Model Checking of Conjunctive Queries

Simone Bova

Vanderbilt University

Vienna University of Technology
March 23, 2012

Outline

Outline

Model Checking | Example

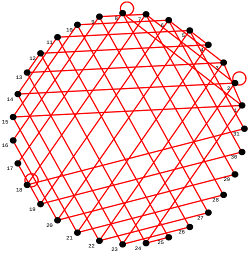


Figure: $G = (\{1, \ldots, 31\}, \{(a, b) \mid a + b \text{ perfect square}\})$.

Model Checking | Example

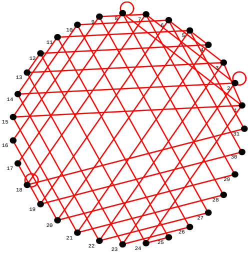


Figure: $G = (\{1, \ldots, 31\}, \{(a, b) \mid a + b \text{ perfect square}\})$. Is G Hamiltonian?

Model Checking | Example

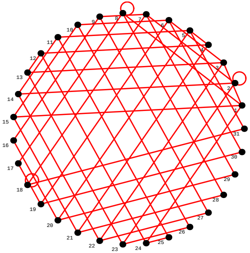


Figure: $G = (\{1, \ldots, 31\}, \{(a, b) \mid a + b \text{ perfect square}\})$. Is G Hamiltonian?

Problem HAMILTONICITY

Model Checking | Example

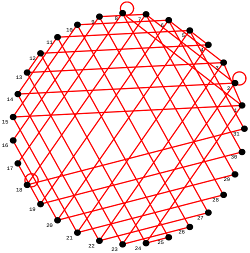


Figure: $G = (\{1, \ldots, 31\}, \{(a, b) \mid a + b \text{ perfect square}\})$. Is G Hamiltonian?

Problem HAMILTONICITY

Instance A *finite* relational $\{E\}$-structure $\mathbf{G} = (G, E^{\mathbf{G}})$ with $E^{\mathbf{G}} \subseteq G^2$.

Model Checking | Example

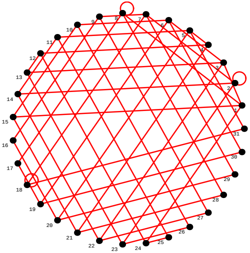


Figure: $G = (\{1, \ldots, 31\}, \{(a, b) \mid a + b \text{ perfect square}\})$. Is G Hamiltonian?

Problem	HAMILTONICITY
Instance	A *finite* relational $\{E\}$-structure $\mathbf{G} = (G, E^{\mathbf{G}})$ with $E^{\mathbf{G}} \subseteq G^2$.
Question	Is ϕ true in $\mathbf{G}$?

$$\phi = \exists X \exists Y (\text{“}X \text{ total order relation”} \wedge \text{“}Y \setminus \{(\max, \min)\} \text{ cover relation of } X\text{”} \\ \wedge \forall x \forall y (Yxy \rightarrow Exy)).$$

Model Checking | Logic

$\sigma = \{R_1, \ldots, R_k\}$ is a finite finitary relational signature, that is,
R_i is a finitary relation symbol of arity $\mathrm{ar}(R_i) \in \mathbb{N}$ for $i = 1, \ldots, k$.

$\mathbf{A} = (A, R_1^{\mathbf{A}}, \ldots, R_k^{\mathbf{A}})$ is a σ-structure, that is,
A is a *finite* nonempty set (universe),
$R_i^{\mathbf{A}} \subseteq A^{\mathrm{ar}(R_i)}$ is an $\mathrm{ar}(R_i)$-ary relation on A ($i = 1, \ldots, k$).

ϕ is a second-order σ-sentence.

Fact
For all σ-sentences ϕ and σ-structures $\mathbf{A}$,
$\mathbf{A} \models \phi$ xor $\mathbf{A} \not\models \phi$ (ϕ is true xor false in $\mathbf{A}$).

Model Checking | *Logic*

Example

$\sigma = \{E\}$ with $\mathrm{ar}(E) = 2$.

$\{\mathbf{A} \mid \mathbf{A}$ is $\{E\}$-structure$\}$ is the class of finite directed graphs.

Model Checking | Logic

Example

$\sigma = \{E\}$ with $\mathrm{ar}(E) = 2$.

$\{\mathbf{A} \mid \mathbf{A}$ is $\{E\}$-structure$\}$ is the class of finite directed graphs.

$\mathbf{A} \models \exists x z_1 z_2 z_3 (Exz_1 \wedge Ez_1z_2 \wedge Ez_2z_3)$ iff

Model Checking | Logic

Example

$\sigma = \{E\}$ with $\mathrm{ar}(E) = 2$.

$\{\mathbf{A} \mid \mathbf{A}$ is $\{E\}$-structure$\}$ is the class of finite directed graphs.

$\mathbf{A} \models \exists x z_1 z_2 z_3 (Exz_1 \wedge Ez_1z_2 \wedge Ez_2z_3)$ iff $\mathbf{A}$ contains a 3-edge (non simple) path.

Model Checking | Logic

Example

$\sigma = \{E\}$ with $\mathrm{ar}(E) = 2$.

$\{\mathbf{A} \mid \mathbf{A}$ is $\{E\}$-structure$\}$ is the class of finite directed graphs.

$\mathbf{A} \models \exists x z_1 z_2 z_3 (Exz_1 \wedge Ez_1z_2 \wedge Ez_2z_3)$ iff $\mathbf{A}$ contains a 3-edge (non simple) path.

$\exists x z_1 z_2 z_3 (Exz_1 \wedge Ez_1z_2 \wedge Ez_2z_3)$ is a *conjunctive query*,
or *primitive positive* first-order sentence ($\mathcal{PP} \subseteq \mathcal{FO}$).

Model Checking | Logic

Example

$\sigma = \{E\}$ with $\mathrm{ar}(E) = 2$.

$\{\mathbf{A} \mid \mathbf{A}$ is $\{E\}$-structure$\}$ is the class of finite directed graphs.

$\mathbf{A} \models \exists x z_1 z_2 z_3 (E x z_1 \wedge E z_1 z_2 \wedge E z_2 z_3)$ iff $\mathbf{A}$ contains a 3-edge (non simple) path.

$\exists x z_1 z_2 z_3 (E x z_1 \wedge E z_1 z_2 \wedge E z_2 z_3)$ is a *conjunctive query*,
or *primitive positive* first-order sentence ($\mathcal{PP} \subseteq \mathcal{FO}$).

$\mathbf{A} \models \exists X \forall x (Xx \vee \exists y (Xy \wedge Eyx))$ iff

Model Checking | Logic

Example

$\sigma = \{E\}$ with $\mathrm{ar}(E) = 2$.

$\{\mathbf{A} \mid \mathbf{A} \text{ is } \{E\}\text{-structure}\}$ is the class of finite directed graphs.

$\mathbf{A} \models \exists x z_1 z_2 z_3 (Exz_1 \wedge Ez_1z_2 \wedge Ez_2z_3)$ iff $\mathbf{A}$ contains a 3-edge (non simple) path.

$\exists x z_1 z_2 z_3 (Exz_1 \wedge Ez_1z_2 \wedge Ez_2z_3)$ is a *conjunctive query*,
or *primitive positive* first-order sentence ($\mathcal{PP} \subseteq \mathcal{FO}$).

$\mathbf{A} \models \exists X \forall x (Xx \vee \exists y (Xy \wedge Eyx))$ iff $\mathbf{A}$ has a dominating set.

Model Checking | Logic

Example

$\sigma = \{E\}$ with $\mathrm{ar}(E) = 2$.

$\{\mathbf{A} \mid \mathbf{A}$ is $\{E\}$-structure$\}$ is the class of finite directed graphs.

$\mathbf{A} \models \exists x z_1 z_2 z_3 (Exz_1 \wedge Ez_1z_2 \wedge Ez_2z_3)$ iff $\mathbf{A}$ contains a 3-edge (non simple) path.

$\exists x z_1 z_2 z_3 (Exz_1 \wedge Ez_1z_2 \wedge Ez_2z_3)$ is a *conjunctive query*,
or *primitive positive* first-order sentence ($\mathcal{PP} \subseteq \mathcal{FO}$).

$\mathbf{A} \models \exists X \forall x (Xx \vee \exists y (Xy \wedge Eyx))$ iff $\mathbf{A}$ has a dominating set.

$\{\mathbf{A} \mid \mathbf{A}$ is $\{E\}$-structure st $\mathbf{A} \models \forall x \neg Exx \wedge \forall xy (Exy \rightarrow Eyx)\}$,

Model Checking | Logic

Example

$\sigma = \{E\}$ with $\mathrm{ar}(E) = 2$.

$\{\mathbf{A} \mid \mathbf{A}$ is $\{E\}$-structure$\}$ is the class of finite directed graphs.

$\mathbf{A} \models \exists x z_1 z_2 z_3 (Exz_1 \wedge Ez_1z_2 \wedge Ez_2z_3)$ iff $\mathbf{A}$ contains a 3-edge (non simple) path.

$\exists x z_1 z_2 z_3 (Exz_1 \wedge Ez_1z_2 \wedge Ez_2z_3)$ is a *conjunctive query*,
or *primitive positive* first-order sentence ($\mathcal{PP} \subseteq \mathcal{FO}$).

$\mathbf{A} \models \exists X \forall x (Xx \vee \exists y (Xy \wedge Eyx))$ iff $\mathbf{A}$ has a dominating set.

$\{\mathbf{A} \mid \mathbf{A}$ is $\{E\}$-structure st $\mathbf{A} \models \forall x \neg Exx \wedge \forall xy (Exy \rightarrow Eyx)\}$,
the class of finite simple (loopless undirected) graphs.

Model Checking | Problem

Problem $\textsc{ModelCheck}(\mathcal{L}, \mathcal{C})$,
where $\mathcal{L}$ is a class of σ-sentences,
and $\mathcal{C}$ is a class of σ-structures.

Instance A σ-sentence ϕ in $\mathcal{L}$ and a σ-structure $\mathbf{A} \in \mathcal{C}$.

Question $\mathbf{A} \models \phi$?

Model Checking | Problem

Problem	ModelCheck$(\mathcal{L}, \mathcal{C})$, where $\mathcal{L}$ is a class of σ-sentences, and $\mathcal{C}$ is a class of σ-structures.
Instance	A σ-sentence ϕ in $\mathcal{L}$ and a σ-structure $\mathbf{A} \in \mathcal{C}$.
Question	$\mathbf{A} \models \phi$?

Remark
ϕ *encoding has size* k, $\mathbf{A}$ *encoding has size* $n = (|A| + 1) + \sum_{R \in \sigma} |A|^{\mathrm{ar}(R)}$.

Model Checking | Combined Complexity

How hard is $\textsc{ModelCheck}(\mathcal{FO}, \mathcal{C})$ computationally,
where $\mathcal{FO}$ is the class of first-order σ-sentences,
and $\mathcal{C}$ is the class of finite σ-structures?

Model Checking | Combined Complexity

How hard is MODELCHECK($\mathcal{FO}, \mathcal{C}$) computationally,
where $\mathcal{FO}$ is the class of first-order σ-sentences,
and $\mathcal{C}$ is the class of finite σ-structures? Hard.

Model Checking | Combined Complexity

How hard is $\textsc{ModelCheck}(\mathcal{FO}, \mathcal{C})$ computationally,
where $\mathcal{FO}$ is the class of first-order σ-sentences,
and $\mathcal{C}$ is the class of finite σ-structures? Hard.

Theorem
The combined complexity of $\textsc{ModelCheck}(\mathcal{FO}, \mathcal{C})$ *is PSPACE-complete.*

Proof (Idea).
Wts $\{(\phi, \mathbf{A}) \in \mathcal{FO} \times \mathcal{C} \mid \mathbf{A} \models \phi\}$ *is PSPACE-complete.*
Checking $\mathbf{A} \models \phi$ *is feasible in space* $O(kn)$.
A quantified Boolean formula $Q_1 x_1 \cdots Q_n x_n \phi(x_1, \ldots, x_n)$ *is true iff*
$\mathbf{A} \models Q_1 x_1 \cdots Q_n x_n \phi'$ *where* $\mathbf{A} = (\{0, 1\}, P^{\mathbf{A}})$, $P^{\mathbf{A}} = \{1\}$, *and*
$\phi' = \phi[x_i / Px_i \mid i = 1, \ldots, n]$. □

Model Checking | Combined Complexity

How hard is $\textsc{ModelCheck}(\mathcal{FO}, \mathcal{C})$ computationally,
where $\mathcal{FO}$ is the class of first-order σ-sentences,
and $\mathcal{C}$ is the class of finite σ-structures? Hard.

Theorem
The combined complexity of $\textsc{ModelCheck}(\mathcal{FO}, \mathcal{C})$ *is PSPACE-complete.*

Proof (Idea).
Wts $\{(\phi, \mathbf{A}) \in \mathcal{FO} \times \mathcal{C} \mid \mathbf{A} \models \phi\}$ *is PSPACE-complete.*
Checking $\mathbf{A} \models \phi$ *is feasible in space* $O(kn)$.
A quantified Boolean formula $Q_1 x_1 \cdots Q_n x_n \phi(x_1, \ldots, x_n)$ *is true iff*
$\mathbf{A} \models Q_1 x_1 \cdots Q_n x_n \phi'$ *where* $\mathbf{A} = (\{0,1\}, P^{\mathbf{A}})$, $P^{\mathbf{A}} = \{1\}$, *and*
$\phi' = \phi[x_i / P x_i \mid i = 1, \ldots, n]$. $\square$

Feasible (and useful) restrictions to $\textsc{ModelCheck}(\mathcal{FO}, \mathcal{C})$?

Model Checking | Restricting MODELCHECK($\mathcal{FO}, \mathcal{C}$)

Restrictions to MODELCHECK($\mathcal{FO}, \mathcal{C}$) $= \{(\phi, \mathbf{A}) \in \mathcal{FO} \times \mathcal{C} \mid \mathbf{A} \models \phi\}$:

Effects on complexity:

Model Checking | *Restricting* MODELCHECK$(\mathcal{FO}, \mathcal{C})$

Restrictions to MODELCHECK$(\mathcal{FO}, \mathcal{C}) = \{(\phi, \mathbf{A}) \in \mathcal{FO} \times \mathcal{C} \mid \mathbf{A} \models \phi\}$:

1 Fix $\phi \in \mathcal{FO}$ as a parameter, ie study $\{\mathbf{A} \mid \mathbf{A} \models \phi\} \subseteq \mathcal{C}$.

Effects on complexity:

Model Checking | *Restricting* ModelCheck($\mathcal{FO}, \mathcal{C}$)

Restrictions to ModelCheck($\mathcal{FO}, \mathcal{C}$) $= \{(\phi, \mathbf{A}) \in \mathcal{FO} \times \mathcal{C} \mid \mathbf{A} \models \phi\}$:

1. Fix $\phi \in \mathcal{FO}$ as a parameter, ie study $\{\mathbf{A} \mid \mathbf{A} \models \phi\} \subseteq \mathcal{C}$.

Effects on complexity:

1. *Data* complexity of ModelCheck($\mathcal{FO}, \mathcal{C}$) is $O(kn^k)$ time.

Model Checking | *Restricting* ModelCheck$(\mathcal{FO}, \mathcal{C})$

Restrictions to ModelCheck$(\mathcal{FO}, \mathcal{C}) = \{(\phi, \mathbf{A}) \in \mathcal{FO} \times \mathcal{C} \mid \mathbf{A} \models \phi\}$:

1. Fix $\phi \in \mathcal{FO}$ as a parameter, ie study $\{\mathbf{A} \mid \mathbf{A} \models \phi\} \subseteq \mathcal{C}$.
2. Fix $\mathbf{A} \in \mathcal{C}$ as a parameter, ie study $\{\phi \mid \mathbf{A} \models \phi\} \subseteq \mathcal{FO}$.

Effects on complexity:

1. *Data* complexity of ModelCheck$(\mathcal{FO}, \mathcal{C})$ is $O(kn^k)$ time.

Model Checking | *Restricting* MODELCHECK($\mathcal{FO}, \mathcal{C}$)

Restrictions to MODELCHECK($\mathcal{FO}, \mathcal{C}$) $= \{(\phi, \mathbf{A}) \in \mathcal{FO} \times \mathcal{C} \mid \mathbf{A} \models \phi\}$:

1. Fix $\phi \in \mathcal{FO}$ as a parameter, ie study $\{\mathbf{A} \mid \mathbf{A} \models \phi\} \subseteq \mathcal{C}$.
2. Fix $\mathbf{A} \in \mathcal{C}$ as a parameter, ie study $\{\phi \mid \mathbf{A} \models \phi\} \subseteq \mathcal{FO}$.

Effects on complexity:

1. *Data* complexity of MODELCHECK($\mathcal{FO}, \mathcal{C}$) is $O(kn^k)$ time.
2. *Expression* complexity of MODELCHECK($\mathcal{FO}, \mathcal{C}$) is PSPACE-complete.

Model Checking | *Restricting* ModelCheck$(\mathcal{FO}, \mathcal{C})$

Restrictions to ModelCheck$(\mathcal{FO}, \mathcal{C}) = \{(\phi, \mathbf{A}) \in \mathcal{FO} \times \mathcal{C} \mid \mathbf{A} \models \phi\}$:

1. Fix $\phi \in \mathcal{FO}$ as a parameter, ie study $\{\mathbf{A} \mid \mathbf{A} \models \phi\} \subseteq \mathcal{C}$.
2. Fix $\mathbf{A} \in \mathcal{C}$ as a parameter, ie study $\{\phi \mid \mathbf{A} \models \phi\} \subseteq \mathcal{FO}$.
3. Study ModelCheck$(\mathcal{L}, \mathcal{S})$ with $\mathcal{L} \subseteq \mathcal{FO}$ or $\mathcal{S} \subseteq \mathcal{C}$.

Effects on complexity:

1. *Data* complexity of ModelCheck$(\mathcal{FO}, \mathcal{C})$ is $O(kn^k)$ time.
2. *Expression* complexity of ModelCheck$(\mathcal{FO}, \mathcal{C})$ is PSPACE-complete.

Model Checking | *Restricting* $\textsc{ModelCheck}(\mathcal{FO}, \mathcal{C})$

Restrictions to $\textsc{ModelCheck}(\mathcal{FO}, \mathcal{C}) = \{(\phi, \mathbf{A}) \in \mathcal{FO} \times \mathcal{C} \mid \mathbf{A} \models \phi\}$:

1 Fix $\phi \in \mathcal{FO}$ as a parameter, ie study $\{\mathbf{A} \mid \mathbf{A} \models \phi\} \subseteq \mathcal{C}$.

2 Fix $\mathbf{A} \in \mathcal{C}$ as a parameter, ie study $\{\phi \mid \mathbf{A} \models \phi\} \subseteq \mathcal{FO}$.

3 Study $\textsc{ModelCheck}(\mathcal{L}, \mathcal{S})$ with $\mathcal{L} \subseteq \mathcal{FO}$ or $\mathcal{S} \subseteq \mathcal{C}$.

Effects on complexity:

1 *Data* complexity of $\textsc{ModelCheck}(\mathcal{FO}, \mathcal{C})$ is $O(kn^k)$ time.

2 *Expression* complexity of $\textsc{ModelCheck}(\mathcal{FO}, \mathcal{C})$ is PSPACE-complete.

1&3 Data complexity of $\textsc{ModelCheck}(\mathcal{FO}, \mathcal{T})$ is $O(f(k)n)$ time, where $\mathcal{T}$ is the class of finite (simple) trees.

Model Checking | Restricting $\textsc{ModelCheck}(\mathcal{FO}, \mathcal{C})$

Restrictions to $\textsc{ModelCheck}(\mathcal{FO}, \mathcal{C}) = \{(\phi, \mathbf{A}) \in \mathcal{FO} \times \mathcal{C} \mid \mathbf{A} \models \phi\}$:

1. Fix $\phi \in \mathcal{FO}$ as a parameter, ie study $\{\mathbf{A} \mid \mathbf{A} \models \phi\} \subseteq \mathcal{C}$.
2. Fix $\mathbf{A} \in \mathcal{C}$ as a parameter, ie study $\{\phi \mid \mathbf{A} \models \phi\} \subseteq \mathcal{FO}$.
3. Study $\textsc{ModelCheck}(\mathcal{L}, \mathcal{S})$ with $\mathcal{L} \subseteq \mathcal{FO}$ or $\mathcal{S} \subseteq \mathcal{C}$.

Effects on complexity:

1. *Data* complexity of $\textsc{ModelCheck}(\mathcal{FO}, \mathcal{C})$ is $O(kn^k)$ time.
2. *Expression* complexity of $\textsc{ModelCheck}(\mathcal{FO}, \mathcal{C})$ is PSPACE-complete.

1&3 Data complexity of $\textsc{ModelCheck}(\mathcal{FO}, \mathcal{T})$ is $O(f(k)n)$ time, where $\mathcal{T}$ is the class of finite (simple) trees.

2&3 Expression complexity of $\textsc{ModelCheck}(\mathcal{PP}, \mathcal{B})$ is polytime, where $\mathcal{PP} \subseteq \mathcal{FO}$ is the class of conjunctive queries, and $\mathcal{B}$ is the class of finite (simple) bipartite graphs.

Model Checking | *Restricting* MODELCHECK($\mathcal{FO}, \mathcal{C}$)

Restrictions to MODELCHECK($\mathcal{FO}, \mathcal{C}$) $= \{(\phi, \mathbf{A}) \in \mathcal{FO} \times \mathcal{C} \mid \mathbf{A} \models \phi\}$:

1 Fix $\phi \in \mathcal{FO}$ as a parameter, ie study $\{\mathbf{A} \mid \mathbf{A} \models \phi\} \subseteq \mathcal{C}$.

2 Fix $\mathbf{A} \in \mathcal{C}$ as a parameter, ie study $\{\phi \mid \mathbf{A} \models \phi\} \subseteq \mathcal{FO}$.

3 Study MODELCHECK($\mathcal{L}, \mathcal{S}$) with $\mathcal{L} \subseteq \mathcal{FO}$ or $\mathcal{S} \subseteq \mathcal{C}$.

Effects on complexity:

1 *Data* complexity of MODELCHECK($\mathcal{FO}, \mathcal{C}$) is $O(kn^k)$ time.

2 *Expression* complexity of MODELCHECK($\mathcal{FO}, \mathcal{C}$) is PSPACE-complete.

1&3 Data complexity of MODELCHECK($\mathcal{FO}, \mathcal{T}$) is $O(f(k)n)$ time, where $\mathcal{T}$ is the class of finite (simple) trees. *Not this talk.*

2&3 Expression complexity of MODELCHECK($\mathcal{PP}, \mathcal{B}$) is polytime, where $\mathcal{PP} \subseteq \mathcal{FO}$ is the class of conjunctive queries, and $\mathcal{B}$ is the class of finite (simple) bipartite graphs.

Model Checking | *Restricting* MODELCHECK($\mathcal{FO}, \mathcal{C}$)

Restrictions to MODELCHECK($\mathcal{FO}, \mathcal{C}$) $= \{(\phi, \mathbf{A}) \in \mathcal{FO} \times \mathcal{C} \mid \mathbf{A} \models \phi\}$:

1. Fix $\phi \in \mathcal{FO}$ as a parameter, ie study $\{\mathbf{A} \mid \mathbf{A} \models \phi\} \subseteq \mathcal{C}$.
2. Fix $\mathbf{A} \in \mathcal{C}$ as a parameter, ie study $\{\phi \mid \mathbf{A} \models \phi\} \subseteq \mathcal{FO}$.
3. Study MODELCHECK($\mathcal{L}, \mathcal{S}$) with $\mathcal{L} \subseteq \mathcal{FO}$ or $\mathcal{S} \subseteq \mathcal{C}$.

Effects on complexity:

1. *Data* complexity of MODELCHECK($\mathcal{FO}, \mathcal{C}$) is $O(kn^k)$ time.
2. *Expression* complexity of MODELCHECK($\mathcal{FO}, \mathcal{C}$) is PSPACE-complete.

1&3 Data complexity of MODELCHECK($\mathcal{FO}, \mathcal{T}$) is $O(f(k)n)$ time, where $\mathcal{T}$ is the class of finite (simple) trees. *Not this talk.*

2&3 Expression complexity of MODELCHECK($\mathcal{PP}, \mathcal{B}$) is polytime, where $\mathcal{PP} \subseteq \mathcal{FO}$ is the class of conjunctive queries, and $\mathcal{B}$ is the class of finite (simple) bipartite graphs. *This talk.*

Outline

(Nonuniform) CSP | Problem

Problem $\textsc{ModelCheck}(\mathcal{PP}, \mathcal{C})$,
where $\mathcal{PP} \subseteq \mathcal{FO}$ is the class of conjunctive queries (over σ),
and $\mathcal{C}$ is the class of σ-structures.

Instance A σ-sentence ϕ in $\mathcal{PP}$ and a σ-structure $\mathbf{A} \in \mathcal{C}$.

Question $\mathbf{A} \models \phi$?

Remark
$\mathrm{CSP}(\mathbf{A})$ *is a shortcut for* $\{\phi \mid \mathbf{A} \models \phi\} \subseteq \mathcal{PP}$.

CSP | Hardness

Theorem

The expression complexity of $\textsc{ModelCheck}(\mathcal{PP}, \mathcal{C})$ *is NP-complete.*

Proof (Idea).

Checking $\mathbf{A} \models \phi$ *with* $\phi \in \mathcal{PP}$ *is in NP (in fact, wrt to both k and n) for all* $\mathbf{A} \in \mathcal{C}$*, then the (combined so) expression complexity of* $\textsc{ModelCheck}(\mathcal{PP}, \mathcal{C})$ *is in NP.*

$3\textsc{Col} \leq_p^m \{\phi \mid \mathbf{C}_3 \models \phi\}$, *where* $\mathbf{C}_3 = (C_3, I^{\mathbf{C}_3})$,

$C_3 = \{\textcolor{red}{\bullet}, \textcolor{blue}{\bullet}, \textcolor{green}{\bullet}\}$, $I^{\mathbf{C}_3} = \{(\textcolor{red}{\bullet}, \textcolor{blue}{\bullet}), (\textcolor{red}{\bullet}, \textcolor{green}{\bullet}), (\textcolor{blue}{\bullet}, \textcolor{red}{\bullet}), (\textcolor{blue}{\bullet}, \textcolor{green}{\bullet}), (\textcolor{green}{\bullet}, \textcolor{red}{\bullet}), (\textcolor{green}{\bullet}, \textcolor{blue}{\bullet})\}$. $\square$

CSP | Hardness

Theorem

The expression complexity of MODELCHECK($\mathcal{PP}, \mathcal{C}$) *is NP-complete.*

Proof (Idea).

Checking $\mathbf{A} \models \phi$ *with* $\phi \in \mathcal{PP}$ *is in NP (in fact, wrt to both k and n) for all* $\mathbf{A} \in \mathcal{C}$, *then the (combined so) expression complexity of* MODELCHECK($\mathcal{PP}, \mathcal{C}$) *is in NP.*
3COL $\leq_p^m \{\phi \mid \mathbf{C}_3 \models \phi\}$, *where* $\mathbf{C}_3 = (C_3, I^{\mathbf{C}_3})$,
$C_3 = \{\color{red}{\bullet}, \color{blue}{\bullet}, \color{green}{\bullet}\}$, $I^{\mathbf{C}_3} = \{(\color{red}{\bullet}, \color{blue}{\bullet}), (\color{red}{\bullet}, \color{green}{\bullet}), (\color{blue}{\bullet}, \color{red}{\bullet}), (\color{blue}{\bullet}, \color{green}{\bullet}), (\color{green}{\bullet}, \color{red}{\bullet}), (\color{green}{\bullet}, \color{blue}{\bullet})\}$. $\square$

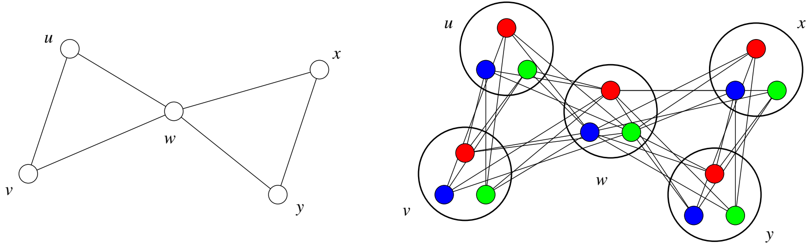


Figure: $G \mapsto \phi_G = \exists u \exists v \exists w \exists x \exists y (Iwu \wedge Iwv \wedge Iwx \wedge Iwy \wedge Iuv \wedge Ixy)$.

CSP | Tractability

Fact

CSP($\mathbf{C}_2$) *is in P, where* $\mathbf{C}_2 = (C_2, I^{\mathbf{C}_2})$,
$C_2 = \{\bullet, \bullet\}$, $I^{\mathbf{C}_2} = \{(\bullet, \bullet), (\bullet, \bullet)\}$.

Proof.

$\{\phi \mid \mathbf{C}_2 \models \phi\} \leq_p^m$ 2Col. $\square$

CSP | Tractability

Fact

CSP($\mathbf{C}_2$) *is in P, where* $\mathbf{C}_2 = (C_2, I^{\mathbf{C}_2})$,
$C_2 = \{{\color{red}\bullet}, {\color{green}\bullet}\}$, $I^{\mathbf{C}_2} = \{({\color{red}\bullet}, {\color{green}\bullet}), ({\color{green}\bullet}, {\color{red}\bullet})\}$.

Proof.

$\{\phi \mid \mathbf{C}_2 \models \phi\} \leq_p^m$ 2COL. $\square$

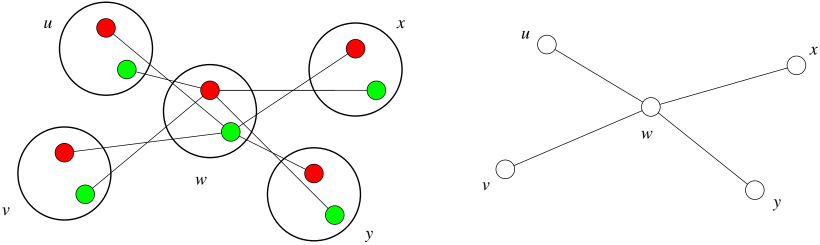


Figure: $\exists u \exists v \exists w \exists x \exists y (Iwu \wedge Iwv \wedge Iwx \wedge Iwy) = \phi \mapsto G_\phi$.

CSP | Dichotomy

Hard problem.

*Ie, avoids NPI, nonobvious if P≠NP.

CSP | Dichotomy

Hard problem. Find some useful easy cases…

*Ie, avoids NPI, nonobvious if P≠NP.

CSP | Dichotomy

Hard problem. Find some useful easy cases...

Theorem (Hell and Nešetřil)

Let $\mathcal{C}$ be the class of finite connected simple graphs and $\mathbf{A} \in \mathcal{C}$.
Then $\mathrm{CSP}(\mathbf{A})$ is in P if $\mathbf{A}$ is bipartite, and NP-complete otherwise.

---

*Ie, avoids NPI, nonobvious if P≠NP.

CSP | Dichotomy

Hard problem. Find some useful easy cases…

Theorem (Hell and Nešetřil)
Let $\mathcal{C}$ be the class of finite connected simple graphs and $\mathbf{A} \in \mathcal{C}$.
Then $\mathrm{CSP}(\mathbf{A})$ is in P if $\mathbf{A}$ is bipartite, and NP-complete otherwise.

…or better classify *all* easy cases.

---

*Ie, avoids NPI, nonobvious if P≠NP.

CSP | Dichotomy

Hard problem. Find some useful easy cases…

Theorem (Hell and Nešetřil)
Let $\mathcal{C}$ be the class of finite connected simple graphs and $\mathbf{A} \in \mathcal{C}$.
Then $\mathrm{CSP}(\mathbf{A})$ is in P if $\mathbf{A}$ is bipartite, and NP-complete otherwise.

…or better classify *all* easy cases.

Conjecture (Feder and Vardi Dichotomy Conjecture, early 90s)
Let $\mathcal{C}$ be the class of finite σ-structures and $\mathbf{A} \in \mathcal{C}$.
*Then $\mathrm{CSP}(\mathbf{A})$ is in P or NP-complete.**

*Ie, avoids NPI, nonobvious if P≠NP.

CSP | Dichotomy

Hard problem. Find some useful easy cases...

Theorem (Hell and Nešetřil)
Let $\mathcal{C}$ be the class of finite connected simple graphs and $\mathbf{A} \in \mathcal{C}$.
Then $\mathrm{CSP}(\mathbf{A})$ is in P if $\mathbf{A}$ is bipartite, and NP-complete otherwise.

...or better classify *all* easy cases.

Conjecture (Feder and Vardi Dichotomy Conjecture, early 90s)
Let $\mathcal{C}$ be the class of finite σ-structures and $\mathbf{A} \in \mathcal{C}$.
*Then $\mathrm{CSP}(\mathbf{A})$ is in P or NP-complete.**

No reasonable line of attack until Jeavons (late 90s) discovers *polymorphisms*, a pertinent classification criterion for finite structures.

*Ie, avoids NPI, nonobvious if P$\neq$NP.

CSP | Polymorphisms

If **A** "admits" only "trivial polymorphisms",
then CSP(**A**) is NP-complete.

Example (Trivial Polymorphisms ⇒ Hardness)

$\mathbf{C}_3$ admits only trivial polymorphisms, ie, *projection* polymorphisms,
$f \in \mathrm{Pol}(I^{\mathbf{C}_3})$ iff $f(x_1, \ldots, x_i, \ldots, x_n) = x_i$ for some $i \in [n]$.

CSP | Polymorphisms

If **A** “admits” only “trivial polymorphisms”,
then CSP(**A**) is NP-complete.

Example (Trivial Polymorphisms ⇒ Hardness)

$\mathbf{C}_3$ admits only trivial polymorphisms, ie, *projection* polymorphisms,
$f \in \mathrm{Pol}(I^{\mathbf{C}_3})$ iff $f(x_1, \ldots, x_i, \ldots, x_n) = x_i$ for some $i \in [n]$.

If **A** “admits” some “nontrivial polymorphism”,
then CSP(**A**) is polytime tractable.

Example (Nontrivial Polymorphisms ⇒ Tractability)

$\mathbf{C}_2$ admits a nontrivial *majority* polymorphism,
$t(x, x, y) = t(x, y, x) = t(y, x, x) = x$.

CSP | Polymorphisms

If **A** "admits" only "trivial polymorphisms",
then CSP(**A**) is NP-complete.
(The solution space of $\mathbf{A} \models \phi$ lacks any algorithmically useful feature.)

Example (Trivial Polymorphisms ⇒ Hardness)

$\mathbf{C}_3$ admits only trivial polymorphisms, ie, *projection* polymorphisms,
$f \in \mathrm{Pol}(I^{\mathbf{C}_3})$ iff $f(x_1, \ldots, x_i, \ldots, x_n) = x_i$ for some $i \in [n]$.

If **A** "admits" some "nontrivial polymorphism",
then CSP(**A**) is polytime tractable.

Example (Nontrivial Polymorphisms ⇒ Tractability)

$\mathbf{C}_2$ admits a nontrivial *majority* polymorphism,
$t(x, x, y) = t(x, y, x) = t(y, x, x) = x$.

CSP | Polymorphisms

If **A** "admits" only "trivial polymorphisms",
then CSP(**A**) is NP-complete.
(The solution space of $\mathbf{A} \models \phi$ lacks any algorithmically useful feature.)

Example (Trivial Polymorphisms ⇒ Hardness)

$\mathbf{C}_3$ admits only trivial polymorphisms, ie, *projection* polymorphisms,
$f \in \mathrm{Pol}(I^{\mathbf{C}_3})$ iff $f(x_1, \ldots, x_i, \ldots, x_n) = x_i$ for some $i \in [n]$.

If **A** "admits" some "nontrivial polymorphism",
then CSP(**A**) is polytime tractable.
(The solution space of $\mathbf{A} \models \phi$ displays some algorithmically useful feature.)

Example (Nontrivial Polymorphisms ⇒ Tractability)

$\mathbf{C}_2$ admits a nontrivial *majority* polymorphism,
$t(x, x, y) = t(x, y, x) = t(y, x, x) = x$.

CSP | Polymorphisms

$R \subseteq A^k$ k-ary relation on A. $f : A^n \to A$ n-ary operation on A.

CSP | Polymorphisms

$R \subseteq A^k$ k-ary relation on A. $f : A^n \to A$ n-ary operation on A.
f is a *polymorphism* of R, in symbols $f \in \mathrm{Pol}(R)$, if:

CSP | Polymorphisms

$R \subseteq A^k$ k-ary relation on A. $f : A^n \to A$ n-ary operation on A.
f is a *polymorphism* of R, in symbols $f \in \mathrm{Pol}(R)$, if:

$$
\begin{array}{ccccc}
a_{11} & a_{12} & \dots & a_{1k} & \in R \\
a_{21} & a_{22} & \dots & a_{2k} & \in R \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
a_{n1} & a_{n2} & \dots & a_{nk} & \in R
\end{array}
$$

CSP | Polymorphisms

$R \subseteq A^k$ k-ary relation on A. $f \colon A^n \to A$ n-ary operation on A.
f is a *polymorphism* of R, in symbols $f \in \mathrm{Pol}(R)$, if:

$$
\begin{array}{ccccc}
\overset{f}{\frown} & \overset{f}{\frown} & \cdots & \overset{f}{\frown} & \\
a_{11} & a_{12} & \dots & a_{1k} & \in R \\
a_{21} & a_{22} & \dots & a_{2k} & \in R \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
a_{n1} & a_{n2} & \dots & a_{nk} & \in R \\
\| & \| & \dots & \| & \\
b_1 & b_2 & \dots & b_k &
\end{array}
$$

CSP | Polymorphisms

$R \subseteq A^k$ k-ary relation on A. $f\colon A^n \to A$ n-ary operation on A.
f is a *polymorphism* of R, in symbols $f \in \mathrm{Pol}(R)$, if:

$$
\begin{array}{ccccc}
\stackrel{f}{\frown} & \stackrel{f}{\frown} & \cdots & \stackrel{f}{\frown} & \\
a_{11} & a_{12} & \ldots & a_{1k} & \in R \\
a_{21} & a_{22} & \ldots & a_{2k} & \in R \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
a_{n1} & a_{n2} & \ldots & a_{nk} & \in R \\
\parallel & \parallel & \ldots & \parallel & \Downarrow \\
b_1 & b_2 & \ldots & b_k & \in R
\end{array}
$$

CSP | Polymorphisms

$R \subseteq A^k$ k-ary relation on A. $f\colon A^n \to A$ n-ary operation on A.
f is a *polymorphism* of R, in symbols $f \in \mathrm{Pol}(R)$, if:

$$
\begin{array}{ccccc}
f & f & \cdots & f & \\
a_{11} & a_{12} & \dots & a_{1k} & \in R \\
a_{21} & a_{22} & \dots & a_{2k} & \in R \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
a_{n1} & a_{n2} & \dots & a_{nk} & \in R \\
\| & \| & \dots & \| & \Downarrow \\
b_1 & b_2 & \dots & b_k & \in R
\end{array}
$$

$\mathbf{A} = (A, R_1^{\mathbf{A}}, \dots, R_m^{\mathbf{A}})$ relational structure. $\mathrm{Pol}(\mathbf{A}) = \bigcap_{i \in [m]} \mathrm{Pol}(R_i^{\mathbf{A}})$.

CSP | Polymorphisms

$R \subseteq A^k$ k-ary relation on A. $f : A^n \to A$ n-ary operation on A.
f is a *polymorphism* of R, in symbols $f \in \mathrm{Pol}(R)$, if:

$$
\begin{array}{ccccc}
\curvearrowright f & \curvearrowright f & \cdots & \curvearrowright f & \\
a_{11} & a_{12} & \dots & a_{1k} & \in R \\
a_{21} & a_{22} & \dots & a_{2k} & \in R \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
a_{n1} & a_{n2} & \dots & a_{nk} & \in R \\
\| & \| & \dots & \| & \Downarrow \\
b_1 & b_2 & \dots & b_k & \in R
\end{array}
$$

$\mathbf{A} = (A, R_1^{\mathbf{A}}, \dots, R_m^{\mathbf{A}})$ relational structure. $\mathrm{Pol}(\mathbf{A}) = \bigcap_{i \in [m]} \mathrm{Pol}(R_i^{\mathbf{A}})$.

Theorem (Jeavons)

The complexity of $\mathrm{CSP}(\mathbf{A})$ *is characterized by the algebra* $\mathbb{A} = (A, \mathrm{Pol}(\mathbf{A}))$*:*

1. $\mathrm{Pol}(\mathbf{A}_1) \subseteq \mathrm{Pol}(\mathbf{A}_2)$ *implies* $\mathrm{CSP}(\mathbf{A}_2) \leq_m^p \mathrm{CSP}(\mathbf{A}_1)$;
2. $\mathrm{Pol}(\mathbf{A}_1) = \mathrm{Pol}(\mathbf{A}_2)$ *implies* $\mathrm{CSP}(\mathbf{A}_1) \equiv_m^p \mathrm{CSP}(\mathbf{A}_2)$.

CSP | Polymorphisms | Classification

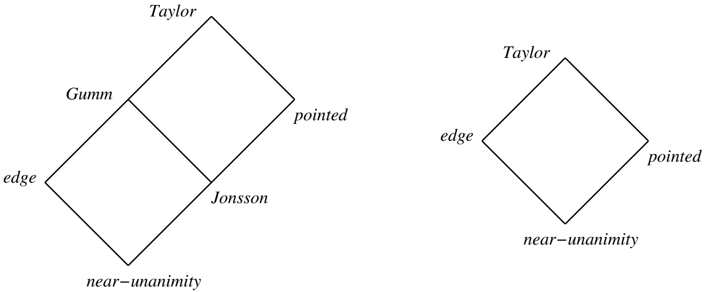
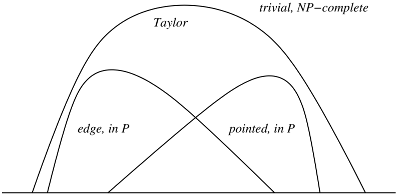


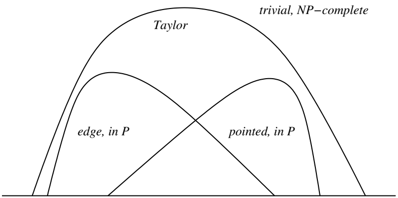
Figure: Nontrivial polimorphisms classified in 6 blocks, orderer by increasing triviality (left). Modulo Valeriote conjecture (right).

Taylor polymorphisms are maximally trivial among nontrivial polymorphisms.

CSP | Polymorphisms | Complexity



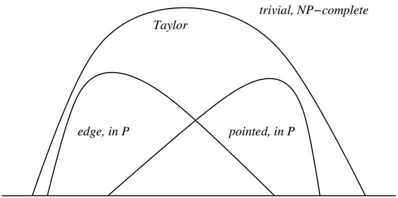
CSP | Polymorphisms | Complexity



Theorem

1. $\mathrm{Pol}(\mathbf{A})$ *trivial implies* $\mathrm{CSP}(\mathbf{A})$ *NP-complete.*

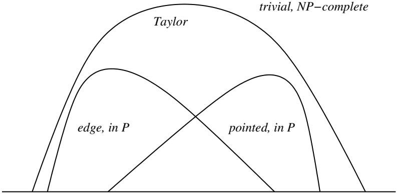
CSP | Polymorphisms | Complexity



Theorem

1. $\mathrm{Pol}(\mathbf{A})$ *trivial implies* $\mathrm{CSP}(\mathbf{A})$ *NP-complete.*
2. $\mathrm{Pol}(\mathbf{A})$ *has edge operations iff*
   $\mathrm{CSP}(\mathbf{A})$ *in P via Dalmau algorithm (Berman et al.).*

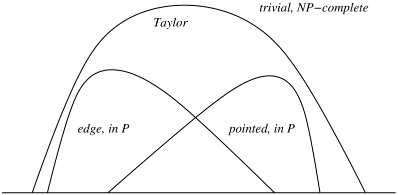
CSP | Polymorphisms | Complexity



Theorem

1. Pol($\mathbf{A}$) *trivial implies* CSP($\mathbf{A}$) *NP-complete.*
2. Pol($\mathbf{A}$) *has edge operations iff*
   CSP($\mathbf{A}$) *in P via Dalmau algorithm (Berman et al.).*
3. Pol($\mathbf{A}$) *has pointed operations iff*
   CSP($\mathbf{A}$) *in P via local consistency (Barto and Kozik, Bulatov).*

CSP | Polymorphisms | Complexity



Theorem

1. Pol($\mathbf{A}$) *trivial implies* CSP($\mathbf{A}$) *NP-complete.*
2. Pol($\mathbf{A}$) *has edge operations iff*
   CSP($\mathbf{A}$) *in P via Dalmau algorithm (Berman et al.).*
3. Pol($\mathbf{A}$) *has pointed operations iff*
   CSP($\mathbf{A}$) *in P via local consistency (Barto and Kozik, Bulatov).*

CSP($\mathbf{A}$) is conjectured in P in the uncovered case (BJK conjecture).

Tractability | Algorithms

Known tractable cases of the CSP($\mathbf{A}$) rely on two algorithms:

1. *local consistency*,
   works iff $\mathbf{A}$ has pointed polymorphisms;
2. *Dalmau algorithm* (a generalized *Gaussian elimination*),
   works iff $\mathbf{A}$ has edge polymorphisms.

Tractability | Algorithms

Known tractable cases of the CSP($\mathbf{A}$) rely on two algorithms:

1. *local consistency*,
   works iff $\mathbf{A}$ has pointed polymorphisms;
2. *Dalmau algorithm* (a generalized *Gaussian elimination*),
   works iff $\mathbf{A}$ has edge polymorphisms.

Algebraic properties of $\mathbf{A}$ yield computational *tractability* of CSP($\mathbf{A}$).

Tractability | Algorithms

Known tractable cases of the CSP($\mathbf{A}$) rely on two algorithms:

1. *local consistency*,
   works iff $\mathbf{A}$ has pointed polymorphisms;
2. *Dalmau algorithm* (a generalized *Gaussian elimination*),
   works iff $\mathbf{A}$ has edge polymorphisms.

Algebraic properties of $\mathbf{A}$ yield computational *tractability* of CSP($\mathbf{A}$). How?

Tractability | Algorithms

Known tractable cases of the CSP($\mathbf{A}$) rely on two algorithms:

1. *local consistency*,
   works iff $\mathbf{A}$ has pointed polymorphisms;
2. *Dalmau algorithm* (a generalized *Gaussian elimination*),
   works iff $\mathbf{A}$ has edge polymorphisms.

Algebraic properties of $\mathbf{A}$ yield computational *tractability* of CSP($\mathbf{A}$). How?

1. Pointed polymorphisms bypass the *incompleteness* of local search.

Tractability | *Algorithms*

Known tractable cases of the CSP($\mathbf{A}$) rely on two algorithms:

1. *local consistency*,
   works iff $\mathbf{A}$ has pointed polymorphisms;
2. *Dalmau algorithm* (a generalized *Gaussian elimination*),
   works iff $\mathbf{A}$ has edge polymorphisms.

Algebraic properties of $\mathbf{A}$ yield computational *tractability* of CSP($\mathbf{A}$). How?

1. Pointed polymorphisms bypass the *incompleteness* of local search.

2. Edge polymorphisms bypass the *complexity* of exhaustive search.

Tractability | Algorithms

Known tractable cases of the CSP($\mathbf{A}$) rely on two algorithms:

1. *local consistency*,
   works iff $\mathbf{A}$ has pointed polymorphisms;
2. *Dalmau algorithm* (a generalized *Gaussian elimination*),
   works iff $\mathbf{A}$ has edge polymorphisms.

Algebraic properties of $\mathbf{A}$ yield computational *tractability* of CSP($\mathbf{A}$). How?

1. Pointed polymorphisms bypass the *incompleteness* of local search.
   If $\mathbf{A}$ admits pointed polymorphisms,
   then the local consistency algorithm is *complete* on CSP($\mathbf{A}$).
2. Edge polymorphisms bypass the *complexity* of exhaustive search.

Tractability | Algorithms

Known tractable cases of the CSP(**A**) rely on two algorithms:

1. *local consistency*,
   works iff **A** has pointed polymorphisms;
2. *Dalmau algorithm* (a generalized *Gaussian elimination*),
   works iff **A** has edge polymorphisms.

Algebraic properties of **A** yield computational *tractability* of CSP(**A**). How?

1. Pointed polymorphisms bypass the *incompleteness* of local search.
   If **A** admits pointed polymorphisms, eg a *semilattice* polymorphism, then the local consistency algorithm is *complete* on CSP(**A**).
2. Edge polymorphisms bypass the *complexity* of exhaustive search.

Tractability | *Algorithms*

Known tractable cases of the CSP($\mathbf{A}$) rely on two algorithms:

1. *local consistency*,
   works iff $\mathbf{A}$ has pointed polymorphisms;
2. *Dalmau algorithm* (a generalized *Gaussian elimination*),
   works iff $\mathbf{A}$ has edge polymorphisms.

Algebraic properties of $\mathbf{A}$ yield computational *tractability* of CSP($\mathbf{A}$). How?

1. Pointed polymorphisms bypass the *incompleteness* of local search.
   If $\mathbf{A}$ admits pointed polymorphisms, eg a *semilattice* polymorphism,
   then the local consistency algorithm is *complete* on CSP($\mathbf{A}$).
2. Edge polymorphisms bypass the *complexity* of exhaustive search.
   If $\mathbf{A}$ admits edge polymorphisms,
   then the solution space of CSP($\mathbf{A}$) admits a *compact representation*.

Tractability | *Algorithms*

Known tractable cases of the CSP($\mathbf{A}$) rely on two algorithms:

1. *local consistency*,
   works iff $\mathbf{A}$ has pointed polymorphisms;
2. *Dalmau algorithm* (a generalized *Gaussian elimination*),
   works iff $\mathbf{A}$ has edge polymorphisms.

Algebraic properties of $\mathbf{A}$ yield computational *tractability* of CSP($\mathbf{A}$). How?

1. Pointed polymorphisms bypass the *incompleteness* of local search.
   If $\mathbf{A}$ admits pointed polymorphisms, eg a *semilattice* polymorphism, then the local consistency algorithm is *complete* on CSP($\mathbf{A}$).
2. Edge polymorphisms bypass the *complexity* of exhaustive search.
   If $\mathbf{A}$ admits edge polymorphisms, eg a *Mal'tsev* polymorphism, then the solution space of CSP($\mathbf{A}$) admits a *compact representation*.

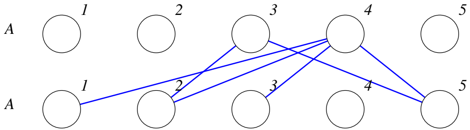
Tractability | Semilattice | Local Consistency

$t\colon A^2 \longrightarrow A$ is a *semilattice* operation if
$t(x,x) = x$, $t(x,y) = t(y,x)$, and $t(x,t(y,z)) = t(t(x,y),z)$.

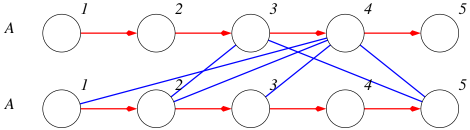
Theorem
If $\mathbf{A}$ *admits a semilattice polymorphism, then* $\mathbf{A}$ *has width* 1
(ie, the 1*-consistency algorithm decides* $\mathrm{CSP}(\mathbf{A})$ *in polytime).*

Tractability | *Semilattice* | *Local Consistency*

$\mathbf{A} = (A, E^{\mathbf{A}})$ where $A = \{1, 2, 3, 4, 5\}$ and
$E^{\mathbf{A}} = \{(3, a), (a, 3), (4, b), (b, 4) \mid a = 2, 5, b = 1, 2, 3, 5\} \subseteq A^2$:



$E^{\mathbf{A}}$ admits the semilattice polymorphism $t(x, y) = \max\{x, y\}$
($x \leq y$ iff $x \rightarrow y$):

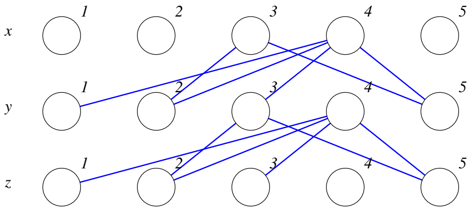


The 1-consistency algorithm decides $\mathrm{CSP}(\mathbf{A})$.

Tractability | Semilattice | Local Consistency

Example

Run 1-consistency on instance $\phi = \exists x \exists y \exists z (Exy \wedge Eyz)$ to CSP($\mathbf{A}$):

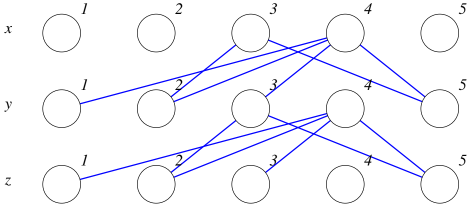


Tractability | Semilattice | Local Consistency

Example (Cont'd)

Initializing 1-consistency on instance $\phi = \exists x \exists y \exists z (Exy \wedge Eyz)$ to CSP($\mathbf{A}$) . . .

$\{x\}$

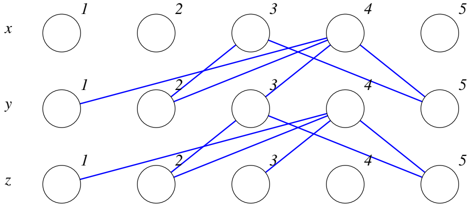


Tractability | Semilattice | Local Consistency

Example (Cont'd)

Initializing 1-consistency on instance $\phi = \exists x \exists y \exists z(Exy \wedge Eyz)$ to CSP$(\mathbf{A})\ldots$

$\{x\} \rightsquigarrow \phi$ has no constraints acting on x individually

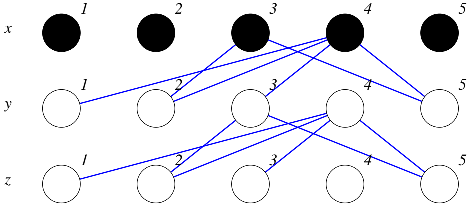


Tractability | Semilattice | Local Consistency

Example (Cont'd)

Initializing 1-consistency on instance $\phi = \exists x \exists y \exists z(Exy \wedge Eyz)$ to $\mathrm{CSP}(\mathbf{A}) \ldots$

$\{x\} \rightsquigarrow \phi$ has no constraints acting on x individually $\rightsquigarrow x \mapsto \{1, 2, 3, 4, 5\}$:

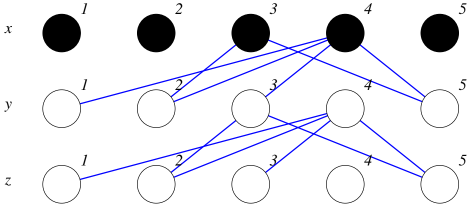


Tractability | Semilattice | Local Consistency

Example (Cont'd)

Initializing 1-consistency on instance $\phi = \exists x \exists y \exists z (Exy \wedge Eyz)$ to CSP$(\mathbf{A})$ . . .

$\{y\}$

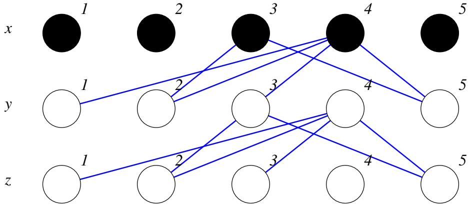


Tractability | Semilattice | Local Consistency

Example (Cont'd)

Initializing 1-consistency on instance $\phi = \exists x \exists y \exists z(Exy \wedge Eyz)$ to CSP($\mathbf{A}$) . . .

$\{y\} \rightsquigarrow \phi$ has no constraints acting on y individually

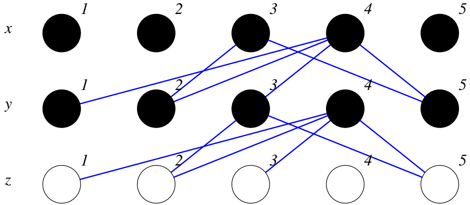


Tractability | Semilattice | Local Consistency

Example (Cont'd)

Initializing 1-consistency on instance $\phi = \exists x \exists y \exists z (Exy \wedge Eyz)$ to $\text{CSP}(\mathbf{A})\ldots$

$\{y\} \rightsquigarrow \phi$ has no constraints acting on y individually $\rightsquigarrow y \mapsto \{1,2,3,4,5\}$:

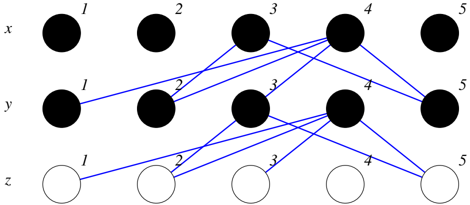


Tractability | Semilattice | Local Consistency

Example (Cont'd)

Initializing 1-consistency on instance $\phi = \exists x \exists y \exists z (Exy \wedge Eyz)$ to CSP($\mathbf{A}$) . . .

$\{z\}$

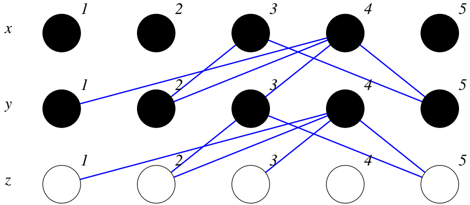


Tractability | Semilattice | Local Consistency

Example (Cont'd)

Initializing 1-consistency on instance $\phi = \exists x \exists y \exists z (Exy \wedge Eyz)$ to CSP($\mathbf{A}$) . . .

$\{z\} \rightsquigarrow \phi$ has no constraints acting on z individually

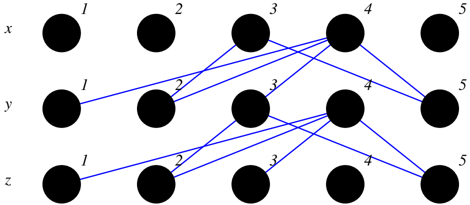


Tractability | Semilattice | Local Consistency

Example (Cont'd)

Initializing 1-consistency on instance $\phi = \exists x \exists y \exists z (Exy \wedge Eyz)$ to $\mathrm{CSP}(\mathbf{A}) \ldots$

$\{z\} \rightsquigarrow \phi$ has no constraints acting on z individually $\rightsquigarrow z \mapsto \{1, 2, 3, 4, 5\}$:

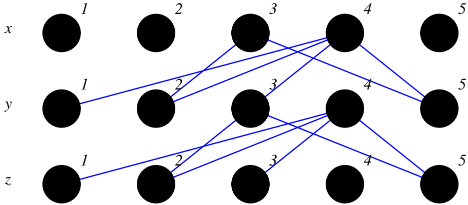


Tractability | Semilattice | Local Consistency

Example (Cont'd)

Iterating 1-consistency on instance $\phi = \exists x \exists y \exists z (Exy \wedge Eyz)$ to $\mathrm{CSP}(\mathbf{A}) \ldots$

$\{x, y\}$

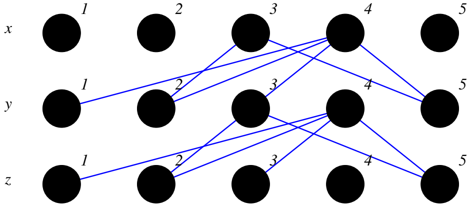


Tractability | Semilattice | Local Consistency

Example (Cont'd)

Iterating 1-consistency on instance $\phi = \exists x \exists y \exists z (Exy \wedge Eyz)$ to $\mathrm{CSP}(\mathbf{A}) \ldots$

$\{x, y\} \rightsquigarrow \phi$ has constraint Exy

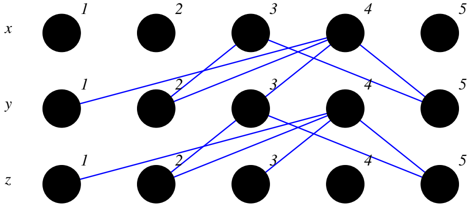


Tractability | Semilattice | Local Consistency

Example (Cont'd)

Iterating 1-consistency on instance $\phi = \exists x \exists y \exists z(Exy \wedge Eyz)$ to CSP($\mathbf{A}$) . . .

$\{x, y\} \rightsquigarrow \phi$ has constraint $Exy \rightsquigarrow (x, y) \mapsto E^{\mathbf{A}}$

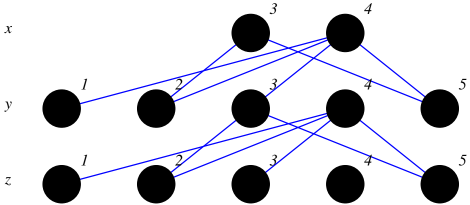


Tractability | Semilattice | Local Consistency

Example (Cont'd)

Iterating 1-consistency on instance $\phi = \exists x \exists y \exists z (Exy \wedge Eyz)$ to $\mathrm{CSP}(\mathbf{A})\ldots$

$\{x, y\} \rightsquigarrow \phi$ has constraint $Exy \rightsquigarrow (x, y) \mapsto E^{\mathbf{A}} \rightsquigarrow x \not\mapsto \{1, 2, 5\}$:

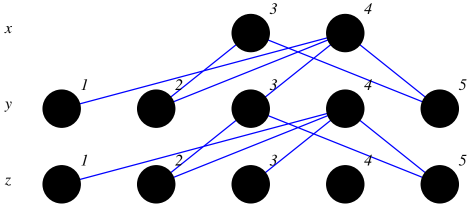


Tractability | Semilattice | Local Consistency

Example (Cont'd)

Iterating 1-consistency on instance $\phi = \exists x \exists y \exists z (Exy \wedge Eyz)$ to $\mathrm{CSP}(\mathbf{A})\ldots$

$\{x, z\}$

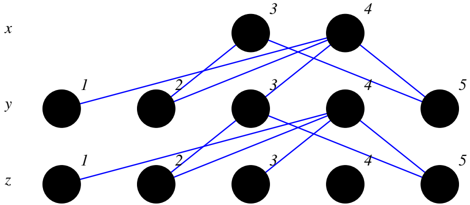


Tractability | Semilattice | Local Consistency

Example (Cont'd)

Iterating 1-consistency on instance $\phi = \exists x \exists y \exists z (Exy \wedge Eyz)$ to CSP($\mathbf{A}$) . . .

$\{x, z\} \rightsquigarrow \phi$ has no constraints on x and z

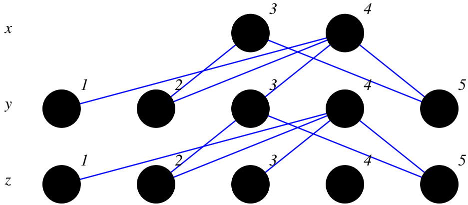


Tractability | Semilattice | Local Consistency

Example (Cont'd)

Iterating 1-consistency on instance $\phi = \exists x \exists y \exists z (Exy \wedge Eyz)$ to CSP($\mathbf{A}$) . . .

$\{x, z\} \rightsquigarrow \phi$ has no constraints on x and $z \rightsquigarrow$ no changes:

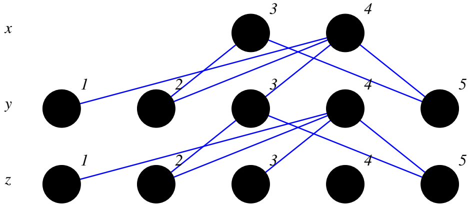


Tractability | Semilattice | Local Consistency

Example (Cont'd)

Iterating 1-consistency on instance $\phi = \exists x \exists y \exists z (Exy \wedge Eyz)$ to CSP($\mathbf{A}$) . . .

$\{y, x\}$

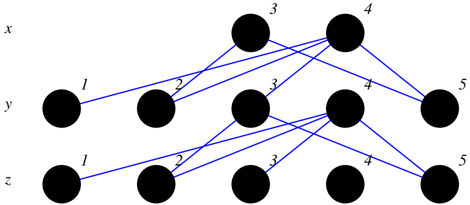


Tractability | Semilattice | Local Consistency

Example (Cont'd)

Iterating 1-consistency on instance $\phi = \exists x \exists y \exists z (Exy \wedge Eyz)$ to $\mathrm{CSP}(\mathbf{A})\ldots$

$\{y, x\} \rightsquigarrow \phi$ has constraint Exy

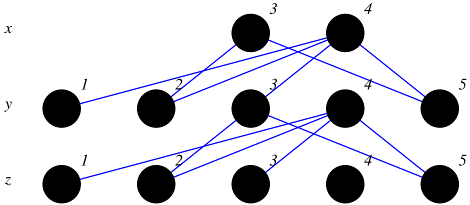


Tractability | Semilattice | Local Consistency

Example (Cont'd)

Iterating 1-consistency on instance $\phi = \exists x \exists y \exists z(Exy \wedge Eyz)$ to CSP($\mathbf{A}$) . . .

$\{y, x\} \rightsquigarrow \phi$ has constraint $Exy \rightsquigarrow (x, y) \mapsto E^{\mathbf{A}}$

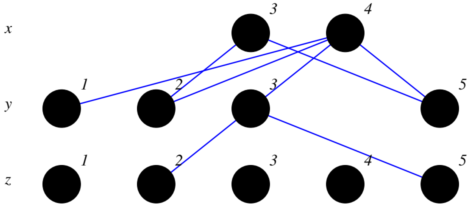


Tractability | Semilattice | Local Consistency

Example (Cont'd)

Iterating 1-consistency on instance $\phi = \exists x \exists y \exists z (Exy \wedge Eyz)$ to $\mathrm{CSP}(\mathbf{A}) \ldots$

$\{y, x\} \rightsquigarrow \phi$ has constraint $Exy \rightsquigarrow (x, y) \mapsto E^{\mathbf{A}} \rightsquigarrow y \not\mapsto \{4\}$:

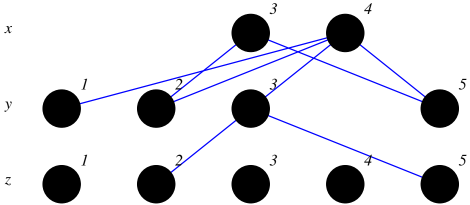


Tractability | Semilattice | Local Consistency

Example (Cont'd)

Iterating 1-consistency on instance $\phi = \exists x \exists y \exists z (Exy \wedge Eyz)$ to $\mathrm{CSP}(\mathbf{A}) \ldots$

$\{y, z\}$

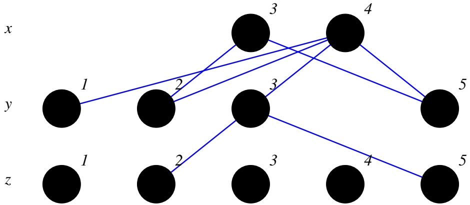


Tractability | Semilattice | Local Consistency

Example (Cont'd)

Iterating 1-consistency on instance $\phi = \exists x \exists y \exists z(Exy \wedge Eyz)$ to $\mathrm{CSP}(\mathbf{A}) \ldots$

$\{y, z\} \rightsquigarrow \phi$ has constraint Eyz

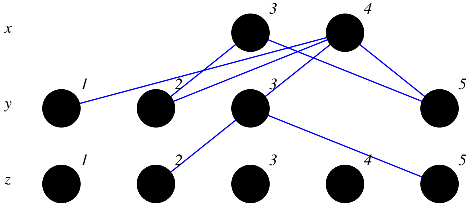


Tractability | Semilattice | Local Consistency

Example (Cont'd)

Iterating 1-consistency on instance $\phi = \exists x \exists y \exists z (Exy \wedge Eyz)$ to $\mathrm{CSP}(\mathbf{A}) \ldots$

$\{y, z\} \rightsquigarrow \phi$ has constraint $Eyz \rightsquigarrow (y, z) \mapsto E^{\mathbf{A}}$

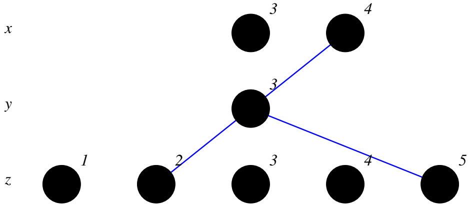


Tractability | Semilattice | Local Consistency

Example (Cont'd)

Iterating 1-consistency on instance $\phi = \exists x \exists y \exists z(Exy \wedge Eyz)$ to $\mathrm{CSP}(\mathbf{A}) \ldots$

$\{y, z\} \rightsquigarrow \phi$ has constraint $Eyz \rightsquigarrow (y, z) \mapsto E^{\mathbf{A}} \rightsquigarrow y \not\mapsto \{1, 2, 5\}$:

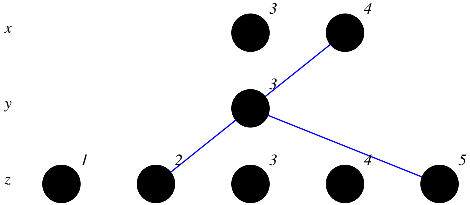


Tractability | Semilattice | Local Consistency

Example (Cont'd)

Iterating 1-consistency on instance $\phi = \exists x \exists y \exists z(Exy \wedge Eyz)$ to CSP($\mathbf{A}$) . . .

$\{z, x\}, \{z, y\}, \{x, y\}$

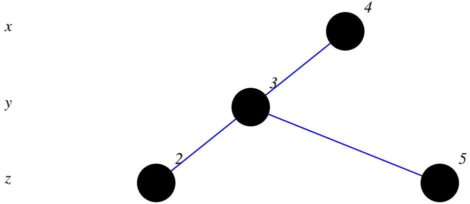


Tractability | Semilattice | Local Consistency

Example (Cont'd)

Iterating 1-consistency on instance $\phi = \exists x \exists y \exists z (Exy \wedge Eyz)$ to CSP($\mathbf{A}$) ...

$\{z, x\}, \{z, y\}, \{x, y\} \rightsquigarrow$ stabilizes at $A_x = \{4\}$, $A_y = \{3\}$, $A_z = \{2, 5\}$:

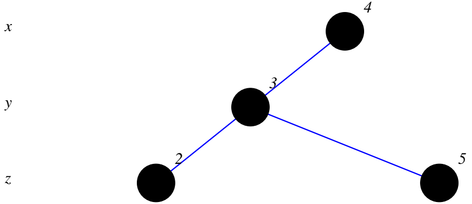


Tractability | Semilattice | Local Consistency

Example (Cont'd)

Iterating 1-consistency on instance $\phi = \exists x \exists y \exists z (Exy \wedge Eyz)$ to CSP($\mathbf{A}$) ...

$\{z, x\}, \{z, y\}, \{x, y\} \rightsquigarrow$ stabilizes at $A_x = \{4\}$, $A_y = \{3\}$, $A_z = \{2, 5\}$:



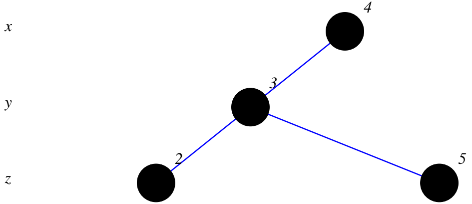
$A_x, A_y, A_z \neq \emptyset$

Tractability | Semilattice | Local Consistency

Example (Cont'd)

Iterating 1-consistency on instance $\phi = \exists x \exists y \exists z (Exy \wedge Eyz)$ to CSP($\mathbf{A}$) . . .

$\{z, x\}, \{z, y\}, \{x, y\} \rightsquigarrow$ stabilizes at $A_x = \{4\}, A_y = \{3\}, A_z = \{2, 5\}$:



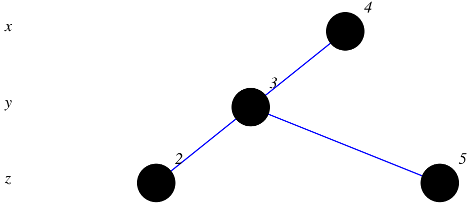
$A_x, A_y, A_z \neq \emptyset \rightsquigarrow$ return "Yes!"

Tractability | Semilattice | Local Consistency

Example (Cont'd)

Iterating 1-consistency on instance $\phi = \exists x \exists y \exists z (Exy \wedge Eyz)$ to CSP($\mathbf{A}$) . . .

$\{z, x\}, \{z, y\}, \{x, y\} \rightsquigarrow$ stabilizes at $A_x = \{4\}$, $A_y = \{3\}$, $A_z = \{2, 5\}$:



$A_x, A_y, A_z \neq \emptyset \rightsquigarrow$ return "Yes!" Correct?

Tractability | Semilattice | Local Consistency

Example (Cont'd)

$A_x = \{4\}, A_y = \{3\}, A_z = \{2, 5\}.$

Tractability | Semilattice | Local Consistency

Example (Cont'd)

$A_x = \{4\}, A_y = \{3\}, A_z = \{2, 5\}$.

$w \mapsto \bigvee A_w$ for $w \in \{x, y, z\}$ witnesses $\mathbf{A} \models \exists x \exists y \exists z (Exy \wedge Eyz)$.

Tractability | Semilattice | Local Consistency

Example (Cont'd)

$A_x = \{4\}, A_y = \{3\}, A_z = \{2, 5\}.$

$w \mapsto \bigvee A_w$ for $w \in \{x, y, z\}$ witnesses $\mathbf{A} \models \exists x \exists y \exists z (Exy \wedge Eyz)$.

$$\left(\mathbf{A}, \begin{array}{l} x \mapsto 4 \\ y \mapsto 3 \end{array}\right) \models Exy \text{ iff } (4, 3) \in E^{\mathbf{A}}.$$

Tractability | Semilattice | Local Consistency

Example (Cont'd)

$A_x = \{4\}, A_y = \{3\}, A_z = \{2, 5\}.$

$w \mapsto \bigvee A_w$ for $w \in \{x, y, z\}$ witnesses $\mathbf{A} \models \exists x \exists y \exists z (Exy \wedge Eyz)$.

$\left(\mathbf{A}, \begin{array}{l} x \mapsto 4 \\ y \mapsto 3 \end{array}\right) \models Exy$ iff $(4, 3) \in E^{\mathbf{A}}$.

Check:

Tractability | Semilattice | Local Consistency

Example (Cont'd)

$A_x = \{4\}, A_y = \{3\}, A_z = \{2, 5\}.$

$w \mapsto \bigvee A_w$ for $w \in \{x, y, z\}$ witnesses $\mathbf{A} \models \exists x \exists y \exists z (Exy \wedge Eyz)$.

$$\left(\mathbf{A}, \begin{array}{l} x \mapsto 4 \\ y \mapsto 3 \end{array} \right) \models Exy \text{ iff } (4, 3) \in E^{\mathbf{A}}.$$

Check:

1. $x \mapsto 4$ extends to some $y \mapsto a \in A_y$ st $(4, a) \in E^{\mathbf{A}}$, and
   $y \mapsto 3$ extends to some $x \mapsto b \in A_x$ st $(b, 3) \in E^{\mathbf{A}}$, by 1-consistency;

Tractability | Semilattice | Local Consistency

Example (Cont'd)

$A_x = \{4\}, A_y = \{3\}, A_z = \{2, 5\}.$

$w \mapsto \bigvee A_w$ for $w \in \{x, y, z\}$ witnesses $\mathbf{A} \models \exists x \exists y \exists z(Exy \wedge Eyz)$.

$\left(\mathbf{A}, \begin{array}{c} x \mapsto 4 \\ y \mapsto 3 \end{array}\right) \models Exy$ iff $(4, 3) \in E^{\mathbf{A}}$.

Check:

1. $x \mapsto 4$ extends to some $y \mapsto a \in A_y$ st $(4, a) \in E^{\mathbf{A}}$, and $y \mapsto 3$ extends to some $x \mapsto b \in A_x$ st $(b, 3) \in E^{\mathbf{A}}$, by 1-consistency;
2.

$$\begin{array}{ccc} 4 & a & \in E^{\mathbf{A}} \\ b & 3 & \in E^{\mathbf{A}} \end{array}$$

Tractability | Semilattice | Local Consistency

Example (Cont'd)

$A_x = \{4\}, A_y = \{3\}, A_z = \{2, 5\}$.

$w \mapsto \bigvee A_w$ for $w \in \{x, y, z\}$ witnesses $\mathbf{A} \models \exists x \exists y \exists z (Exy \wedge Eyz)$.

$\left(\mathbf{A}, \begin{array}{c} x \mapsto 4 \\ y \mapsto 3 \end{array} \right) \models Exy$ iff $(4, 3) \in E^{\mathbf{A}}$.

Check:

1. $x \mapsto 4$ extends to some $y \mapsto a \in A_y$ st $(4, a) \in E^{\mathbf{A}}$, and
   $y \mapsto 3$ extends to some $x \mapsto b \in A_x$ st $(b, 3) \in E^{\mathbf{A}}$, by 1-consistency;
2. $$\begin{array}{ccc} t & t & \\ 4 & a & \in E^{\mathbf{A}} \\ b & 3 & \in E^{\mathbf{A}} \\ \| & \| & \\ 4 & 3 & \end{array}$$

   as t semilattice implies $t(\bigvee A_w, c) = t(c, \bigvee A_w) = \bigvee A_w$ for all $c \in A_w$ and $w \in \{x, y\}$,

Tractability | *Semilattice* | *Local Consistency*

Example (Cont'd)

$A_x = \{4\}, A_y = \{3\}, A_z = \{2, 5\}$.

$w \mapsto \bigvee A_w$ for $w \in \{x, y, z\}$ witnesses $\mathbf{A} \models \exists x \exists y \exists z (Exy \wedge Eyz)$.

$\left(\mathbf{A}, \begin{array}{l} x \mapsto 4 \\ y \mapsto 3 \end{array}\right) \models Exy$ iff $(4, 3) \in E^{\mathbf{A}}$.

Check:

1. $x \mapsto 4$ extends to some $y \mapsto a \in A_y$ st $(4, a) \in E^{\mathbf{A}}$, and $y \mapsto 3$ extends to some $x \mapsto b \in A_x$ st $(b, 3) \in E^{\mathbf{A}}$, by 1-consistency;
2. $$\begin{array}{ccc} t & t & \\ 4 & a & \in E^{\mathbf{A}} \\ b & 3 & \in E^{\mathbf{A}} \\ \| & \| & \Downarrow \\ 4 & 3 & \in E^{\mathbf{A}} \end{array}$$

   as t semilattice implies $t(\bigvee A_w, c) = t(c, \bigvee A_w) = \bigvee A_w$ for all $c \in A_w$ and $w \in \{x, y\}$, and t is a polymorphism of $E^{\mathbf{A}}$.

Tractability | Semilattice | Local Consistency

Theorem

If $\mathbf{A}$ *admits a semilattice polymorphism* t*, then* $\mathbf{A}$ *has width* 1
(ie, the 1*-consistency algorithm decides* $\mathrm{CSP}(\mathbf{A})$ *in polytime).*

Proof (Idea).

ϕ *instance of* $\mathrm{CSP}(\mathbf{A})$ *over variables* $\{x_1, \ldots, x_n\}$*. Run* 1*-consistency.*
If $A_{x_i} = \emptyset$ *for some* $i \in \{1, \ldots, n\}$*, then* $\mathbf{A} \not\models \phi$*.*
Otherwise $\mathbf{A} \models \phi$*, witnessed by* $x_i \mapsto \bigvee A_{x_i} \in A$
(A *is a complete join semilattice under* $a \leq b$ *iff* $t(a, b) = b$*,*
so every $S \subseteq A$ *has a least upper bound in* A*).* $\square$

Tractability | Mal'tsev | Dalmau Algorithm

$t\colon A^3 \to A$ is a *Mal'tsev* operation if $t(x,y,y) = t(y,y,x) = x$.

Theorem

If $\mathbf{A}$ *admits a Mal'tsev polymorphism,*
then the solution space of $\mathrm{CSP}(\mathbf{A})$ *admits a compact representation*
(and Dalmau algorithm decides $\mathrm{CSP}(\mathbf{A})$ *in polytime).*

Tractability | Mal'tsev | Dalmau Algorithm

Example

$\mathbf{A} = (A, R^{\mathbf{A}})$ where $A = \{0, 1, 2, 3, 4\}$, $R^{\mathbf{A}} = \{(a, b, c) \mid \mathbf{Z}_5 \models a + b = c\} \subseteq A^3$.

$R^{\mathbf{A}}$ admits the Mal'tsev polymorphism $t(x, y, z) = x - y + z$ (over $\mathbf{Z}_5$):

$$\begin{array}{ccccccc} a_1 & + & a_2 & = & a_3 & \in R^{\mathbf{A}} \\ b_1 & + & b_2 & = & b_3 & \in R^{\mathbf{A}} \\ c_1 & + & c_2 & = & c_3 & \in R^{\mathbf{A}} \end{array}$$

Tractability | Mal'tsev | Dalmau Algorithm

Example

$\mathbf{A} = (A, R^{\mathbf{A}})$ where $A = \{0, 1, 2, 3, 4\}$, $R^{\mathbf{A}} = \{(a, b, c) \mid \mathbf{Z}_5 \models a + b = c\} \subseteq A^3$.

$R^{\mathbf{A}}$ admits the Mal'tsev polymorphism $t(x, y, z) = x - y + z$ (over $\mathbf{Z}_5$):

$$\begin{array}{ccccccc}
a_1 & + & a_2 & = & a_3 & & \in R^{\mathbf{A}} \\
b_1 & + & b_2 & = & b_3 & & \in R^{\mathbf{A}} \\
c_1 & + & c_2 & = & c_3 & & \in R^{\mathbf{A}} \\
\| & & \| & & \| & & \\
(a_1 - b_1 + c_1) & & (a_2 - b_2 + c_2) & & (a_3 - b_3 + c_3) & &
\end{array}$$

Tractability | Mal'tsev | Dalmau Algorithm

Example

$\mathbf{A} = (A, R^{\mathbf{A}})$ where $A = \{0, 1, 2, 3, 4\}$, $R^{\mathbf{A}} = \{(a, b, c) \mid \mathbf{Z}_5 \models a + b = c\} \subseteq A^3$.

$R^{\mathbf{A}}$ admits the Mal'tsev polymorphism $t(x, y, z) = x - y + z$ (over $\mathbf{Z}_5$):

$$\begin{array}{ccccccc}
\nrightarrow & & \nrightarrow & & \nrightarrow & & \\
a_1 & + & a_2 & = & a_3 & & \in R^{\mathbf{A}} \\
b_1 & + & b_2 & = & b_3 & & \in R^{\mathbf{A}} \\
c_1 & + & c_2 & = & c_3 & & \in R^{\mathbf{A}} \\
\| & & \| & & \| & & \Downarrow \\
(a_1 - b_1 + c_1) & + & (a_2 - b_2 + c_2) & = & (a_3 - b_3 + c_3) & & \in R^{\mathbf{A}}
\end{array}$$

Tractability | Mal'tsev | Dalmau Algorithm

Example

$\mathbf{A} = (A, R^{\mathbf{A}})$ where $A = \{0, 1, 2, 3, 4\}$, $R^{\mathbf{A}} = \{(a, b, c) \mid \mathbf{Z}_5 \models a + b = c\} \subseteq A^3$.

$R^{\mathbf{A}}$ admits the Mal'tsev polymorphism $t(x, y, z) = x - y + z$ (over $\mathbf{Z}_5$):

$$\begin{array}{ccccccc}
t & & t & & t & & \\
a_1 & + & a_2 & = & a_3 & & \in R^{\mathbf{A}} \\
b_1 & + & b_2 & = & b_3 & & \in R^{\mathbf{A}} \\
c_1 & + & c_2 & = & c_3 & & \in R^{\mathbf{A}} \\
\Vert & & \Vert & & \Vert & & \Downarrow \\
(a_1 - b_1 + c_1) & + & (a_2 - b_2 + c_2) & = & (a_3 - b_3 + c_3) & & \in R^{\mathbf{A}}
\end{array}$$

Dalmau algorithm decides $\mathrm{CSP}(\mathbf{A})$ in polytime.

Tractability | Mal'tsev | Dalmau Algorithm

Example (Cont'd)

$\phi = \exists x \exists y \exists z (Rxzx \wedge Rxyy)$ instance of CSP$(\mathbf{A})$ with 2 constraints.

$$C_1 = \left[\begin{array}{ccccc|c} 0 & 1 & 2 & 3 & 4 & \\ a & a & a & a & a & a \in \{0,1,2,3,4\} \\ 0 & 0 & 0 & 0 & 0 & \end{array} \right] \text{ solutions to } Rxzx.$$

$$C_2 = \left[\begin{array}{ccccc|c} 0 & 0 & 0 & 0 & 0 & \\ 0 & 1 & 2 & 3 & 4 & a \in \{0,1,2,3,4\} \\ a & a & a & a & a & \end{array} \right] \text{ solutions to } Rxyy.$$

$$S_2 = C_1 \cap C_2 = \left[\begin{array}{c|c} 0 & \\ a & a \in \{0,1,2,3,4\} \\ 0 & \end{array} \right] \text{ solutions to } Rxzx \wedge Rxyy.$$

$S_2 \neq \emptyset$.

$\mathbf{A} \models \phi$.

Tractability | Mal'tsev | Dalmau Algorithm

ϕ instance of CSP($\mathbf{A}$) with m constraints and k variables,

$$\phi = \exists x_1 \cdots \exists x_k(\phi_1(x_1, \ldots, x_k) \wedge \cdots \wedge \phi_m(x_1, \ldots, x_k)).$$

$C_i \subseteq A^{\{x_1, \ldots, x_k\}}$ solutions ith constraint ($i \in \{1, \ldots, m\}$).
$S_i = C_1 \cap \cdots \cap C_i$ solutions first i constraints ($i \in \{0, 1, \ldots, m\}$).
S_m solutions of ϕ (ie, any $\mathbf{a} \in S_m$ witnesses $\mathbf{A} \models \phi$).

NAIVEALGORITHM(ϕ)

$S_0 = A^{\{x_1, \ldots, x_k\}}$
for $i = 1, \ldots, m$
  $S_i = S_{i-1} \cap C_i$
  if ($S_i = \emptyset$) **output** $\mathbf{A} \not\models \phi$
endfor
output $\mathbf{A} \models \phi$

Tractability | Mal'tsev | Dalmau Algorithm

ϕ instance of CSP($\mathbf{A}$) with m constraints and k variables,

$$\phi = \exists x_1 \cdots \exists x_k (\phi_1(x_1, \ldots, x_k) \wedge \cdots \wedge \phi_m(x_1, \ldots, x_k)).$$

$C_i \subseteq A^{\{x_1, \ldots, x_k\}}$ solutions ith constraint ($i \in \{1, \ldots, m\}$).
$S_i = C_1 \cap \cdots \cap C_i$ solutions first i constraints ($i \in \{0, 1, \ldots, m\}$).
S_m solutions of ϕ (ie, any $\mathbf{a} \in S_m$ witnesses $\mathbf{A} \models \phi$).

NaiveAlgorithm(ϕ)
$S_0 = A^{\{x_1, \ldots, x_k\}}$
for $i = 1, \ldots, m$
  $S_i = S_{i-1} \cap C_i$
  if ($S_i = \emptyset$) **output** $\mathbf{A} \not\models \phi$
endfor
output $\mathbf{A} \models \phi$

Correct,

Tractability | Mal'tsev | Dalmau Algorithm

ϕ instance of $\mathrm{CSP}(\mathbf{A})$ with m constraints and k variables,

$$\phi = \exists x_1 \cdots \exists x_k(\phi_1(x_1, \ldots, x_k) \wedge \cdots \wedge \phi_m(x_1, \ldots, x_k)).$$

$C_i \subseteq A^{\{x_1,\ldots,x_k\}}$ solutions ith constraint ($i \in \{1, \ldots, m\}$).
$S_i = C_1 \cap \cdots \cap C_i$ solutions first i constraints ($i \in \{0, 1, \ldots, m\}$).
S_m solutions of ϕ (ie, any $\mathbf{a} \in S_m$ witnesses $\mathbf{A} \models \phi$).

NAIVEALGORITHM(ϕ)
$S_0 = A^{\{x_1,\ldots,x_k\}}$ ▶ size $O(2^k)$
for $i = 1, \ldots, m$
  $S_i = S_{i-1} \cap C_i$ ▶ size $O(2^k)$
  if ($S_i = \emptyset$) **output** $\mathbf{A} \not\models \phi$
endfor
output $\mathbf{A} \models \phi$

Correct, but not polytime!

Tractability | Mal'tsev | Dalmau Algorithm

ϕ instance of $\mathrm{CSP}(\mathbf{A})$ with m constraints and k variables,

$$\phi = \exists x_1 \cdots \exists x_k (\phi_1(x_1, \ldots, x_k) \wedge \cdots \wedge \phi_m(x_1, \ldots, x_k)).$$

$C_i \subseteq A^{\{x_1, \ldots, x_k\}}$ solutions ith constraint.
$S'_i = C_1 \cap \cdots \cap C_i$ *compact representation* of solutions first i constraints.
S'_m *compact representation* of solutions to ϕ.

```
DALMAUALGORITHM(φ)
    S'_0 = (A^{x_1,...,x_k})'
    for i = 1, ..., m
        S'_i = S'_{i-1} ∩ C_i
        if (S'_i = ∅) output A ⊭ φ
    endfor
    output A ⊨ φ
```

Tractability | Mal'tsev | Dalmau Algorithm

ϕ instance of CSP($\mathbf{A}$) with m constraints and k variables,

$$\phi = \exists x_1 \cdots \exists x_k(\phi_1(x_1, \ldots, x_k) \wedge \cdots \wedge \phi_m(x_1, \ldots, x_k)).$$

$C_i \subseteq A^{\{x_1,\ldots,x_k\}}$ solutions ith constraint.
$S'_i = C_1 \cap \cdots \cap C_i$ *compact representation* of solutions first i constraints.
S'_m *compact representation* of solutions to ϕ.

DALMAUALGORITHM(ϕ)

$S'_0 = (A^{\{x_1,\ldots,x_k\}})'$
for $i = 1, \ldots, m$
    $S'_i = S'_{i-1} \cap C_i$
    if ($S'_i = \emptyset$) **output** $\mathbf{A} \not\models \phi$
endfor
output $\mathbf{A} \models \phi$

Correct,

Tractability | Mal'tsev | Dalmau Algorithm

ϕ instance of CSP($\mathbf{A}$) with m constraints and k variables,

$$\phi = \exists x_1 \cdots \exists x_k (\phi_1(x_1, \ldots, x_k) \wedge \cdots \wedge \phi_m(x_1, \ldots, x_k)).$$

$C_i \subseteq A^{\{x_1,\ldots,x_k\}}$ solutions ith constraint.
$S'_i = C_1 \cap \cdots \cap C_i$ *compact representation* of solutions first i constraints.
S'_m *compact representation* of solutions to ϕ.

DALMAUALGORITHM(ϕ)

$S'_0 = (A^{\{x_1,\ldots,x_k\}})'$ ▶ size $O(k)$
for $i = 1, \ldots, m$
  $S'_i = S'_{i-1} \cap C_i$ ▶ size $O(k)$
  if ($S'_i = \emptyset$) **output** $\mathbf{A} \not\models \phi$
endfor
output $\mathbf{A} \models \phi$

Correct, and (with clever implementation) polytime!

Tractability | Mal'tsev | Dalmau Algorithm

If a k-ary relation $R \subseteq A^k$ on a set A admits a Mal'tsev polymorphism t, then R admits a *compact representation* R', that is a $R' \subseteq R$ such that:

1. $|R'| \leq 2|A|k = O(k)$ versus $|R| \leq |A|^k = O(2^k)$;
2. R is equal to $t(R')$, the smallest relation containing R' closed under t ($\mathbf{a}, \mathbf{b}, \mathbf{c} \in t(R')$ implies $(t(\mathbf{a}_1, \mathbf{b}_1, \mathbf{c}_1), \ldots, t(\mathbf{a}_k, \mathbf{b}_k, \mathbf{c}_k)) \in t(R')$), ie, each $\mathbf{a} \in R$ is derivable from R' using the Mal'tsev polymorphism t of R.

R' and t provide a poly(k) size encoding for R!

R' constructed by: For all $(i, a, b) \in \{1, \ldots, k\} \times A \times A$,
there exist $\mathbf{a}, \mathbf{b} \in R$ such that $\mathbf{a}_j = \mathbf{b}_j$ for all $j < i$, $\mathbf{a}_i = a$, $\mathbf{b}_i = b$,
iff there exist $\mathbf{a}', \mathbf{b}' \in R'$ such that $\mathbf{a}'_j = \mathbf{b}'_j$ for all $j < i$, $\mathbf{a}'_i = a$, $\mathbf{b}'_i = b$.

Tractability | Mal'tsev | Dalmau Algorithm

Example ($k = 3$, $A = \{0, 1, 2, 3, 4\}$)

$R = \{0, 1, 2, 3, 4\}^3$ admits Mal'tsev polymorphism $t(x, y, z) = x - y + z$,

$$t(x, y, y) = t(y, y, x) = x.$$

Define $R' = \{(a, 0, 0), (0, a, 0), (0, 0, a) \mid a \in \{0, 1, 2, 3, 4\}\} \subseteq R$.

Tractability | Mal'tsev | Dalmau Algorithm

Example ($k = 3$, $A = \{0, 1, 2, 3, 4\}$)

$R = \{0, 1, 2, 3, 4\}^3$ admits Mal'tsev polymorphism $t(x, y, z) = x - y + z$,

$$t(x, y, y) = t(y, y, x) = x.$$

Define $R' = \{(a, 0, 0), (0, a, 0), (0, 0, a) \mid a \in \{0, 1, 2, 3, 4\}\} \subseteq R$. Check:

Tractability | Mal'tsev | Dalmau Algorithm

Example ($k = 3$, $A = \{0, 1, 2, 3, 4\}$)

$R = \{0, 1, 2, 3, 4\}^3$ admits Mal'tsev polymorphism $t(x, y, z) = x - y + z$,

$$t(x, y, y) = t(y, y, x) = x.$$

Define $R' = \{(a, 0, 0), (0, a, 0), (0, 0, a) \mid a \in \{0, 1, 2, 3, 4\}\} \subseteq R$. Check:

1. $|R'| \leq |A|k \leq 2|A|k \ll |A|^k = |R|$;

Tractability | Mal'tsev | Dalmau Algorithm

Example ($k = 3$, $A = \{0, 1, 2, 3, 4\}$)

$R = \{0, 1, 2, 3, 4\}^3$ admits Mal'tsev polymorphism $t(x, y, z) = x - y + z$,

$$t(x, y, y) = t(y, y, x) = x.$$

Define $R' = \{(a, 0, 0), (0, a, 0), (0, 0, a) \mid a \in \{0, 1, 2, 3, 4\}\} \subseteq R$. Check:

1. $|R'| \leq |A|k \leq 2|A|k \ll |A|^k = |R|$;
2. For instance, derive $(2, 3, 1) \in R$ from R' using t:

Tractability | Mal'tsev | Dalmau Algorithm

Example ($k = 3$, $A = \{0, 1, 2, 3, 4\}$)

$R = \{0, 1, 2, 3, 4\}^3$ admits Mal'tsev polymorphism $t(x, y, z) = x - y + z$,

$$t(x, y, y) = t(y, y, x) = x.$$

Define $R' = \{(a, 0, 0), (0, a, 0), (0, 0, a) \mid a \in \{0, 1, 2, 3, 4\}\} \subseteq R$. Check:

1. $|R'| \leq |A|k \leq 2|A|k \ll |A|^k = |R|$;
2. For instance, derive $(2, 3, 1) \in R$ from R' using t:

$$\begin{array}{cccl} 2 & 0 & 0 & \in R' \\ 0 & 0 & 0 & \in R' \\ 0 & 3 & 0 & \in R' \end{array}$$

Tractability | Mal'tsev | Dalmau Algorithm

Example ($k = 3$, $A = \{0, 1, 2, 3, 4\}$)

$R = \{0, 1, 2, 3, 4\}^3$ admits Mal'tsev polymorphism $t(x, y, z) = x - y + z$,

$$t(x, y, y) = t(y, y, x) = x.$$

Define $R' = \{(a, 0, 0), (0, a, 0), (0, 0, a) \mid a \in \{0, 1, 2, 3, 4\}\} \subseteq R$. Check:

1. $|R'| \leq |A|k \leq 2|A|k \ll |A|^k = |R|$;
2. For instance, derive $(2, 3, 1) \in R$ from R' using t:

$$\begin{array}{cccc}
t & t & t & \\
2 & 0 & 0 & \in R' \\
0 & 0 & 0 & \in R' \\
0 & 3 & 0 & \in R' \\
\| & \| & \| & \\
2 & 3 & 0 &
\end{array}$$

Tractability | Mal'tsev | Dalmau Algorithm

Example ($k = 3$, $A = \{0, 1, 2, 3, 4\}$)

$R = \{0, 1, 2, 3, 4\}^3$ admits Mal'tsev polymorphism $t(x, y, z) = x - y + z$,

$$t(x, y, y) = t(y, y, x) = x.$$

Define $R' = \{(a, 0, 0), (0, a, 0), (0, 0, a) \mid a \in \{0, 1, 2, 3, 4\}\} \subseteq R$. Check:

1. $|R'| \leq |A|k \leq 2|A|k \ll |A|^k = |R|$;
2. For instance, derive $(2, 3, 1) \in R$ from R' using t:

$$\begin{array}{cccl@{\qquad}cccl}
t & t & t & & & & & \\
2 & 0 & 0 & \in R' & 2 & 3 & 0 & \\
0 & 0 & 0 & \in R' & 0 & 0 & 0 & \in R' \\
0 & 3 & 0 & \in R' & 0 & 0 & 1 & \in R' \\
\| & \| & \| & & & & & \\
2 & 3 & 0 & & & & &
\end{array}$$

Tractability | Mal'tsev | Dalmau Algorithm

Example ($k = 3$, $A = \{0, 1, 2, 3, 4\}$)

$R = \{0, 1, 2, 3, 4\}^3$ admits Mal'tsev polymorphism $t(x, y, z) = x - y + z$,

$$t(x, y, y) = t(y, y, x) = x.$$

Define $R' = \{(a, 0, 0), (0, a, 0), (0, 0, a) \mid a \in \{0, 1, 2, 3, 4\}\} \subseteq R$. Check:

1. $|R'| \leq |A|k \leq 2|A|k \ll |A|^k = |R|$;
2. For instance, derive $(2, 3, 1) \in R$ from R' using t:

$$\begin{array}{ccccccccc}
t & t & t & & & t & t & t & \\
2 & 0 & 0 & \in R' & & 2 & 3 & 0 & \\
0 & 0 & 0 & \in R' & & 0 & 0 & 0 & \in R' \\
0 & 3 & 0 & \in R' & & 0 & 0 & 1 & \in R' \\
\| & \| & \| & & & \| & \| & \| & \\
2 & 3 & 0 & & & 2 & 3 & 1 & \in R
\end{array}$$

Outline

Related Problems

1. If a finite digraph admits Gumm polymorphisms, then it admits edge polymorphisms (Valeriote conjecture)?
2. Classify the complexity of the CSP: P/NP-complete?
3. Classify the complexity of the *quantified* CSP: P/NP-complete/PSPACE-complete?
4. Classify the complexity of the *valued* CSP.

Containment of Conjunctive Queries

Problem PPDEFCONT($\mathcal{C}$),
where $\mathcal{C}$ is the class of finite σ-structures.

Instance Two conjunctive queries ϕ_1 and ϕ_2, over σ,
with free variables $x_1, \ldots, x_n$, and a σ-structure $\mathbf{A}$.

Question $\mathbf{A} \models \phi_1 \subseteq \phi_2$, ie,
for all $\mathbf{b} \in A^{\{x_1,\ldots,x_n\}}$, $\mathbf{A}, \mathbf{b} \models \phi_1$ implies $\mathbf{A}, \mathbf{b} \models \phi_2$?

Containment of Conjunctive Queries

Problem $\text{PPDefCont}(\mathcal{C})$,
where $\mathcal{C}$ is the class of finite σ-structures.

Instance Two conjunctive queries ϕ_1 and ϕ_2, over σ,
with free variables $x_1, \ldots, x_n$, and a σ-structure $\mathbf{A}$.

Question $\mathbf{A} \models \phi_1 \subseteq \phi_2$, ie,
for all $\mathbf{b} \in A^{\{x_1,\ldots,x_n\}}$, $\mathbf{A}, \mathbf{b} \models \phi_1$ implies $\mathbf{A}, \mathbf{b} \models \phi_2$?

Theorem (B, Chen and Valeriote)

Let $\mathcal{C}$ be the class of finite σ-structures and $\mathbf{A} \in \mathcal{C}$.
Then $\text{PPDefCont}(\mathbf{A}) = \{(\phi_1, \phi_2) \mid \mathbf{A} \models \phi_1 \subseteq \phi_2\}$ *is:*

1. Π_2^p*-complete, if* $\mathbf{A}$ *omits Taylor polymorphisms;*
2. *coNP-complete, if* $\mathbf{A}$ *admits Taylor but omits Gumm polymorphisms;*
3. *in P, if* $\mathbf{A}$ *admits edge polymorphisms.*

Containment of Conjunctive Queries

Problem $\text{PPDefCont}(\mathcal{C})$,
where $\mathcal{C}$ is the class of finite σ-structures.

Instance Two conjunctive queries ϕ_1 and ϕ_2, over σ,
with free variables $x_1, \ldots, x_n$, and a σ-structure $\mathbf{A}$.

Question $\mathbf{A} \models \phi_1 \subseteq \phi_2$, ie,
for all $\mathbf{b} \in A^{\{x_1,\ldots,x_n\}}$, $\mathbf{A}, \mathbf{b} \models \phi_1$ implies $\mathbf{A}, \mathbf{b} \models \phi_2$?

Theorem (B, Chen and Valeriote)

Let $\mathcal{C}$ be the class of finite σ-structures and $\mathbf{A} \in \mathcal{C}$.
Then $\text{PPDefCont}(\mathbf{A}) = \{(\phi_1, \phi_2) \mid \mathbf{A} \models \phi_1 \subseteq \phi_2\}$ *is:*

1. Π_2^p*-complete, if $\mathbf{A}$ omits Taylor polymorphisms;*
2. *coNP-complete, if $\mathbf{A}$ admits Taylor but omits Gumm polymorphisms;*
3. *in P, if $\mathbf{A}$ admits edge polymorphisms.*

Remark

Complete trichotomy classification modulo Valeriote and BJK conjecture.

References

S. Bova, H. Chen, and M. Valeriote.
Generic Expression Hardness Results for Primitive Positive Formula Comparison.
Information and Computation, 2011.

L. Barto and M. Kozik.
Constraint Satisfaction Problems of Bounded Width.
In *Proceedings of FOCS'09*, 2009.

J. Berman, P. Idziak, P. Markovič, R. McKenzie, M. Valeriote, and R. Willard.
Varieties with Few Subalgebras of Powers.
Trans. Amer. Math. Soc., 362(3):1445–1473, 2010.

A. Bulatov and V. Dalmau.
A Simple Algorithm for Mal'tsev Constraints.
SIAM J. Comput., 36(1):16–27, 2006.

A. Bulatov, P. Jeavons, and A. Krokhin.
Classifying the Complexity of Constraints unsing Finite Algebras.
SIAM J. Comput., 34(3):720–742, 2005.

T. Feder and M. Vardi.
The Computational Structure of Monotone Monadic SNP and Constraint Satisfaction: a Study Through Datalog and Group Theory.
SIAM J. Comput., 28:57–104, 1998.

P. Hell and J. Nešetřil.
On the Complexity of H-Coloring.
Journal of Combinatorial Theory, B, 48:92–110, 1990.

Danke für Ihre Aufmerksamkeit.