monadic Second-Order Logic:

- Büchi's theorem, 1960: MSO on words = regular languages.
- Courcelle's theorem, 1990: MSO properties can be decided in linear time on graphs of bounded treewidth.

Datalog:

- "Prolog without function symbols."
- "Local consistency algorithms."
- Query answering in database theory.
- Fragment of second-order logic.

Introduction 1

Monadic Second-Order Logic:

- Büchi's theorem, 1960: MSO on words = regular languages.
- Courcelle's theorem, 1990: MSO properties can be decided in linear time on graphs of bounded treewidth.

Datalog:

- "Prolog without function symbols."
- "Local consistency algorithms."
- Query answering in database theory.
- Fragment of second-order logic.

Wish: Combination of the good computational properties of expressibility in Datalog and of expressibility in MSO.

Which computational problems are expressible in MSO
AND can be solved by a Datalog program?

Introduction 2

Contributions:

Introduction 2

Contributions:

- Description of MSO $\cap$ Datalog in terms of Constraint Satisfaction Problems.

Introduction 2

Contributions:

- Description of MSO $\cap$ Datalog in terms of Constraint Satisfaction Problems.
- A necessary condition whether a given Datalog program is in MSO.

Introduction 2

Contributions:

- Description of MSO $\cap$ Datalog in terms of Constraint Satisfaction Problems.
- A necessary condition whether a given Datalog program is in MSO.
- Pebble game characterization of MSO $\cap$ Datalog. Not in this talk!

Introduction 2

Contributions:

- Description of MSO $\cap$ Datalog in terms of Constraint Satisfaction Problems.
- A necessary condition whether a given Datalog program is in MSO.
- Pebble game characterization of MSO $\cap$ Datalog. Not in this talk!
- All results also hold more generally for GSO (Guarded Second-Order Logic) instead of MSO. Also not in this talk!

Monadic Second-Order Logic

τ: finite relational signature.

Monadic Second-Order Logic

τ: finite relational signature.

Second-order logic: extension of first-order logic by (existential and universal) quantification over relations.

Monadic Second-Order Logic

τ: finite relational signature.

Second-order logic: extension of first-order logic by (existential and universal) quantification over relations.

Monadic second-order logic (MSO): all quantified relations are unary.

Monadic Second-Order Logic

τ: finite relational signature.

Second-order logic: extension of first-order logic by (existential and universal) quantification over relations.

Monadic second-order logic (MSO): all quantified relations are unary.

Monadic second-order τ-sentence: all first-order variables are quantified, τ symbols are not quantified.

Monadic Second-Order Logic

τ: finite relational signature.

Second-order logic: extension of first-order logic by (existential and universal) quantification over relations.

Monadic second-order logic (MSO): all quantified relations are unary.

Monadic second-order τ-sentence: all first-order variables are quantified, τ symbols are not quantified.

Example

Monadic second-order $\{E\}$-sentence:

$$\forall R: \ ((\exists z \in R) \Rightarrow (\exists x \in R \ \ \forall y \in R: \ \neg E(y,x)))$$

MSO-definable classes

Φ: MSO τ-sentence.

MSO-definable classes

Φ: MSO τ-sentence.

$[\![\Phi]\!]$: all finite τ-structures $\mathfrak{A}$ such that $\mathfrak{A} \models \Phi$.

MSO-definable classes

Φ: MSO τ-sentence.

$[\![\Phi]\!]$: all finite τ-structures $\mathfrak{A}$ such that $\mathfrak{A} \models \Phi$.

Example

Consider the MSO $\{E\}$-sentence Φ

$$\forall R : \;((\exists z \in R) \Rightarrow (\exists x \in R \;\; \forall y \in R : \; \neg E(y,x)))$$

MSO-definable classes

Φ: MSO τ-sentence.

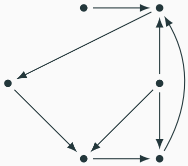
$[\![\Phi]\!]$: all finite τ-structures $\mathfrak{A}$ such that $\mathfrak{A} \models \Phi$.

Example

Consider the MSO $\{E\}$-sentence Φ

$$\forall R: \ ((\exists z \in R) \Rightarrow (\exists x \in R \ \ \forall y \in R: \ \neg E(y, x)))$$

A structure that satisfies $\neg\Phi$:



MSO-definable classes

Φ: MSO τ-sentence.

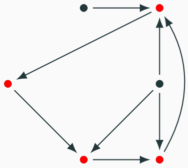
$[\![\Phi]\!]$: all finite τ-structures $\mathfrak{A}$ such that $\mathfrak{A} \models \Phi$.

Example

Consider the MSO $\{E\}$-sentence Φ

$$\forall R: \ ((\exists z \in R) \Rightarrow (\exists x \in R \ \ \forall y \in R: \ \neg E(y,x)))$$

A structure that satisfies $\neg\Phi$:



MSO-definable classes

Φ: MSO τ-sentence.

$[\![\Phi]\!]$: all finite τ-structures $\mathfrak{A}$ such that $\mathfrak{A} \models \Phi$.

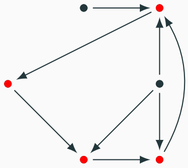
Example

Consider the MSO $\{E\}$-sentence Φ

$$\forall R: \ ((\exists z \in R) \Rightarrow (\exists x \in R \ \ \forall y \in R: \ \neg E(y,x)))$$

The class $[\![\Phi]\!]$ consists of all directed graphs that do not contain a cycle.

A structure that satisfies $\neg\Phi$:



Datalog

EDBs τ and IDBs ρ: disjoint sets of relation symbols, such that ρ contains a symbol **false** of arity 0.

Datalog

EDBs τ and IDBs ρ: disjoint sets of relation symbols, such that ρ contains a symbol **false** of arity 0.

Datalog rule: a term $\psi_0 :- \psi_1, \ldots, \psi_n$, where ψ_0 is an atomic ρ-formula and $\{\psi_1, \ldots, \psi_n\}$ are atomic $\tau \cup \rho$-formulas.

Datalog

EDBs τ and IDBs ρ: disjoint sets of relation symbols, such that ρ contains a symbol **false** of arity 0.

Datalog rule: a term $\psi_0 :- \psi_1, \ldots, \psi_n$, where ψ_0 is an atomic ρ-formula and $\{\psi_1, \ldots, \psi_n\}$ are atomic $\tau \cup \rho$-formulas.

Datalog program: set of Datalog rules.

Datalog

EDBs τ and IDBs ρ: disjoint sets of relation symbols, such that ρ contains a symbol **false** of arity 0.

Datalog rule: a term $\psi_0 :- \psi_1, \ldots, \psi_n$, where ψ_0 is an atomic ρ-formula and $\{\psi_1, \ldots, \psi_n\}$ are atomic $\tau \cup \rho$-formulas.

Datalog program: set of Datalog rules.

Datalog Semantics

A Datalog program Π rejects an instance, if the predicate **false** can be derived by *iterative rule application*. Otherwise Π accepts the instance.

Datalog

EDBs τ and IDBs ρ: disjoint sets of relation symbols, such that ρ contains a symbol **false** of arity 0.

Datalog rule: a term $\psi_0 : -\psi_1, \ldots, \psi_n$, where ψ_0 is an atomic ρ-formula and $\{\psi_1, \ldots, \psi_n\}$ are atomic $\tau \cup \rho$-formulas.

Datalog program: set of Datalog rules.

Datalog Semantics

A Datalog program Π rejects an instance, if the predicate **false** can be derived by *iterative rule application*. Otherwise Π accepts the instance.

We denote the class of accepted instances by $[\![\Pi]\!]$.

Datalog example

Datalog program Π^{succ} with EDB Succ and IDBs L, S, T and **false**:

$$
\begin{aligned}
S(x,y) &\ \text{:-}\ Succ(x,y) \\
L(x,z) &\ \text{:-}\ L(x,y),\ L(y,z) \\
S(y,x') &\ \text{:-}\ L(x,y),\ S(x,x') \\
L(x',y') &\ \text{:-}\ S(x,x'),\ S(x,y')
\end{aligned}
$$

$$
\begin{aligned}
T(x,y) &\ \text{:-}\ S(x,y) \\
T(x,z) &\ \text{:-}\ T(x,y),\ T(y,z) \\
\textbf{false} &\ \text{:-}\ T(x,x)
\end{aligned}
$$

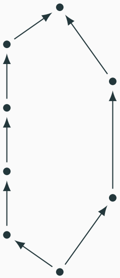
Datalog example

Datalog program Π^{succ} with EDB Succ and IDBs L, S, T and **false**:

$$
\begin{aligned}
S(x,y) &\text{ :- } \mathrm{Succ}(x,y) \\
L(x,z) &\text{ :- } L(x,y),\ L(y,z) \\
S(y,x') &\text{ :- } L(x,y),\ S(x,x') \\
L(x',y') &\text{ :- } S(x,x'),\ S(x,y')
\end{aligned}
$$

$$
\begin{aligned}
T(x,y) &\text{ :- } S(x,y) \\
T(x,z) &\text{ :- } T(x,y),\ T(y,z) \\
\textbf{false} &\text{ :- } T(x,x)
\end{aligned}
$$

Instance $\mathfrak{I}$:

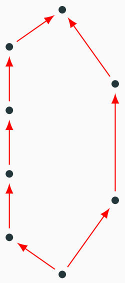


Datalog example

Datalog program Π^{succ} with EDB Succ and IDBs L, S, T and **false**:

$$
\begin{aligned}
S(x,y) &\text{ :- } Succ(x,y) \\
L(x,z) &\text{ :- } L(x,y),\ L(y,z) \\
S(y,x') &\text{ :- } L(x,y),\ S(x,x') \\
L(x',y') &\text{ :- } S(x,x'),\ S(x,y')
\end{aligned}
\qquad
\begin{aligned}
T(x,y) &\text{ :- } S(x,y) \\
T(x,z) &\text{ :- } T(x,y),\ T(y,z) \\
\textbf{false} &\text{ :- } T(x,x)
\end{aligned}
$$

Instance $\mathfrak{I}$:



Datalog example

Datalog program Π^{succ} with EDB Succ and IDBs L, S, T and **false**:

S(x,y) :-Succ(x,y)

L(x,z) :- L(x,y), L(y,z)

S(y,x') :- L(x,y), S(x,x')

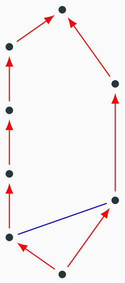
L(x',y') :- S(x,x'), S(x,y')

T(x,y) :- S(x,y)

T(x,z) :- T(x,y), T(y,z)

false :- T(x,x)

Instance $\mathfrak{I}$:

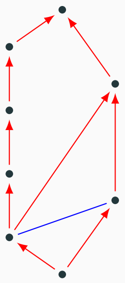


Datalog example

Datalog program Π^{succ} with EDB Succ and IDBs L, S, T and **false**:

S(x,y) :-Succ(x,y)		T(x,y) :- S(x,y)	
L(x,z) :- L(x,y), L(y,z)		T(x,z) :- T(x,y), T(y,z)	
S(y,x') :- L(x,y), S(x,x')		**false** :- T(x,x)	
L(x',y') :- S(x,x'), S(x,y')			

Instance $\mathfrak{I}$:



Datalog example

Datalog program Π^{succ} with EDB Succ and IDBs L, S, T and **false**:

S(x,y) :-Succ(x,y)

L(x,z) :- L(x,y), L(y,z)

S(y,x') :- L(x,y), S(x,x')

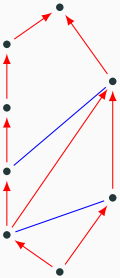
L(x',y') :- S(x,x'), S(x,y')

T(x,y) :- S(x,y)

T(x,z) :- T(x,y), T(y,z)

false :- T(x,x)

Instance $\mathfrak{I}$:



Datalog example

Datalog program Π^{succ} with EDB Succ and IDBs L, S, T and **false**:

S(x,y) :-Succ(x,y)

L(x,z) :- L(x,y), L(y,z)

S(y,x') :- L(x,y), S(x,x')

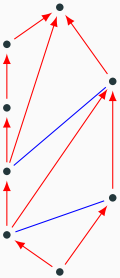
L(x',y') :- S(x,x'), S(x,y')

T(x,y) :- S(x,y)

T(x,z) :- T(x,y), T(y,z)

false :- T(x,x)

Instance $\mathfrak{I}$:



Datalog example

Datalog program Π^{succ} with EDB Succ and IDBs L, S, T and **false**:

$$S(x,y) \text{ :- } Succ(x,y)$$

$$L(x,z) \text{ :- } L(x,y),\ L(y,z)$$

$$S(y,x') \text{ :- } L(x,y),\ S(x,x')$$

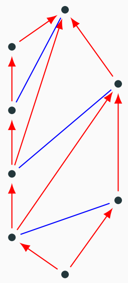
$$L(x',y') \text{ :- } S(x,x'),\ S(x,y')$$

$$T(x,y) \text{ :- } S(x,y)$$

$$T(x,z) \text{ :- } T(x,y),\ T(y,z)$$

$$\textbf{false} \text{ :- } T(x,x)$$

Instance $\mathfrak{I}$:



Datalog example

Datalog program Π^{succ} with EDB Succ and IDBs L, S, T and **false**:

$S(x,y)$:- $Succ(x,y)$

$L(x,z)$:- $L(x,y)$, $L(y,z)$

$S(y,x')$:- $L(x,y)$, $S(x,x')$

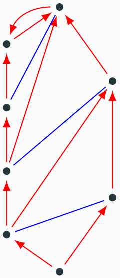
$L(x',y')$:- $S(x,x')$, $S(x,y')$

$T(x,y)$:- $S(x,y)$

$T(x,z)$:- $T(x,y)$, $T(y,z)$

false :- $T(x,x)$

Instance $\mathfrak{I}$:



Datalog example

Datalog program Π^{succ} with EDB Succ and IDBs L, S, T and **false**:

S(x,y) :-Succ(x,y)

L(x,z) :- L(x,y), L(y,z)

S(y,x') :- L(x,y), S(x,x')

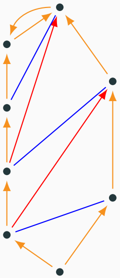
L(x',y') :- S(x,x'), S(x,y')

T(x,y) :- S(x,y)

T(x,z) :- T(x,y), T(y,z)

false :- T(x,x)

Instance $\mathfrak{I}$:



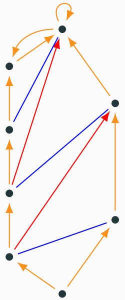
Datalog example

Datalog program Π^{succ} with EDB Succ and IDBs L, S, T and **false**:

$$
\begin{aligned}
S(x,y) &\text{ :- } Succ(x,y) \\
L(x,z) &\text{ :- } L(x,y),\ L(y,z) \\
S(y,x') &\text{ :- } L(x,y),\ S(x,x') \\
L(x',y') &\text{ :- } S(x,x'),\ S(x,y')
\end{aligned}
$$

$$
\begin{aligned}
T(x,y) &\text{ :- } S(x,y) \\
T(x,z) &\text{ :- } T(x,y),\ T(y,z) \\
\textbf{false} &\text{ :- } T(x,x)
\end{aligned}
$$

Instance $\mathfrak{I}$:



Datalog example

Datalog program Π^{succ} with EDB Succ and IDBs L, S, T and **false**:

S(x,y) :-Succ(x,y)

L(x,z) :- L(x,y), L(y,z)

S(y,x') :- L(x,y), S(x,x')

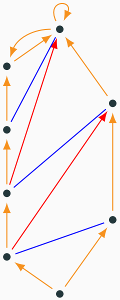
L(x',y') :- S(x,x'), S(x,y')

T(x,y) :- S(x,y)

T(x,z) :- T(x,y), T(y,z)

false :- T(x,x)

Instance $\mathfrak{I}$:



$\longrightarrow$ Π^{succ} derives **false** on $\mathfrak{I}$

Datalog example

Datalog program Π^{succ} with EDB Succ and IDBs L, S, T and **false**:

$$
\begin{aligned}
S(x,y) &:\!- \mathrm{Succ}(x,y) \\
L(x,z) &:\!- L(x,y),\ L(y,z) \\
S(y,x') &:\!- L(x,y),\ S(x,x') \\
L(x',y') &:\!- S(x,x'),\ S(x,y')
\end{aligned}
$$

$$
\begin{aligned}
T(x,y) &:\!- S(x,y) \\
T(x,z) &:\!- T(x,y),\ T(y,z) \\
\mathbf{false} &:\!- T(x,x)
\end{aligned}
$$

Π^{succ} accepts exactly the structures where for every cycle the number of traversed forward edges equals the number of traversed backward edges.

An observation about Datalog

Datalog program Π with EDB E and IDB R:

$$R(x,y) :- E(x,y)$$

$$R(x,y) :- R(x,z), R(z,y)$$

$$\textbf{false} :- R(x,x)$$

An observation about Datalog

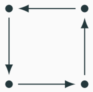
Datalog program Π with EDB E and IDB R:

$$R(x,y) :- E(x,y)$$

$$R(x,y) :- R(x,z), R(z,y)$$

$$\textbf{false} :- R(x,x)$$

Instance $\mathfrak{I}$:



An observation about Datalog

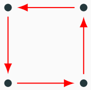
Datalog program Π with EDB E and IDB R:

$$R(x,y) :- E(x,y)$$

$$R(x,y) :- R(x,z), R(z,y)$$

$$\textbf{false} :- R(x,x)$$

Instance $\mathfrak{I}$:



An observation about Datalog

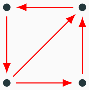
Datalog program Π with EDB E and IDB R:

$$R(x,y) :- E(x,y)$$

$$R(x,y) :- R(x,z), R(z,y)$$

$$\textbf{false} :- R(x,x)$$

Instance $\mathfrak{I}$:



An observation about Datalog

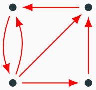
Datalog program Π with EDB E and IDB R:

$$R(x, y) :- E(x, y)$$

$$R(x, y) :- R(x, z), R(z, y)$$

$$\textbf{false} :- R(x, x)$$

Instance $\mathfrak{I}$:



An observation about Datalog

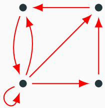
Datalog program Π with EDB E and IDB R:

$$R(x, y) :- E(x, y)$$

$$R(x, y) :- R(x, z), R(z, y)$$

$$\textbf{false} :- R(x, x)$$

Instance $\mathfrak{I}$:



Π derives **false** on $\mathfrak{I}$.

An observation about Datalog

Datalog program Π with EDB E and IDB R:

$$R(x, y) :- E(x, y)$$

$$R(x, y) :- R(x, z), R(z, y)$$

$$\textbf{false} :- R(x, x)$$

Instance $\mathfrak{I}$: Instance $\mathfrak{J}$:

Π derives **false** on $\mathfrak{I}$.

An observation about Datalog

Datalog program Π with EDB E and IDB R:

$$R(x, y) :- E(x, y)$$

$$R(x, y) :- R(x, z), R(z, y)$$

$$\textbf{false} :- R(x, x)$$

Instance $\mathfrak{I}$:

Instance $\mathfrak{J}$:

Π derives **false** on $\mathfrak{I}$.

An observation about Datalog

Datalog program Π with EDB E and IDB R:

$$R(x,y) :- E(x,y)$$

$$R(x,y) :- R(x,z), R(z,y)$$

$$\textbf{false} :- R(x,x)$$

Instance $\mathfrak{I}$: Instance $\mathfrak{J}$:

Π derives **false** on $\mathfrak{I}$.

An observation about Datalog

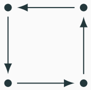
Datalog program Π with EDB E and IDB R:

$$R(x,y) :- E(x,y)$$

$$R(x,y) :- R(x,z), R(z,y)$$

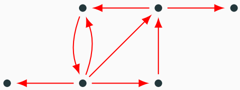
$$\textbf{false} :- R(x,x)$$

Instance $\mathfrak{I}$:



homomorphism

Instance $\mathfrak{J}$:



Π derives **false** on $\mathfrak{I}$.

An observation about Datalog

Datalog program Π with EDB E and IDB R:

$$R(x,y) :- E(x,y)$$

$$R(x,y) :- R(x,z), R(z,y)$$

$$\textbf{false} :- R(x,x)$$

Instance $\mathfrak{I}$: Instance $\mathfrak{J}$:

Π derives **false** on $\mathfrak{I}$. Π derives **false** on $\mathfrak{J}$.

An observation about Datalog

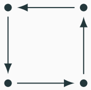
Datalog program Π with EDB E and IDB R:

$$R(x,y) :- E(x,y)$$

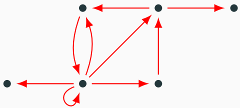
$$R(x,y) :- R(x,z), R(z,y)$$

$$\textbf{false} :- R(x,x)$$

Instance $\mathfrak{I}$:



Instance $\mathfrak{J}$:



Π derives **false** on $\mathfrak{I}$.

Π derives **false** on $\mathfrak{J}$.

Observation

Let $\mathcal{C}$ be defined by a Datalog program. Let $\mathfrak{A}$ and $\mathfrak{B}$ be structures where $\mathfrak{B}$ is in $\mathcal{C}$ and $\mathfrak{A}$ has a homomorphism to $\mathfrak{B}$. Then $\mathfrak{A}$ is in $\mathcal{C}$.

We say that $\mathcal{C}$ is closed under inverse homomorphisms.

Result 1

Result 1

Let Φ be an MSO sentence such that $[\![\Phi]\!]$ is closed under inverse homomorphisms.

Result 1

Result 1

Let Φ be an MSO sentence such that $[\![\Phi]\!]$ is closed under inverse homomorphisms. Then $[\![\Phi]\!]$ is a finite union of MSO-definable CSPs.

Result 1

Result 1

Let Φ be an MSO sentence such that $[\![\Phi]\!]$ is closed under inverse homomorphisms. Then $[\![\Phi]\!]$ is a finite union of MSO-definable CSPs.

Definition

Let $\mathfrak{D}$ be a τ-structure. The CSP of $\mathfrak{D}$ is the class of all finite τ-structures that have a homomorphism to $\mathfrak{D}$. We denote it by $\mathrm{CSP}(\mathfrak{D})$.

Result 1

Let Φ be an MSO sentence such that $[\![\Phi]\!]$ is closed under inverse homomorphisms. Then $[\![\Phi]\!]$ is a finite union of MSO-definable CSPs.

Definition

Let $\mathfrak{D}$ be a τ-structure. The CSP of $\mathfrak{D}$ is the class of all finite τ-structures that have a homomorphism to $\mathfrak{D}$. We denote it by $\mathrm{CSP}(\mathfrak{D})$.

Example:

- $\mathrm{CSP}(\mathbb{Q}; <)$ is the class of all digraphs without a cycle.

Result 1

Let Φ be an MSO sentence such that $[\![\Phi]\!]$ is closed under inverse homomorphisms. Then $[\![\Phi]\!]$ is a finite union of MSO-definable CSPs.

Definition

Let $\mathfrak{D}$ be a τ-structure. The CSP of $\mathfrak{D}$ is the class of all finite τ-structures that have a homomorphism to $\mathfrak{D}$. We denote it by $\mathrm{CSP}(\mathfrak{D})$.

Example:

- $\mathrm{CSP}(\mathbb{Q}; <)$ is the class of all digraphs without a cycle.
- $\mathrm{CSP}(\mathbb{Z}; \mathrm{Succ} = \{(x, y) \mid x + 1 = y\}) = [\![\Pi^{succ}]\!]$ is the class of all graphs where for each two nodes all connecting paths have the same length.

Result 1

Result 1

Let Φ be an MSO sentence such that $[\![\Phi]\!]$ is closed under inverse homomorphisms. Then $[\![\Phi]\!]$ is a finite union of MSO-definable CSPs.

Definition

Let $\mathfrak{D}$ be a τ-structure. The CSP of $\mathfrak{D}$ is the class of all finite τ-structures that have a homomorphism to $\mathfrak{D}$. We denote it by $\mathrm{CSP}(\mathfrak{D})$.

Proposition

A class of τ-structures $\mathcal{C}$ is a CSP for some structure iff it is closed under inverse homomorphisms and closed under disjoint unions,
i.e. if $\mathfrak{A}, \mathfrak{B} \in \mathcal{C}$, then $\mathfrak{A} \uplus \mathfrak{B} \in \mathcal{C}$.

Application of Result 1

Let B and R be unary and let E be a binary relation symbol.

Datalog program Π

$$L(x,y) :- B(x), B(y)$$
$$L(x,y) :- L(x',y'), E(x',x), E(y',y)$$
$$\mathbf{false} :- R(x), L(x,x'), E(x',y)$$

Application of Result 1

Let B and R be unary and let E be a binary relation symbol.

Datalog program Π

$$L(x,y) :- B(x), B(y)$$

$$L(x,y) :- L(x',y'), E(x',x), E(y',y)$$

$$\textbf{false} :- R(x), L(x,x'), E(x',y)$$

Claim

$[\![\Pi]\!]$ is not MSO-definable.

Application of Result 1

Let B and R be unary and let E be a binary relation symbol.

Datalog program Π

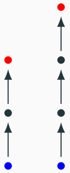
$$L(x,y) :- B(x), B(y)$$

$$L(x,y) :- L(x',y'), E(x',x), E(y',y)$$

$$\textbf{false} :- R(x), L(x,x'), E(x',y)$$

Claim

$[\![\Pi]\!]$ is not MSO-definable.



Application of Result 1

Let B and R be unary and let E be a binary relation symbol.

Datalog program Π

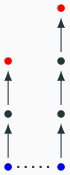
$$L(x, y) :- B(x), B(y)$$

$$L(x, y) :- L(x', y'), E(x', x), E(y', y)$$

$$\textbf{false} :- R(x), L(x, x'), E(x', y)$$

Claim

$[\![\Pi]\!]$ is not MSO-definable.



Application of Result 1

Let B and R be unary and let E be a binary relation symbol.

Datalog program Π

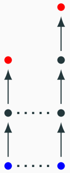
$$L(x,y) :- B(x), B(y)$$

$$L(x,y) :- L(x',y'), E(x',x), E(y',y)$$

$$\textbf{false} :- R(x), L(x,x'), E(x',y)$$

Claim

$[\![\Pi]\!]$ is not MSO-definable.



Application of Result 1

Let B and R be unary and let E be a binary relation symbol.

Datalog program Π

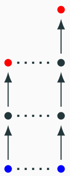
$$L(x,y) :- B(x), B(y)$$

$$L(x,y) :- L(x',y'), E(x',x), E(y',y)$$

$$\mathbf{false} :- R(x), L(x,x'), E(x',y)$$

Claim

$[\![\Pi]\!]$ is not MSO-definable.



Application of Result 1

Let B and R be unary and let E be a binary relation symbol.

Datalog program Π

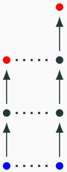
$$L(x,y) :- B(x), B(y)$$

$$L(x,y) :- L(x',y'), E(x',x), E(y',y)$$

$$\textbf{false} :- R(x), L(x,x'), E(x',y)$$

Claim

$[\![\Pi]\!]$ is not MSO-definable.



$\notin [\![\Pi]\!]$

Application of Result 1

Let B and R be unary and let E be a binary relation symbol.

Datalog program Π

$$L(x,y) :- B(x), B(y)$$

$$L(x,y) :- L(x',y'), E(x',x), E(y',y)$$

$$\textbf{false} :- R(x), L(x,x'), E(x',y)$$

Claim

$[\![\Pi]\!]$ is not MSO-definable.

 $\in [\![\Pi]\!]$

Application of Result 1

Let B and R be unary and let E be a binary relation symbol.

Datalog program Π

$$L(x,y) :- B(x), B(y)$$

$$L(x,y) :- L(x',y'), E(x',x), E(y',y)$$

$$\textbf{false} :- R(x), L(x,x'), E(x',y)$$

Claim

$[\![\Pi]\!]$ is not MSO-definable.

 $\in [\![\Pi]\!]$

Application of Result 1

Let B and R be unary and let E be a binary relation symbol.

Datalog program Π

$$L(x, y) :- B(x), B(y)$$

$$L(x, y) :- L(x', y'), E(x', x), E(y', y)$$

$$\textbf{false} :- R(x), L(x, x'), E(x', y)$$

Claim

$[\![\Pi]\!]$ is not MSO-definable.

$\in [\![\Pi]\!]$

Application of Result 1

Let B and R be unary and let E be a binary relation symbol.

Datalog program Π

$$L(x, y) :- B(x), B(y)$$

$$L(x, y) :- L(x', y'), E(x', x), E(y', y)$$

$$\textbf{false} :- R(x), L(x, x'), E(x', y)$$

Claim

$[\![\Pi]\!]$ is not MSO-definable.

$\in [\![\Pi]\!]$

Application of Result 1

Let B and R be unary and let E be a binary relation symbol.

Datalog program Π

$$L(x,y) :- B(x), B(y)$$

$$L(x,y) :- L(x',y'), E(x',x), E(y',y)$$

$$\textbf{false} :- R(x), L(x,x'), E(x',y)$$

Claim

$[\![\Pi]\!]$ is not MSO-definable.

Observation: There exists an infinite set $X \subset [\![\Pi]\!]$ such that for all distinct elements $\mathfrak{A}, \mathfrak{B} \in X$ the disjoint union $\mathfrak{A} \uplus \mathfrak{B}$ is not in $[\![\Pi]\!]$.

Application of Result 1

Let B and R be unary and let E be a binary relation symbol.

Datalog program Π

$$L(x,y) :- B(x), B(y)$$
$$L(x,y) :- L(x',y'), E(x',x), E(y',y)$$
$$\textbf{false} :- R(x), L(x,x'), E(x',y)$$

Claim

$[\![\Pi]\!]$ is not MSO-definable.

Observation: There exists an infinite set $X \subset [\![\Pi]\!]$ such that for all distinct elements $\mathfrak{A}, \mathfrak{B} \in X$ the disjoint union $\mathfrak{A} \uplus \mathfrak{B}$ is not in $[\![\Pi]\!]$.

$\longrightarrow$ $[\![\Pi]\!]$ is not a finite union of CSPs.

Application of Result 1

Let B and R be unary and let E be a binary relation symbol.

Datalog program Π

$$L(x,y) :- B(x), B(y)$$

$$L(x,y) :- L(x',y'), E(x',x), E(y',y)$$

$$\textbf{false} :- R(x), L(x,x'), E(x',y)$$

Claim

$[\![\Pi]\!]$ is not MSO-definable.

Observation: There exists an infinite set $X \subset [\![\Pi]\!]$ such that for all distinct elements $\mathfrak{A}, \mathfrak{B} \in X$ the disjoint union $\mathfrak{A} \uplus \mathfrak{B}$ is not in $[\![\Pi]\!]$.

$\longrightarrow$ $[\![\Pi]\!]$ is not a finite union of CSPs.

$\longrightarrow$ By Result 1 and the observation about Datalog, the class $[\![\Pi]\!]$ is not MSO-definable.

Result 2

Result 2

Let Φ be an MSO sentence such that $[\![\Phi]\!]$ is closed under disjoint unions and inverse homomorphisms.

Result 2

Result 2

Let Φ be an MSO sentence such that $[\![\Phi]\!]$ is closed under disjoint unions and inverse homomorphisms. Then $[\![\Phi]\!]$ is the CSP of an ω-categorical structure.

Result 2

Let Φ be an MSO sentence such that $[\![\Phi]\!]$ is closed under disjoint unions and inverse homomorphisms. Then $[\![\Phi]\!]$ is the CSP of an ω-categorical structure.

Definition

A countable structure $\mathfrak{D}$ is called ω-categorical if all countable models of the first-order theory of $\mathfrak{D}$ are isomorphic to $\mathfrak{D}$.

Result 2

Result 2

Let Φ be an MSO sentence such that $[\![\Phi]\!]$ is closed under disjoint unions and inverse homomorphisms. Then $[\![\Phi]\!]$ is the CSP of an ω-categorical structure.

Definition

A countable structure $\mathfrak{D}$ is called ω-categorical if all countable models of the first-order theory of $\mathfrak{D}$ are isomorphic to $\mathfrak{D}$.

Example: $(\mathbb{Q}; <)$ is ω-categorical.

- There exists up to isomorphism only one countable dense linear order without endpoints.

Result 2

Result 2

Let Φ be an MSO sentence such that $[\![\Phi]\!]$ is closed under disjoint unions and inverse homomorphisms. Then $[\![\Phi]\!]$ is the CSP of an ω-categorical structure.

Definition

A countable structure $\mathfrak{D}$ is called ω-categorical if all countable models of the first-order theory of $\mathfrak{D}$ are isomorphic to $\mathfrak{D}$.

Example: $(\mathbb{Q}; <)$ is ω-categorical.

- There exists up to isomorphism only one countable dense linear order without endpoints.
- "Dense linear order without endpoints" can be expressed in first-order logic.

Result 2

Result 2

Let Φ be an MSO sentence such that $[\![\Phi]\!]$ is closed under disjoint unions and inverse homomorphisms. Then $[\![\Phi]\!]$ is the CSP of an ω-categorical structure.

Definition

A countable structure $\mathfrak{D}$ is called ω-categorical if all countable models of the first-order theory of $\mathfrak{D}$ are isomorphic to $\mathfrak{D}$.

Remark

Result 2 can be used to achieve a characterization of MSO $\cap$ Datalog in terms of *existential pebble games*.

Application of Result 2

Claim

For the Datalog program Π^{succ} the class $[\![\Pi^{succ}]\!]$ is not definable in MSO.

Application of Result 2

Claim

For the Datalog program Π^{succ} the class $[\![\Pi^{succ}]\!]$ is not definable in MSO.

- Consider $\mathfrak{A} = (\mathbb{Z}; \mathrm{Succ} = \{(x, y) \mid x + 1 = y\})$.

Application of Result 2

Claim

For the Datalog program Π^{succ} the class $[\![\Pi^{succ}]\!]$ is not definable in MSO.

- Consider $\mathfrak{A} = (\mathbb{Z}; \mathrm{Succ} = \{(x, y) \mid x + 1 = y\})$.
- $\mathrm{CSP}(\mathfrak{A}) = [\![\Pi^{succ}]\!]$.

Application of Result 2

Claim

For the Datalog program Π^{succ} the class $[\![\Pi^{succ}]\!]$ is not definable in MSO.

- Consider $\mathfrak{A} = (\mathbb{Z}; \mathrm{Succ} = \{(x, y) \mid x + 1 = y\})$.
- $\mathrm{CSP}(\mathfrak{A}) = [\![\Pi^{succ}]\!]$.
- $\mathfrak{A}$ and $\mathfrak{A} \uplus \mathfrak{A}$ satisfy exactly the same first-order sentences, e.g.,

$$\forall x \exists y, z.\ \mathrm{Succ}(y, x) \wedge \mathrm{Succ}(x, y).$$

Application of Result 2

Claim

For the Datalog program Π^{succ} the class $[\![\Pi^{succ}]\!]$ is not definable in MSO.

- Consider $\mathfrak{A} = (\mathbb{Z}; \mathrm{Succ} = \{(x, y) \mid x + 1 = y\})$.
- $\mathrm{CSP}(\mathfrak{A}) = [\![\Pi^{succ}]\!]$.
- $\mathfrak{A}$ and $\mathfrak{A} \uplus \mathfrak{A}$ satisfy exactly the same first-order sentences, e.g.,

$$\forall x \exists y, z.\ \mathrm{Succ}(y, x) \wedge \mathrm{Succ}(x, y).$$

- $\mathfrak{A}$ and $\mathfrak{A} \uplus \mathfrak{A}$ are not isomorphic.

Application of Result 2

Claim

For the Datalog program Π^{succ} the class $[\![\Pi^{succ}]\!]$ is not definable in MSO.

- Consider $\mathfrak{A} = (\mathbb{Z}; \mathrm{Succ} = \{(x,y) \mid x+1=y\})$.
- $\mathrm{CSP}(\mathfrak{A}) = [\![\Pi^{succ}]\!]$.
- $\mathfrak{A}$ and $\mathfrak{A} \uplus \mathfrak{A}$ satisfy exactly the same first-order sentences, e.g.,

$$\forall x \exists y, z.\ \mathrm{Succ}(y,x) \land \mathrm{Succ}(x,y).$$

- $\mathfrak{A}$ and $\mathfrak{A} \uplus \mathfrak{A}$ are not isomorphic.
- Therefore $\mathfrak{A}$ is not ω-categorical.

Application of Result 2

Claim

For the Datalog program Π^{succ} the class $[\![\Pi^{succ}]\!]$ is not definable in MSO.

- Consider $\mathfrak{A} = (\mathbb{Z}; \mathrm{Succ} = \{(x, y) \mid x + 1 = y\})$.
- $\mathrm{CSP}(\mathfrak{A}) = [\![\Pi^{succ}]\!]$.
- $\mathfrak{A}$ and $\mathfrak{A} \uplus \mathfrak{A}$ satisfy exactly the same first-order sentences, e.g.,

$$\forall x \exists y, z.\ \mathrm{Succ}(y, x) \wedge \mathrm{Succ}(x, y).$$

- $\mathfrak{A}$ and $\mathfrak{A} \uplus \mathfrak{A}$ are not isomorphic.
- Therefore $\mathfrak{A}$ is not ω-categorical.
- Even more: there exists no ω-categorical structure $\mathfrak{B}$ with $\mathrm{CSP}(\mathfrak{B}) = \mathrm{CSP}(\mathfrak{A})$ (needs short proof).

Application of Result 2

Claim

For the Datalog program Π^{succ} the class $[\![\Pi^{succ}]\!]$ is not definable in MSO.

- Consider $\mathfrak{A} = (\mathbb{Z}; \mathrm{Succ} = \{(x, y) \mid x + 1 = y\})$.
- $\mathrm{CSP}(\mathfrak{A}) = [\![\Pi^{succ}]\!]$.
- $\mathfrak{A}$ and $\mathfrak{A} \uplus \mathfrak{A}$ satisfy exactly the same first-order sentences, e.g.,

$$\forall x \exists y, z.\ \mathrm{Succ}(y, x) \wedge \mathrm{Succ}(x, y).$$

- $\mathfrak{A}$ and $\mathfrak{A} \uplus \mathfrak{A}$ are not isomorphic.
- Therefore $\mathfrak{A}$ is not ω-categorical.
- Even more: there exists no ω-categorical structure $\mathfrak{B}$ with $\mathrm{CSP}(\mathfrak{B}) = \mathrm{CSP}(\mathfrak{A})$ (needs short proof).
- By Result 2, $\mathrm{CSP}(\mathfrak{A}) = [\![\Pi^{succ}]\!]$ is not definable in MSO.

Description of MSO $\cap$ Datalog

Result 1

Let Φ be an MSO sentence such that $[\![\Phi]\!]$ is closed under inverse homomorphisms. Then $[\![\Phi]\!]$ is a finite union of MSO-definable CSPs.

Description of MSO $\cap$ Datalog

Result 1

Let Φ be an MSO sentence such that $[\![\Phi]\!]$ is closed under inverse homomorphisms. Then $[\![\Phi]\!]$ is a finite union of MSO-definable CSPs.

Result 2

Let Φ be an MSO sentence such that $[\![\Phi]\!]$ is closed under disjoint unions and inverse homomorphisms. Then $[\![\Phi]\!]$ is the CSP of an ω-categorical structure.

Description of MSO $\cap$ Datalog

Result 1

Let Φ be an MSO sentence such that $[\![\Phi]\!]$ is closed under inverse homomorphisms. Then $[\![\Phi]\!]$ is a finite union of MSO-definable CSPs.

Result 2

Let Φ be an MSO sentence such that $[\![\Phi]\!]$ is closed under disjoint unions and inverse homomorphisms. Then $[\![\Phi]\!]$ is the CSP of an ω-categorical structure.

Result 3

Every problem in MSO $\cap$ Datalog is the finite union of ω-categorical CSPs.

A generalization of the results

- Guarded second-order logic (GSO) is a generalization of MSO.

A generalization of the results

- Guarded second-order logic (GSO) is a generalization of MSO.
- GSO coincides with Courselle's MSO_2.

A generalization of the results

- Guarded second-order logic (GSO) is a generalization of MSO.
- GSO coincides with Courselle's MSO_2.
- Result 1, Result 2 and Result 3 hold also for GSO.

Open problems

- The logic Nemodeq introduced by Rudolph and Krötzsch is contained in MSO $\cap$ Datalog. Does the converse also hold?

Open problems

- The logic Nemodeq introduced by Rudolph and Krötzsch is contained in MSO $\cap$ Datalog. Does the converse also hold?
- Can the intersection of GSO and Datalog be described by some logic?

Open problems

- The logic Nemodeq introduced by Rudolph and Krötzsch is contained in MSO $\cap$ Datalog. Does the converse also hold?
- Can the intersection of GSO and Datalog be described by some logic?
- Is there an example of a CSP for a reduct of a finitely bounded homogeneous structure that is not in GSO?

Thank you for your attention!

Result 1 (proof idea)

Result 1
Let Φ be an MSO sentence such that $[\![\Phi]\!]$ is closed under inverse homomorphisms. Then $[\![\Phi]\!]$ is a finite union of MSO definable CSPs.

Result 1 (proof idea)

Result 1
Let Φ be an MSO sentence such that $[\![\Phi]\!]$ is closed under inverse homomorphisms. Then $[\![\Phi]\!]$ is a finite union of MSO definable CSPs.

- Equivalence relation $\sim$ on $\mathcal{C} = [\![\Phi]\!]$ with $A \sim B$ if and only if

$$\forall D \in \mathcal{C} : A \uplus D \in \mathcal{C} \Leftrightarrow B \uplus D \in \mathcal{C}.$$

Result 1 (proof idea)

Result 1
Let Φ be an MSO sentence such that $[\![\Phi]\!]$ is closed under inverse homomorphisms. Then $[\![\Phi]\!]$ is a finite union of MSO definable CSPs.

- Equivalence relation $\sim$ on $\mathcal{C} = [\![\Phi]\!]$ with $A \sim B$ if and only if

$$\forall D \in \mathcal{C} : A \uplus D \in \mathcal{C} \Leftrightarrow B \uplus D \in \mathcal{C}.$$

- $\sim$ has finitely many classes.

Result 1 (proof idea)

Result 1
Let Φ be an MSO sentence such that $[\![\Phi]\!]$ is closed under inverse homomorphisms. Then $[\![\Phi]\!]$ is a finite union of MSO definable CSPs.

- Equivalence relation $\sim$ on $\mathcal{C} = [\![\Phi]\!]$ with $A \sim B$ if and only if

$$\forall D \in \mathcal{C} : A \uplus D \in \mathcal{C} \Leftrightarrow B \uplus D \in \mathcal{C}.$$

- $\sim$ has finitely many classes.
  - Let q be the quantifier rank of Φ.

Result 1 (proof idea)

Result 1
Let Φ be an MSO sentence such that $[\![\Phi]\!]$ is closed under inverse homomorphisms. Then $[\![\Phi]\!]$ is a finite union of MSO definable CSPs.

- Equivalence relation $\sim$ on $\mathcal{C} = [\![\Phi]\!]$ with $\mathrm{A} \sim \mathrm{B}$ if and only if

$$\forall \mathrm{D} \in \mathcal{C} : \mathrm{A} \uplus \mathrm{D} \in \mathcal{C} \Leftrightarrow \mathrm{B} \uplus \mathrm{D} \in \mathcal{C}.$$

- $\sim$ has finitely many classes.
  - Let q be the quantifier rank of Φ.
  - $\mathrm{A} \equiv_q \mathrm{B}$ if A and B satisfy the same MSO sentences of quantifier rank q.

Result 1 (proof idea)

Result 1
Let Φ be an MSO sentence such that $[\![\Phi]\!]$ is closed under inverse homomorphisms. Then $[\![\Phi]\!]$ is a finite union of MSO definable CSPs.

- Equivalence relation $\sim$ on $\mathcal{C} = [\![\Phi]\!]$ with A $\sim$ B if and only if

$$\forall \mathrm{D} \in \mathcal{C} : \mathrm{A} \uplus \mathrm{D} \in \mathcal{C} \Leftrightarrow \mathrm{B} \uplus \mathrm{D} \in \mathcal{C}.$$

- $\sim$ has finitely many classes.
    - Let q be the quantifier rank of Φ.
    - A $\equiv_q$ B if A and B satisfy the same MSO sentences of quantifier rank q.
    - $\equiv_q$ has finitely many classes.

Result 1 (proof idea)

Result 1
Let Φ be an MSO sentence such that $[\![\Phi]\!]$ is closed under inverse homomorphisms. Then $[\![\Phi]\!]$ is a finite union of MSO definable CSPs.

- Equivalence relation $\sim$ on $\mathcal{C} = [\![\Phi]\!]$ with $A \sim B$ if and only if

$$\forall D \in \mathcal{C} : A \uplus D \in \mathcal{C} \Leftrightarrow B \uplus D \in \mathcal{C}.$$

- $\sim$ has finitely many classes.
    - Let q be the quantifier rank of Φ.
    - $A \equiv_q B$ if A and B satisfy the same MSO sentences of quantifier rank q.
    - $\equiv_q$ has finitely many classes.
    - If $A \equiv_q B$, then $A \sim B$.

Result 1 (proof idea)

Result 1
Let Φ be an MSO sentence such that $[\![\Phi]\!]$ is closed under inverse homomorphisms. Then $[\![\Phi]\!]$ is a finite union of MSO definable CSPs.

- Equivalence relation $\sim$ on $\mathcal{C} = [\![\Phi]\!]$ with $A \sim B$ if and only if

$$\forall D \in \mathcal{C} : A \uplus D \in \mathcal{C} \Leftrightarrow B \uplus D \in \mathcal{C}.$$

- $\sim$ has finitely many classes.
  - Let q be the quantifier rank of Φ.
  - $A \equiv_q B$ if A and B satisfy the same MSO sentences of quantifier rank q.
  - $\equiv_q$ has finitely many classes.
  - If $A \equiv_q B$, then $A \sim B$.
  - Therefore $\sim$ has finitely many classes.

Result 1 (proof idea)

Result 1
Let Φ be an MSO sentence such that $[\![\Phi]\!]$ is closed under inverse homomorphisms. Then $[\![\Phi]\!]$ is a finite union of MSO definable CSPs.

- Equivalence relation $\sim$ on $\mathcal{C} = [\![\Phi]\!]$ with $A \sim B$ if and only if

$$\forall D \in \mathcal{C} : A \uplus D \in \mathcal{C} \Leftrightarrow B \uplus D \in \mathcal{C}.$$

- $\sim$ has finitely many classes.
- $\mathcal{C}$ is a CSP iff $\sim$ has only one class.

Result 1 (proof idea)

Result 1
Let Φ be an MSO sentence such that $[\![\Phi]\!]$ is closed under inverse homomorphisms. Then $[\![\Phi]\!]$ is a finite union of MSO definable CSPs.

- Equivalence relation $\sim$ on $\mathcal{C} = [\![\Phi]\!]$ with $A \sim B$ if and only if

$$\forall D \in \mathcal{C} : A \uplus D \in \mathcal{C} \Leftrightarrow B \uplus D \in \mathcal{C}.$$

- $\sim$ has finitely many classes.
- $\mathcal{C}$ is a CSP iff $\sim$ has only one class.
- Induction argument over the number of classes of $\sim$.

Result 2 (proof idea)

Generalize the equivalence relation $\sim$:

Result 2 (proof idea)

Generalize the equivalence relation $\sim$:

- Let $\mathcal{C} = [\![\Phi]\!]$ and let X be a set of new constant symbols, $|X| = n$.

Result 2 (proof idea)

Generalize the equivalence relation $\sim$:

- Let $\mathcal{C} = [\![\Phi]\!]$ and let X be a set of new constant symbols, $|X| = n$.
- Let $\mathcal{C}^X$ be the class of $\tau \cup X$-structures whose τ-reducts are from $\mathcal{C}$.

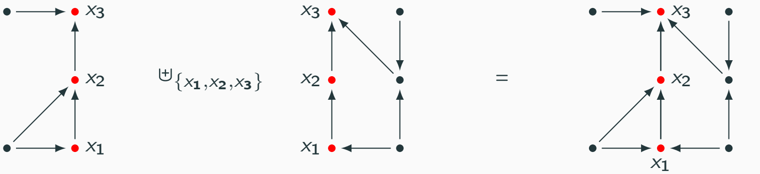
Result 2 (proof idea)

Generalize the equivalence relation $\sim$:

- Let $\mathcal{C} = [\![\Phi]\!]$ and let X be a set of new constant symbols, $|X| = n$.
- Let $\mathcal{C}^X$ be the class of $\tau \cup X$-structures whose τ-reducts are from $\mathcal{C}$.
- For two structures $\mathrm{A}, \mathrm{B} \in \mathcal{C}^X$ the structure $\mathrm{A} \uplus_X \mathrm{B}$ is defined as the pairwise identification of the constants X in $\mathrm{A} \uplus \mathrm{B}$.

Generalize the equivalence relation $\sim$:

- Let $\mathcal{C} = [\![\Phi]\!]$ and let X be a set of new constant symbols, $|X| = n$.
- Let $\mathcal{C}^X$ be the class of $\tau \cup X$-structures whose τ-reducts are from $\mathcal{C}$.
- For two structures $\mathrm{A}, \mathrm{B} \in \mathcal{C}^X$ the structure $\mathrm{A} \uplus_X \mathrm{B}$ is defined as the pairwise identification of the constants X in $\mathrm{A} \uplus \mathrm{B}$.



Result 2 (proof idea)

Generalize the equivalence relation $\sim$:

- Let $\mathcal{C} = [\![\Phi]\!]$ and let X be a set of new constant symbols, $|X| = n$.
- Let $\mathcal{C}^X$ be the class of $\tau \cup X$-structures whose τ-reducts are from $\mathcal{C}$.
- For two structures $\mathrm{A}, \mathrm{B} \in \mathcal{C}^X$ the structure $\mathrm{A} \uplus_X \mathrm{B}$ is defined as the pairwise identification of the constants X in $\mathrm{A} \uplus \mathrm{B}$.
- Consider equivalence relation $\sim_n^{\mathcal{C}}$ on $\mathcal{C}^X$ with $\mathrm{A} \sim_n^{\mathcal{C}} \mathrm{B}$ if and only if

$$\forall \mathrm{D} \in \mathcal{C}^X : \mathrm{A} \uplus_X \mathrm{D} \in \mathcal{C}^X \Leftrightarrow \mathrm{B} \uplus_X \mathrm{D} \in \mathcal{C}^X.$$

Result 2 (proof idea)

Generalize the equivalence relation $\sim$:

- Let $\mathcal{C} = [\![\Phi]\!]$ and let X be a set of new constant symbols, $|X| = n$.
- Let $\mathcal{C}^X$ be the class of $\tau \cup X$-structures whose τ-reducts are from $\mathcal{C}$.
- For two structures $\mathrm{A}, \mathrm{B} \in \mathcal{C}^X$ the structure $\mathrm{A} \uplus_X \mathrm{B}$ is defined as the pairwise identification of the constants X in $\mathrm{A} \uplus \mathrm{B}$.
- Consider equivalence relation $\sim_n^{\mathcal{C}}$ on $\mathcal{C}^X$ with $\mathrm{A} \sim_n^{\mathcal{C}} \mathrm{B}$ if and only if

$$\forall \mathrm{D} \in \mathcal{C}^X : \mathrm{A} \uplus_X \mathrm{D} \in \mathcal{C}^X \Leftrightarrow \mathrm{B} \uplus_X \mathrm{D} \in \mathcal{C}^X.$$

Use of the following theorem:

Theorem (Bodirsky, Hils, Martin)
Let $\mathcal{C}$ be closed under inverse homomorphisms and disjoint unions. Then there exists an ω-categorical structure B such that $\mathrm{CSP}(\mathrm{B}) = \mathcal{C}$ if and only if $\sim_n^{\mathcal{C}}$ has finitely many equivalence classes for each $n \in \mathbb{N}$.