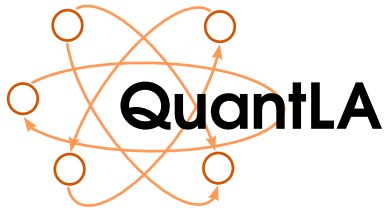
MSO ∩ Datalog

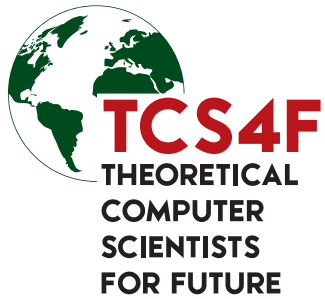
Simon Knäuer (TU Dresden)

Joint work with Manuel Bodirsky and Sebastian Rudolph

QuantLA Workshop 2020



DFG Research Training Group 1763



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.

Which computational problems are expressible in Monadic Second-Order Logic **AND** can be solved by a Datalog program?

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.

Which computational problems are expressible in Monadic Second-Order Logic **AND** can be solved by a Datalog program?

Can we describe such problems in a 'nice' way?

Introduction 2

Contributions:

Introduction 2

Contributions:

- Description of MSO $\cap$ Datalog in terms of CSPs.

Introduction 2

Contributions:

- Description of MSO $\cap$ Datalog in terms of CSPs.
- Necessary conditions for MSO definability in Datalog .

Introduction 2

Contributions:

- Description of MSO ∩ Datalog in terms of CSPs.
- Necessary conditions for MSO definability in Datalog .
- Pebble game characterization of MSO ∩ Datalog. 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:

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

MSO definable classes

Φ: MSO τ-sentence.

MSO definable classes

Φ: MSO τ-sentence.

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

MSO definable classes

Φ: MSO τ-sentence.

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

Example

Consider the MSO $\{E\}$-sentence Φ

$$\forall X \neq \varnothing \; \exists x \in X \; \forall y \in X : \; \neg E(x, y).$$

MSO definable classes

Φ: MSO τ-sentence.

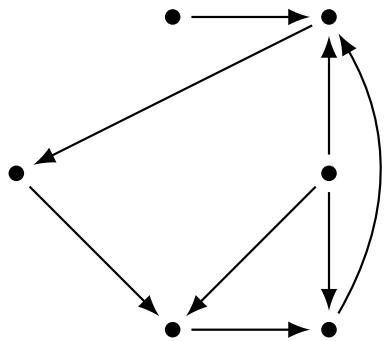
$[\![\Phi]\!]$: all finite τ-structures $\mathbf{A}$ such that $\mathbf{A} \models \Phi$.

Example

Consider the MSO $\{E\}$-sentence Φ

$$\forall X \neq \varnothing \; \exists x \in X \; \forall y \in X : \; \neg E(x, y).$$

A structure that satisfies $\neg\Phi$:



MSO definable classes

Φ: MSO τ-sentence.

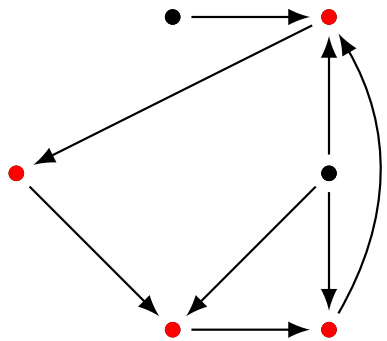
$[\![\Phi]\!]$: all finite τ-structures $\mathbf{A}$ such that $\mathbf{A} \models \Phi$.

Example

Consider the MSO $\{E\}$-sentence Φ

$$\forall X \neq \varnothing \; \exists x \in X \; \forall y \in X : \; \neg E(x, y).$$

A structure that satisfies $\neg\Phi$:



MSO definable classes

Φ: MSO τ-sentence.

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

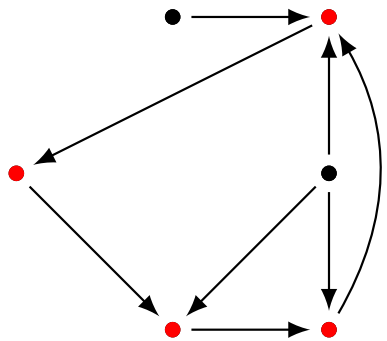
Example

Consider the MSO $\{E\}$-sentence Φ

$$\forall X \neq \varnothing \; \exists x \in X \; \forall y \in X : \; \neg E(x, y).$$

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 *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 *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**:

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

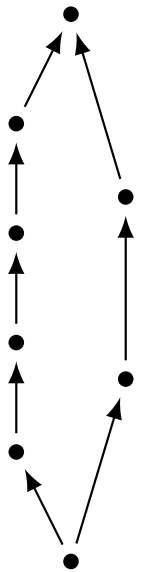
$$T(x,y) :- S(x,y)$$
$$T(x,z) :- T(x,y), T(y,z)$$
$$\textbf{false} :- T(x,x)$$

Datalog example

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

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

Instance I:



Datalog example

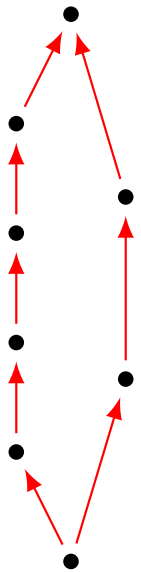
Datalog program Π^{succ} with EDB Succ and IDBs L, S, T and **false**:

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

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

$$T(x,y) :- S(x,y)$$
$$T(x,z) :- T(x,y), T(y,z)$$
$$\textbf{false} :- T(x,x)$$

Instance I:



Datalog example

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

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

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

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

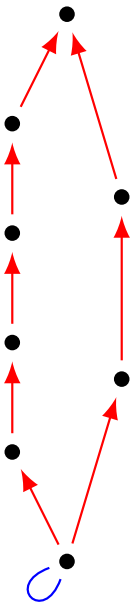
$$S(y,x') :- L(x,y), S(x,x')$$

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

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

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

Instance I:



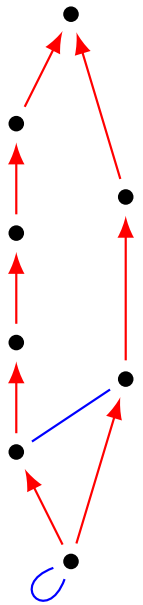
Datalog example

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

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

$$T(x,y) :- S(x,y)$$
$$T(x,z) :- T(x,y), T(y,z)$$
$$\textbf{false} :- T(x,x)$$

Instance I:



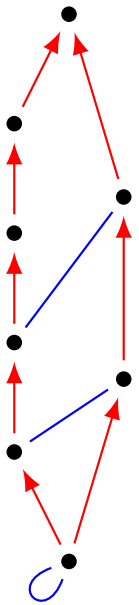
Datalog example

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

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

$$T(x,y) :- S(x,y)$$
$$T(x,z) :- T(x,y), T(y,z)$$
$$\textbf{false} :- T(x,x)$$

Instance I:



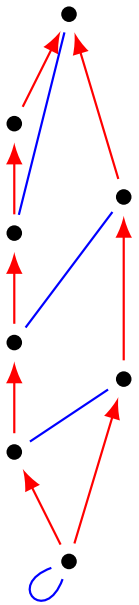
Datalog example

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

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

$$T(x,y) :- S(x,y)$$
$$T(x,z) :- T(x,y), T(y,z)$$
$$\textbf{false} :- T(x,x)$$

Instance I:



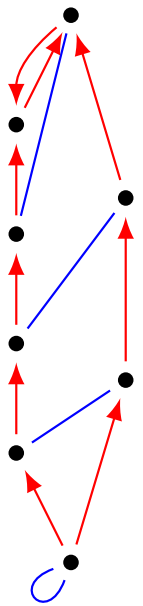
Datalog example

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

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

$$T(x,y) :- S(x,y)$$
$$T(x,z) :- T(x,y), T(y,z)$$
$$\textbf{false} :- T(x,x)$$

Instance I:



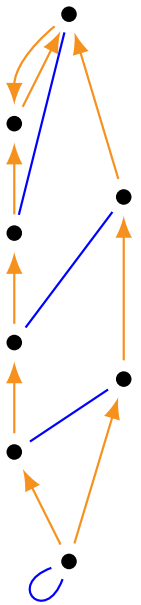
Datalog example

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

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

$$T(x,y) :- S(x,y)$$
$$T(x,z) :- T(x,y), T(y,z)$$
$$\textbf{false} :- T(x,x)$$

Instance I:



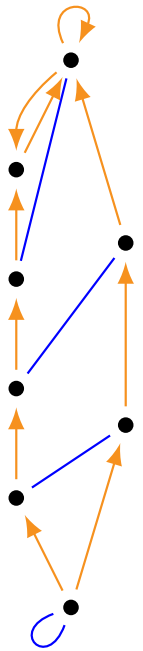
Datalog example

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

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

$$T(x,y) :- S(x,y)$$
$$T(x,z) :- T(x,y), T(y,z)$$
$$\textbf{false} :- T(x,x)$$

Instance I:



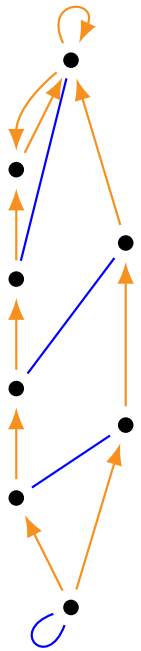
Datalog example

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

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

$$T(x,y) :- S(x,y)$$
$$T(x,z) :- T(x,y), T(y,z)$$
$$\textbf{false} :- T(x,x)$$

Instance I:



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

Datalog example

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

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

Π^{succ} accepts exactly the structures where for each two nodes all connecting paths have the same length.

An observation about Datalog

Datalog program Π with EDB E and IDB R:

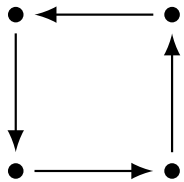
$$R(x, y) :- E(x, y)$$
$$R(x, y) :- R(x, z), R(z, y)$$
$$\textbf{false} :- R(x, x)$$

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 $\mathbf{I}$:

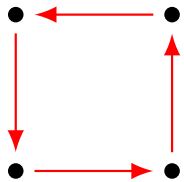


An observation about Datalog

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 $\mathbf{I}$:

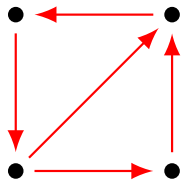


An observation about Datalog

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 $\mathbf{I}$:

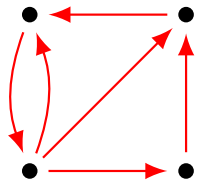


An observation about Datalog

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 $\mathbf{I}$:

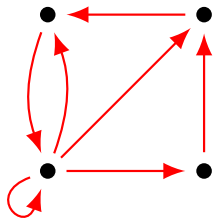


An observation about Datalog

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 $\mathbf{I}$:



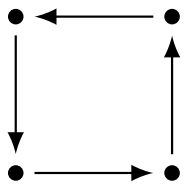
Π derives **false** on $\mathbf{I}$.

An observation about Datalog

Datalog program Π with EDB E and IDB R:

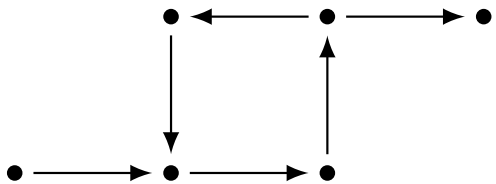
$$R(x,y) :- E(x,y)$$
$$R(x,y) :- R(x,z), R(z,y)$$
$$\textbf{false} :- R(x,x)$$

Instance **I**:



homomorphism

Instance **J**:



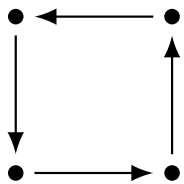
Π derives **false** on **I**.

An observation about Datalog

Datalog program Π with EDB E and IDB R:

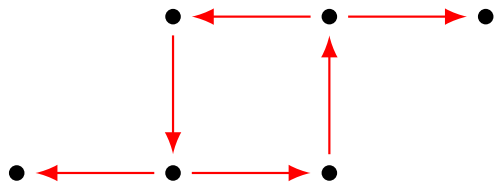
$$R(x,y) :- E(x,y)$$
$$R(x,y) :- R(x,z), R(z,y)$$
$$\textbf{false} :- R(x,x)$$

Instance **I**:



homomorphism

Instance **J**:



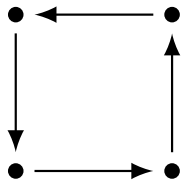
Π derives **false** on **I**.

An observation about Datalog

Datalog program Π with EDB E and IDB R:

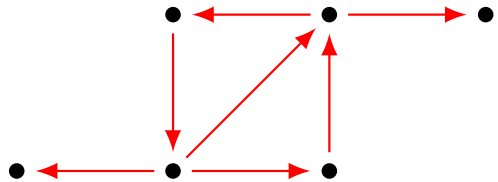
$$R(x,y) :- E(x,y)$$
$$R(x,y) :- R(x,z), R(z,y)$$
$$\textbf{false} :- R(x,x)$$

Instance **I**:



homomorphism

Instance **J**:



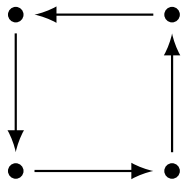
Π derives **false** on **I**.

An observation about Datalog

Datalog program Π with EDB E and IDB R:

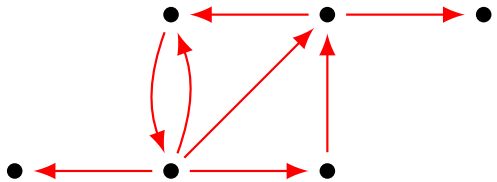
$$R(x,y) :- E(x,y)$$
$$R(x,y) :- R(x,z), R(z,y)$$
$$\textbf{false} :- R(x,x)$$

Instance **I**:



homomorphism

Instance **J**:



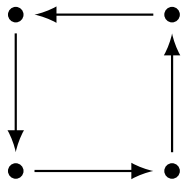
Π derives **false** on **I**.

An observation about Datalog

Datalog program Π with EDB E and IDB R:

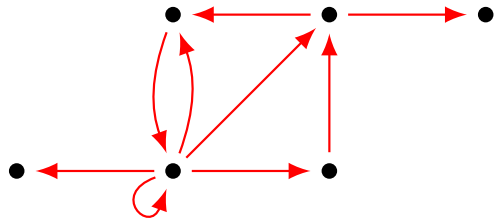
$$R(x,y) :- E(x,y)$$
$$R(x,y) :- R(x,z), R(z,y)$$
$$\textbf{false} :- R(x,x)$$

Instance **I**:



homomorphism

Instance **J**:



Π derives **false** on **I**.

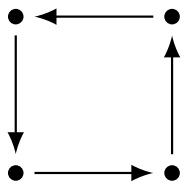
Π derives **false** on **J**.

An observation about Datalog

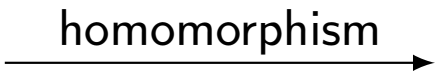
Datalog program Π with EDB E and IDB R:

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

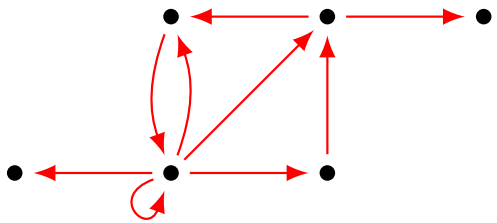
Instance **I**:



Π derives **false** on **I**.



Instance **J**:



Π derives **false** on **J**.

Observation

Let $\mathcal{C}$ be defined by a Datalog program. Let **A** and **B** be structures where **B** is in $\mathcal{C}$ and **A** has a homomorphism to **B**. Then **A** is in $\mathcal{C}$.
We say that $\mathcal{C}$ is closed under inverse homomorphisms.

Result 1

Result 1

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

Result 1

Result 1

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

Definition

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

Result 1

Result 1

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

Definition

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

Example:

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

Result 1

Result 1

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

Definition

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

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 a MSO sentence such that $[\![\Phi]\!]$ is closed under inverse homomorphisms. Then $[\![\Phi]\!]$ is a finite union of MSO definable constraint satisfaction problems.

Definition

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

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 $\mathbf{A}, \mathbf{B} \in \mathcal{C}$, then $\mathbf{A} \uplus \mathbf{B} \in \mathcal{C}$.

Application of Result 1

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

Application of Result 1

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

Datalog program Π:

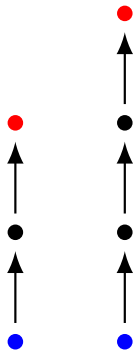
$$L(x, y) :- B(x), B(y)$$
$$L(x, y) :- L(x', y'), E(x', x), E(y', y)$$
$$\textbf{false} :- R(x), L(x, x'), E(x', y)$$

Application of Result 1

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

Datalog program Π:

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

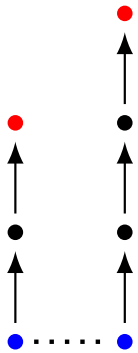


Application of Result 1

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

Datalog program Π:

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

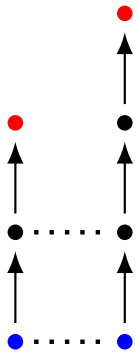


Application of Result 1

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

Datalog program Π:

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

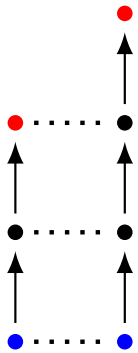


Application of Result 1

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

Datalog program Π:

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

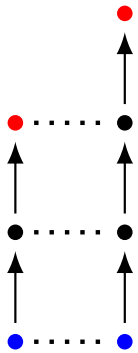


Application of Result 1

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

Datalog program Π:

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



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

Application of Result 1

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

Datalog program Π:

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

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

Application of Result 1

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

Datalog program Π:

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

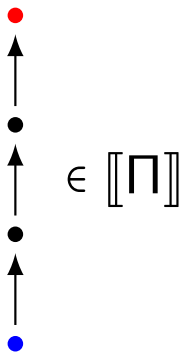
$\in [\![\Pi]\!]$

Application of Result 1

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

Datalog program Π:

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



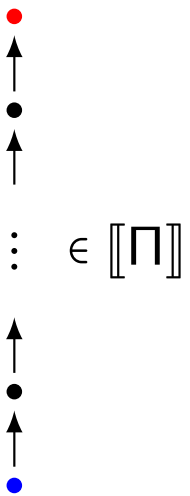
$\in [\![\Pi]\!]$

Application of Result 1

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

Datalog program Π:

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



Application of Result 1

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

Datalog program Π:

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

Observation: There exists an infinite set $X \subset [\![\Pi]\!]$ such that for each distinct elements $\mathbf{A}, \mathbf{B} \in X$ the disjoint union $\mathbf{A} \uplus \mathbf{B}$ 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:

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

Observation: There exists an infinite set $X \subset [\![\Pi]\!]$ such that for each distinct elements $\mathbf{A}, \mathbf{B} \in X$ the disjoint union $\mathbf{A} \uplus \mathbf{B}$ 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:

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

Observation: There exists an infinite set $X \subset [\![\Pi]\!]$ such that for each distinct elements $\mathbf{A}, \mathbf{B} \in X$ the disjoint union $\mathbf{A} \uplus \mathbf{B}$ 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 a MSO sentence such that $[\![\Phi]\!]$ is closed under inverse homomorphisms and disjoint unions. Then $[\![\Phi]\!]$ is the CSP of an ω-categorical structure.

Result 2

Result 2

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

Definition

A countable structure $\mathbf{D}$ is called ω-categorical if its first-order theory has an up to isomorphism unique countable model.

Result 2

Result 2

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

Definition

A countable structure $\mathbf{D}$ is called ω-categorical if its first-order theory has an up to isomorphism unique countable model.

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

Definition

A countable structure $\mathbf{D}$ is called ω-categorical if its first-order theory has an up to isomorphism unique countable model.

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

Definition

A countable structure $\mathbf{D}$ is called ω-categorical if its first-order theory has an up to isomorphism unique countable model.

Remark

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

Application of Result 2

Consider $\mathbf{Z} = (\mathbb{Z}; \mathrm{Succ} = \{(x, y) \mid x + 1 = y\})$ and the Datalog program Π^{succ}.

Application of Result 2

Consider $\mathbf{Z} = (\mathbb{Z}; \text{Succ} = \{(x, y) \mid x + 1 = y\})$ and the Datalog program Π^{succ}.

- $\text{CSP}(\mathbf{Z}) = [\![\Pi^{succ}]\!]$.

Application of Result 2

Consider $\mathbf{Z} = (\mathbb{Z}; \mathrm{Succ} = \{(x, y) \mid x + 1 = y\})$ and the Datalog program Π^{succ}.

- $\mathrm{CSP}(\mathbf{Z}) = [\![\Pi^{succ}]\!]$.
- $\mathbf{Z}$ and $\mathbf{Z} \uplus \mathbf{Z}$ 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

Consider $\mathbf{Z} = (\mathbb{Z}; \mathrm{Succ} = \{(x, y) \mid x + 1 = y\})$ and the Datalog program Π^{succ}.

- $\mathrm{CSP}(\mathbf{Z}) = [\![\Pi^{succ}]\!]$.
- $\mathbf{Z}$ and $\mathbf{Z} \uplus \mathbf{Z}$ satisfy exactly the same first-order sentences, e.g.,

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

- $\mathbf{Z}$ and $\mathbf{Z} \uplus \mathbf{Z}$ are not isomorphic.

Application of Result 2

Consider $\mathbf{Z} = (\mathbb{Z}; \mathrm{Succ} = \{(x, y) \mid x + 1 = y\})$ and the Datalog program Π^{succ}.

- $\mathrm{CSP}(\mathbf{Z}) = [\![\Pi^{succ}]\!]$.
- $\mathbf{Z}$ and $\mathbf{Z} \uplus \mathbf{Z}$ satisfy exactly the same first-order sentences, e.g.,

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

- $\mathbf{Z}$ and $\mathbf{Z} \uplus \mathbf{Z}$ are not isomorphic.
- Therefore $\mathbf{Z}$ is not ω-categorical.

Application of Result 2

Consider $\mathbf{Z} = (\mathbb{Z}; \mathrm{Succ} = \{(x, y) \mid x + 1 = y\})$ and the Datalog program Π^{succ}.

- $\mathrm{CSP}(\mathbf{Z}) = [\![\Pi^{succ}]\!]$.
- $\mathbf{Z}$ and $\mathbf{Z} \uplus \mathbf{Z}$ satisfy exactly the same first-order sentences, e.g.,

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

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

Application of Result 2

Consider $\mathbf{Z} = (\mathbb{Z}; \mathrm{Succ} = \{(x, y) \mid x + 1 = y\})$ and the Datalog program Π^{succ}.

- $\mathrm{CSP}(\mathbf{Z}) = [\![\Pi^{succ}]\!]$.
- $\mathbf{Z}$ and $\mathbf{Z} \uplus \mathbf{Z}$ satisfy exactly the same first-order sentences, e.g.,

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

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

Description of MSO ∩ Datalog

Result 1

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

Description of MSO ∩ Datalog

Result 1

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

Description of MSO ∩ Datalog

Result 1

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

Result 3

Every problem in MSO ∩ Datalog is the finite union of ω-categorical CSPs.

A generalization of the results

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

A generalization of the results

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

A generalization of the results

- Guarded second order logic (GSO) is a generalisation 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 ∩ Datalog. Does the converse also hold?

Open problems

- The logic Nemodeq introduced by Rudolph and Krötzsch is contained in MSO ∩ Datalog. Does the converse also hold?
- Is the intersection of GSO and Datalog describable by some logic?

Open problems

- The logic Nemodeq introduced by Rudolph and Krötzsch is contained in MSO ∩ Datalog. Does the converse also hold?
- Is the intersection of GSO and Datalog describable 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!

Publications

- *Hardness of Network Satisfaction for Relation Algebras with Normal Representations*, Manuel Bodirsky and Simon Knäuer.
  In Proceedings of RAMICS 2020.
- *ASNP: A Tame Fragment of Existential Second-Order Logic*
  Manuel Bodirsky, Simon Knäuer and Florian Starke
  In Proceedings of CiE 2020.
- *Network satisfaction for symmetric relation algebras with a flexible atom*,
  Manuel Bodirsky and Simon Knäuer.
  Submitted.
- *Datalog-Expressibility for Monadic and Guarded Second-Order Logic*
  Manuel Bodirsky, Simon Knäuer and Sebastian Rudolph
  Submitted.

Result 1 (proof idea)

Result 1

Let Φ be a 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 a 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 $\mathbf{A} \sim \mathbf{B}$ if and only if

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

Result 1 (proof idea)

Result 1

Let Φ be a 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 $\mathbf{A} \sim \mathbf{B}$ if and only if

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

- $\sim$ has finitely many classes.

Result 1 (proof idea)

Result 1

Let Φ be a 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 $\mathbf{A} \sim \mathbf{B}$ if and only if

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

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

Result 1 (proof idea)

Result 1

Let Φ be a 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 $\mathbf{A} \sim \mathbf{B}$ if and only if

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

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

Result 1 (proof idea)

Result 1

Let Φ be a 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 $\mathbf{A} \sim \mathbf{B}$ if and only if

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

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

Result 1 (proof idea)

Result 1

Let Φ be a 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 $\mathbf{A} \sim \mathbf{B}$ if and only if

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

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

Result 1 (proof idea)

Result 1

Let Φ be a 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 $\mathbf{A} \sim \mathbf{B}$ if and only if

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

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

Result 1 (proof idea)

Result 1

Let Φ be a 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 $\mathbf{A} \sim \mathbf{B}$ if and only if

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

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

Result 1 (proof idea)

Result 1

Let Φ be a 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 $\mathbf{A} \sim \mathbf{B}$ if and only if

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

- $\sim$ 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)

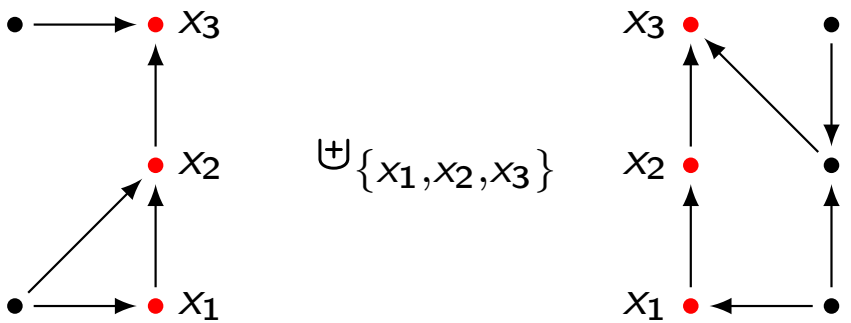
Generalize the equivalence relation $\sim$:

- 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 such that their τ -reducts are from $\mathcal{C}$.

Result 2 (proof idea)

Generalize the equivalence relation $\sim$:

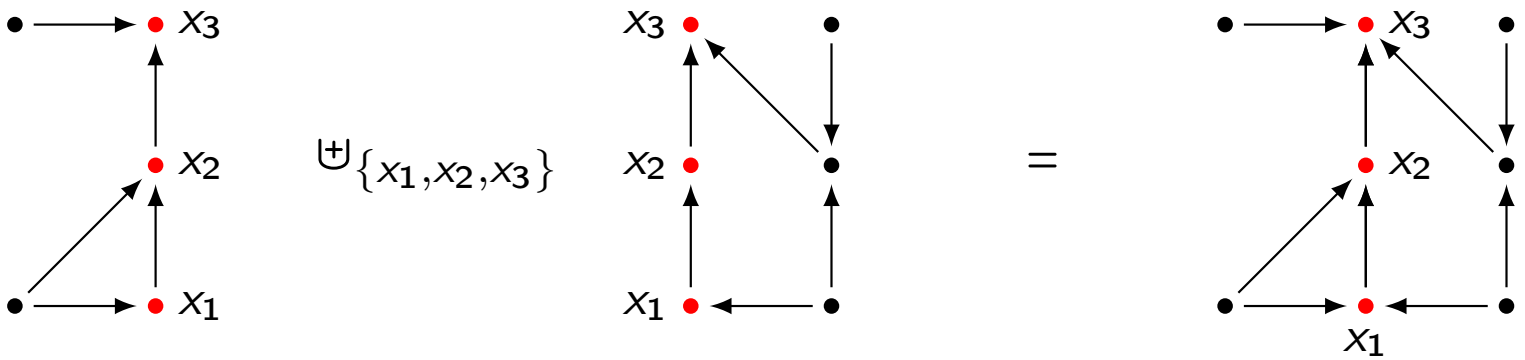
- Let $\mathcal{C} = [\![\Phi]\!]$ and let X be a set of new constant symbols, $|X| = n$.
- Let $\mathcal{C}^X$ be the class of $\tau \cup X$-structures such that their τ -reducts are from $\mathcal{C}$.
- For two structures $\mathbf{A}, \mathbf{B} \in \mathcal{C}^X$ the structure $\mathbf{A} \uplus_X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



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 such that their τ -reducts are from $\mathcal{C}$.
- For two structures $\mathbf{A}, \mathbf{B} \in \mathcal{C}^X$ the structure $\mathbf{A} \uplus_X \mathbf{B}$ is defined as the pairwise identification of the constants X in $\mathbf{A} \uplus \mathbf{B}$.



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 such that their τ -reducts are from $\mathcal{C}$.
- For two structures $\mathbf{A}, \mathbf{B} \in \mathcal{C}^X$ the structure $\mathbf{A} \uplus_X \mathbf{B}$ is defined as the pairwise identification of the constants X in $\mathbf{A} \uplus \mathbf{B}$.
- Consider equivalence relation $\sim_n^{\mathcal{C}}$ on $\mathcal{C}^X$ with $\mathbf{A} \sim_n^{\mathcal{C}} \mathbf{B}$ if and only if

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

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 such that their τ -reducts are from $\mathcal{C}$.
- For two structures $\mathbf{A}, \mathbf{B} \in \mathcal{C}^X$ the structure $\mathbf{A} \uplus_X \mathbf{B}$ is defined as the pairwise identification of the constants X in $\mathbf{A} \uplus \mathbf{B}$.
- Consider equivalence relation $\sim_n^{\mathcal{C}}$ on $\mathcal{C}^X$ with $\mathbf{A} \sim_n^{\mathcal{C}} \mathbf{B}$ if and only if

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

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 $\mathbf{B}$ such that $\mathrm{CSP}(\mathbf{B}) = \mathcal{C}$ if and only if $\sim_n^{\mathcal{C}}$ has finitely many equivalence classes for each $n \in \mathbb{N}$.