Constraint Satisfaction with Succinctly Specified Relations

Martin Grohe

Department of Computer Science
Humboldt University Berlin

Joint work with Hubie Chen

CSP instances

$$\mathcal{I} = (\underbrace{V}_{\text{variables}}, \underbrace{D}_{\text{domain}}, \underbrace{C}_{\substack{\text{constraints} \\ \text{of the form } Rx_1 \ldots x_k}})$$

CSP instances

$$\mathcal{I} = (\underbrace{V}_{\text{variables}}, \underbrace{D}_{\text{domain}}, \underbrace{C}_{\substack{\text{constraints} \\ \text{of the form } Rx_1 \ldots x_k}})$$

Two relational structures associated with $\mathcal{I}$:

- $\mathcal{A}(\mathcal{I})$ structure induced by constraints on the variables.

CSP instances

$$\mathcal{I} = (\underbrace{V}_{\text{variables}}, \underbrace{D}_{\text{domain}}, \underbrace{C}_{\substack{\text{constraints} \\ \text{of the form } Rx_1 \ldots x_k}})$$

Two relational structures associated with $\mathcal{I}$:

- $\mathcal{A}(\mathcal{I})$ structure induced by constraints on the variables. Restrictions on $\mathcal{A}(\mathcal{I})$ are called structural restrictions.

CSP instances

$$\mathcal{I} = (\underbrace{V}_{\text{variables}}, \underbrace{D}_{\text{domain}}, \underbrace{C}_{\substack{\text{constraints} \\ \text{of the form } Rx_1 \ldots x_k}})$$

Two relational structures associated with $\mathcal{I}$:

- $\mathcal{A}(\mathcal{I})$ structure induced by constraints on the variables. Restrictions on $\mathcal{A}(\mathcal{I})$ are called structural restrictions.
- $\mathcal{B}(\mathcal{I})$ structure induced by constraint relations on the domain.

CSP instances

$$\mathcal{I} = (\underbrace{V}_{\text{variables}}, \underbrace{D}_{\text{domain}}, \underbrace{C}_{\substack{\text{constraints} \\ \text{of the form } Rx_1 \ldots x_k}})$$

Two relational structures associated with $\mathcal{I}$:

- $\mathcal{A}(\mathcal{I})$ structure induced by constraints on the variables. Restrictions on $\mathcal{A}(\mathcal{I})$ are called structural restrictions.
- $\mathcal{B}(\mathcal{I})$ structure induced by constraint relations on the domain.
  Restrictions on $\mathcal{B}(\mathcal{I})$ are called constraint language restrictions.

Uniform CSPs

For classes $\mathbf{K}, \mathbf{L}$ of relational structures:

CSP($\mathbf{K}, \mathbf{L}$)

Instance: CSP instance $\mathcal{I}$
with $\mathcal{A}(\mathcal{I}) \in \mathbf{K}$ and $\mathcal{B}(\mathcal{I}) \in \mathbf{L}$.

Problem: Decide if $\mathcal{I}$ has a solution.

Uniform CSPs

For classes **K**, **L** of relational structures:

CSP(**K**, **L**)	
Instance:	CSP instance $\mathcal{I}$ with $\mathcal{A}(\mathcal{I}) \in$ **K** and $\mathcal{B}(\mathcal{I}) \in$ **L**.
Problem:	Decide if $\mathcal{I}$ has a solution.

Remark

- We write CSP(**K**, $-$) or CSP($-$, **L**) if **L** resp. **K** is the class of all structures.

For classes **K**, **L** of relational structures:

CSP(**K**, **L**)	
Instance:	CSP instance $\mathcal{I}$ with $\mathcal{A}(\mathcal{I}) \in$ **K** and $\mathcal{B}(\mathcal{I}) \in$ **L**.
Problem:	Decide if $\mathcal{I}$ has a solution.

Remark

- We write CSP(**K**, $-$) or CSP($-$, **L**) if **L** resp. **K** is the class of all structures.
- We are mainly interested in restrictions of the form CSP(**K**, $-$) (structural restrictions).

Tractable structural restrictions

Two important ideas

1. Instances with a tree-like structure can be solved in polynomial time by dynamic programming.

Tractable structural restrictions

Two important ideas

1. Instances with a tree-like structure can be solved in polynomial time by dynamic programming.
2. It suffices for the structure of an instance to be homomorphically equivalent to a tree-like structure.

Tractable structural restrictions

Two important ideas

1. Instances with a tree-like structure can be solved in polynomial time by dynamic programming.
2. It suffices for the structure of an instance to be homomorphically equivalent to a tree-like structure.

Definition

A class of **K** structures has bounded tree width modulo homomorphic equivalence if there is a k such that every structure in **K** is homomorphically equivalent to a structure of tree width at most k.

Tractable structural restrictions

Two important ideas

1. Instances with a tree-like structure can be solved in polynomial time by dynamic programming.
2. It suffices for the structure of an instance to be homomorphically equivalent to a tree-like structure.

Definition

A class of **K** structures has bounded tree width modulo homomorphic equivalence if there is a k such that every structure in **K** is homomorphically equivalent to a structure of tree width at most k.

Theorem (Dalmau, Kolaitis, and Vardi 2002)

Let **K** *be a class of structures of bounded tree width modulo homomorphic equivalence.*
Then $\mathrm{CSP}(\mathbf{K}, -)$ *is tractable.*

Bounded arity

The arity of a structure is the maximum of the arities of its relations.

The arity of a structure is the maximum of the arities of its relations.

Theorem (G. 2003)

Assume that FPT $\neq$ W[1] *(or stronger:* 3-SAT $\notin$ DTIME($2^{o(n)}$)*).*

Bounded arity

The arity of a structure is the maximum of the arities of its relations.

Theorem (G. 2003)

Assume that FPT $\neq$ W[1] *(or stronger:* 3-SAT $\notin$ DTIME$(2^{o(n)})$*).*
For every recursively enumerable class **K** *of structures of bounded arity, the following are equivalent:*

1. CSP$(\mathbf{K}, -)$ *is tractable.*
2. **K** *has bounded tree width modulo homomorphic equivalence.*

Digression — a few remarks on the theorem

Theorem
Assume that FPT $\neq$ W[1] (or stronger: 3-SAT $\notin$ DTIME($2^{o(n)}$)).
For every recursively enumerable class **K** of structures of bounded arity, the following are equivalent:
1. CSP(**K**, $-$) is tractable.
2. **K** has bounded tree width modulo homomorphic equivalence.

Digression — a few remarks on the theorem

Theorem
Assume that FPT $\neq$ W[1] (or stronger: 3-SAT $\notin$ DTIME($2^{o(n)}$)).
For every recursively enumerable class **K** of structures of bounded arity, the following are equivalent:
1. CSP(**K**, $-$) is tractable.
2. **K** has bounded tree width modulo homomorphic equivalence.

- This is not a dichotomy theorem.

Digression — a few remarks on the theorem

Theorem

Assume that FPT $\neq$ W[1] (or stronger: 3-SAT $\notin$ DTIME($2^{o(n)}$)).
For every recursively enumerable class **K** of structures of bounded arity, the following are equivalent:

1. CSP(**K**, $-$) is tractable.
2. **K** has bounded tree width modulo homomorphic equivalence.

- This is not a dichotomy theorem.
  There are classes **K** of graphs of unbounded treewidth modulo homomorphic equivalence such that CSP(**K**, $-$) $\in$ NP[$\log^2 n$].

Digression — a few remarks on the theorem

Theorem
Assume that FPT $\neq$ W[1] (or stronger: 3-SAT $\notin$ DTIME($2^{o(n)}$)).
For every recursively enumerable class **K** of structures of bounded arity, the following are equivalent:
1. CSP(**K**, $-$) is tractable.
2. **K** has bounded tree width modulo homomorphic equivalence.

- This is not a dichotomy theorem.
- The assumption FPT $\neq$ W[1] cannot be replaced by anything weaker.

Digression — a few remarks on the theorem

Theorem
Assume that FPT $\neq$ W[1] (or stronger: 3-SAT $\notin$ DTIME($2^{o(n)}$)).
For every recursively enumerable class **K** of structures of bounded arity, the following are equivalent:
1. CSP(**K**, $-$) is tractable.
2. **K** has bounded tree width modulo homomorphic equivalence.

- This is not a dichotomy theorem.
- The assumption FPT $\neq$ W[1] cannot be replaced by anything weaker.
  If FPT $=$ W[1] then there are classes **K** of graphs of unbounded treewidth modulo homomorphic equivalence such that CSP(**K**, $-$) is tractable.

Digression — a few remarks on the theorem

Theorem
Assume that FPT $\neq$ W[1] (or stronger: 3-SAT $\notin$ DTIME($2^{o(n)}$)).
For every recursively enumerable class **K** of structures of bounded arity, the following are equivalent:
1. CSP(**K**, $-$) is tractable.
2. **K** has bounded tree width modulo homomorphic equivalence.

- This is not a dichotomy theorem.
- The assumption FPT $\neq$ W[1] cannot be replaced by anything weaker.
- The requirement that **K** be recursively enumerable is inessential.

Digression — a few remarks on the theorem

Theorem
Assume that FPT $\neq$ W[1] (or stronger: 3-SAT $\notin$ DTIME($2^{o(n)}$)).
For every recursively enumerable class **K** of structures of bounded arity, the following are equivalent:
1. CSP(**K**, $-$) is tractable.
2. **K** has bounded tree width modulo homomorphic equivalence.

- This is not a dichotomy theorem.
- The assumption FPT $\neq$ W[1] cannot be replaced by anything weaker.
- The requirement that **K** be recursively enumerable is inessential. With a slightly stronger complexity theoretic assumption, we can get rid of it.

Digression — a few remarks on the theorem

Theorem
Assume that FPT $\neq$ W[1] (or stronger: 3-SAT $\notin$ DTIME($2^{o(n)}$)).
For every recursively enumerable class **K** of structures of bounded arity, the following are equivalent:
1. CSP(**K**, $-$) is tractable.
2. **K** has bounded tree width modulo homomorphic equivalence.

- This is not a dichotomy theorem.
- The assumption FPT $\neq$ W[1] cannot be replaced by anything weaker.
- The requirement that **K** be recursively enumerable is inessential.
- The requirement that **K** be of bounded arity is crucial!

Digression — a few remarks on the theorem

Theorem
Assume that FPT $\neq$ W[1] (or stronger: 3-SAT $\notin$ DTIME($2^{o(n)}$)).
For every recursively enumerable class **K** of structures of bounded arity, the following are equivalent:
1. CSP(**K**, $-$) is tractable.
2. **K** has bounded tree width modulo homomorphic equivalence.

- This is not a dichotomy theorem.
- The assumption FPT $\neq$ W[1] cannot be replaced by anything weaker.
- The requirement that **K** be recursively enumerable is inessential.
- The requirement that **K** be of bounded arity is crucial!

A new version of the paper is on my webpage:

http://www.informatik.hu-berlin.de/~grohe/pub/gro06b+.html

A third idea

If all variables are covered by a bounded number of constraints, then the number of solutions is polynomially bounded.

A third idea

If all variables are covered by a bounded number of constraints, then the number of solutions is polynomially bounded.

Let k be a constant and $\mathcal{I} = (V, D, C)$ a CSP-instance. Suppose that the constraints

$$R_1 x_{11} \dots x_{1r_1}, \dots, R_k x_{k1} \dots x_{kr_k}$$

cover all variables, i.e., $V \subseteq \{x_{ij} \mid 1 \leq i \leq k, 1 \leq j \leq r_i\}$.

A third idea

If all variables are covered by a bounded number of constraints, then the number of solutions is polynomially bounded.

Let k be a constant and $\mathcal{I} = (V, D, C)$ a CSP-instance.
Suppose that the constraints

$$R_1 x_{11} \dots x_{1r_1}, \dots, R_k x_{k1} \dots x_{kr_k}$$

cover all variables, i.e., $V \subseteq \{x_{ij} \mid 1 \leq i \leq k, 1 \leq j \leq r_i\}$.
Then for the set S of all solutions of $\mathcal{I}$, we have:

$$S \subseteq R_1 \bowtie \cdots \bowtie R_k$$

($S \subseteq R_1 \times \cdots \times R_k$ if the variable sets are disjoint).

A third idea

If all variables are covered by a bounded number of constraints, then the number of solutions is polynomially bounded.

Let k be a constant and $\mathcal{I} = (V, D, C)$ a CSP-instance. Suppose that the constraints

$$R_1 x_{11} \ldots x_{1r_1}, \ldots, R_k x_{k1} \ldots x_{kr_k}$$

cover all variables, i.e., $V \subseteq \{x_{ij} \mid 1 \leq i \leq k, 1 \leq j \leq r_i\}$. Then for the set S of all solutions of $\mathcal{I}$, we have:

$$S \subseteq R_1 \bowtie \cdots \bowtie R_k$$

($S \subseteq R_1 \times \cdots \times R_k$ if the variable sets are disjoint). Hence

$$|S| \leq |R_1| \cdot |R_2| \cdots |R_k| \leq ||\mathcal{I}||^k,$$

and S can be computed in polynomial time.

Variants and extensions

- The set of solutions is still polynomial if the instance has a fractional edge cover of bounded size, and this set can be computed in polynomial time.

Variants and extensions

- The set of solutions is still polynomial if the instance has a fractional edge cover of bounded size, and this set can be computed in polynomial time.
- These ideas can be combined with tree decompositions, leading to hypertree width and fractional hypertree width.

Variants and extensions

- The set of solutions is still polynomial if the instance has a fractional edge cover of bounded size, and this set can be computed in polynomial time.
- These ideas can be combined with tree decompositions, leading to hypertree width and fractional hypertree width.

That's all great — but wait . . .

...haven't we just proved $(N = 1 \vee P = 0)$

- If viewed as a (Boolean) CSP, the satisfiability problem SAT(Φ) for the following class of formulas is tractable:

$$\Phi := \{\phi \wedge (x_1 \vee \ldots \vee x_n) \mid n \geq 1, \phi \text{ CNF formula with variables among } x_1, \ldots, x_n\}$$

...haven't we just proved $(N = 1 \vee P = 0)$

- If viewed as a (Boolean) CSP, the satisfiability problem SAT(Φ) for the following class of formulas is tractable:

$$\Phi := \left\{ \phi \wedge (x_1 \vee \ldots \vee x_n) \;\middle|\; n \geq 1, \phi \text{ CNF formula with variables among } x_1, \ldots, x_n \right\}$$

- SAT is easily reducible to SAT(Φ) via

$$\phi \mapsto \phi \wedge (x_1 \vee \ldots \vee x_{n+1}) \quad \text{for } \phi \text{ with variables among } x_1, \ldots, x_n.$$

. . . haven't we just proved $(N = 1 \vee P = 0)$

- If viewed as a (Boolean) CSP, the satisfiability problem SAT(Φ) for the following class of formulas is tractable:

$$\Phi := \{\phi \wedge (x_1 \vee \ldots \vee x_n) \mid n \geq 1, \phi \text{ CNF formula with variables among } x_1, \ldots, x_n\}$$

- SAT is easily reducible to SAT(Φ) via

$$\phi \mapsto \phi \wedge (x_1 \vee \ldots \vee x_{n+1}) \quad \text{for } \phi \text{ with variables among } x_1, \ldots, x_n.$$

- Hence P = NP.

Conclusion

- By a simple reduction to constraint satisfaction problems, we have proved the satisfiability problem to be in polynomial time and hence $\mathrm{P} = \mathrm{NP}$.
- This impressively shows the power of the algorithmic ideas developed in the area.

Explicit and implicit representations

- We usually assume the constraint relations to be specified explicitly, that is, by listing all tuples.

Explicit and implicit representations

- We usually assume the constraint relations to be specified explicitly, that is, by listing all tuples.
- As long as the arity is bounded (in particular for finite constraint languages), this is irrelevant, because the size of the relations is polynomially bounded in terms of the domain.

Explicit and implicit representations

- We usually assume the constraint relations to be specified explicitly, that is, by listing all tuples.
- As long as the arity is bounded (in particular for finite constraint languages), this is irrelevant, because the size of the relations is polynomially bounded in terms of the domain.
- If the arity is unbounded, the size of the relations may be exponential in the domain size, which affects the complexity.

Explicit and implicit representations

- We usually assume the constraint relations to be specified explicitly, that is, by listing all tuples.
- As long as the arity is bounded (in particular for finite constraint languages), this is irrelevant, because the size of the relations is polynomially bounded in terms of the domain.
- If the arity is unbounded, the size of the relations may be exponential in the domain size, which affects the complexity.
- For problems of unbounded arity that occur in practice, e.g., SAT, LINEAR PROGRAMMING, constraints are usually specified implicitly, e.g. by clauses or linear inequalities. This makes the problems harder!

Explicit and implicit representations

- We usually assume the constraint relations to be specified explicitly, that is, by listing all tuples.
- As long as the arity is bounded (in particular for finite constraint languages), this is irrelevant, because the size of the relations is polynomially bounded in terms of the domain.
- If the arity is unbounded, the size of the relations may be exponential in the domain size, which affects the complexity.
- For problems of unbounded arity that occur in practice, e.g., SAT, LINEAR PROGRAMMING, constraints are usually specified implicitly, e.g. by clauses or linear inequalities. This makes the problems harder!

In this work, we try to initiate a systematic study of the complexity of CSP with implicitly represented relations.

The Boolean case

Obvious idea

Specify constraints by Boolean circuits or formulas.

The Boolean case

Obvious idea

Specify constraints by Boolean circuits or formulas.

Problem

Even structurally very simple instances become hard:
The Boolean CSP with one n-ary constraint

$$Rx_1 \dots x_n,$$

where R is represented by a Boolean circuit, is NP-complete.

The Boolean case

Obvious idea

Specify constraints by Boolean circuits or formulas.

Problem

Even structurally very simple instances become hard:
The Boolean CSP with one n-ary constraint

$$Rx_1 \ldots x_n,$$

where R is represented by a Boolean circuit, is NP-complete.
This remains true if the relations are specified by CNF-formulas, or any class of formulas with a hard satisfiability problem.

Obvious idea

Specify constraints by Boolean circuits or formulas.

Problem

Even structurally very simple instances become hard:
The Boolean CSP with one n-ary constraint

$$Rx_1 \dots x_n,$$

where R is represented by a Boolean circuit, is NP-complete.
This remains true if the relations are specified by CNF-formulas, or any class of formulas with a hard satisfiability problem.

DNF Representations

DNF-formulas form natural large class of formulas with a tractable satisfiability problem.

The Boolean case

Obvious idea

Specify constraints by Boolean circuits or formulas.

Problem

Even structurally very simple instances become hard:
The Boolean CSP with one n-ary constraint

$$Rx_1 \dots x_n,$$

where R is represented by a Boolean circuit, is NP-complete.
This remains true if the relations are specified by CNF-formulas, or any class of formulas with a hard satisfiability problem.

DNF Representations

DNF-formulas form natural large class of formulas with a tractable satisfiability problem.
Therefore, we decided to choose DNF-formulas as succinct representations of Boolean constraint relations.

Arbitrary domains — the GDNF-representation

Definition

Let D be a set. A generalised DNF (GDNF) representation of a relation $R \subseteq D^k$ is an expression of the form

$$\bigcup_{i=1}^{m} (P_{i1} \times \cdots \times P_{ik})$$

where $m \geq 0$ and $P_{ij} \subseteq D$ for $1 \leq i \leq m, 1 \leq j \leq k$.

Arbitrary domains — the GDNF-representation

Definition

Let D be a set. A generalised DNF (GDNF) representation of a relation $R \subseteq D^k$ is an expression of the form

$$\bigcup_{i=1}^{m} (P_{i1} \times \cdots \times P_{ik})$$

where $m \geq 0$ and $P_{ij} \subseteq D$ for $1 \leq i \leq m, 1 \leq j \leq k$.

Remark

The GDNF enables us to represent relations of size $\Omega(D^k)$ by expressions of size $O(m \cdot |D| \cdot k)$.

Example

Notation

For all $n \in \mathbb{N}$: $\quad [n] = \{1, \ldots, n\}$.

Example

Notation

For all $n \in \mathbb{N}$: $[n] = \{1, \ldots, n\}$.

Example

Let MAX $\subseteq [n]^{k+1}$ be defined by

$$\text{MAX} = \big\{(x_1, \ldots, x_k, y) \bigm| y = \max\{x_1, \ldots, x_k\}\big\}$$

Then $|\text{MAX}| = \Omega(n^k)$.

Example

Notation

For all $n \in \mathbb{N}$: $[n] = \{1, \ldots, n\}$.

Example

Let MAX $\subseteq [n]^{k+1}$ be defined by

$$\text{MAX} = \big\{(x_1, \ldots, x_k, y) \;\big|\; y = \max\{x_1, \ldots, x_k\}\big\}$$

Then $|\text{MAX}| = \Omega(n^k)$.

MAX has the following GDNF-representation of size $O(k^2 \cdot n^2)$:

$$\bigcup_{\substack{1 \le i \le k \\ 1 \le j \le n}} \Big([j] \times \ldots \times [j] \times \underbrace{\{j\}}_{i\text{th place}} \times [j] \times \ldots \times [j] \times \underbrace{\{j\}}_{(k+1)\text{st place}} \Big).$$

Succinctly specified CSP

Definition

- A CSP-instance $\mathcal{I}$ is represented in GDNF if all constraint relations are represented in GDNF.

Succinctly specified CSP

Definition

- A CSP-instance $\mathcal{I}$ is represented in GDNF if all constraint relations are represented in GDNF.
- By $\mathrm{CSP}_{\mathrm{GDNF}}(\mathbf{K}, \mathbf{L})$ is the version of the problem $\mathrm{CSP}(\mathbf{K}, \mathbf{L})$ where instances are represented in GDNF.

Definition

1. Incidence vocabulary of a relational vocabulary τ:

$$\tau_I = \{R_1, \dots, R_k \mid R \in \tau \text{ } k\text{-ary}\}$$

(R_i binary).

Definition

1. Incidence vocabulary of a relational vocabulary τ:

$$\tau_I = \{R_1, \dots, R_k \mid R \in \tau \text{ } k\text{-ary}\}$$

   (R_i binary).
2. Incidence structure of a τ-structure $\mathcal{A}$: τ_I-structure $\mathcal{A}_I$ with

Definition

1. Incidence vocabulary of a relational vocabulary τ:

$$\tau_I = \{R_1, \ldots, R_k \mid R \in \tau \ k\text{-ary}\}$$

(R_i binary).

2. Incidence structure of a τ-structure $\mathcal{A}$: τ_I-structure $\mathcal{A}_I$ with
   - universe

$$A_I = A \cup \bigcup_{R \in \sigma} \{(R, a_1, \ldots, a_k) : (a_1, \ldots, a_k) \in R^{\mathcal{A}}\},$$

The incidence structure

Definition

1. Incidence vocabulary of a relational vocabulary τ:

$$\tau_I = \{R_1, \ldots, R_k \mid R \in \tau \ k\text{-ary}\}$$

(R_i binary).

2. Incidence structure of a τ-structure $\mathcal{A}$: τ_I-structure $\mathcal{A}_I$ with
   - universe

$$A_I = A \cup \bigcup_{R \in \sigma} \{(R, a_1, \ldots, a_k) : (a_1, \ldots, a_k) \in R^{\mathcal{A}}\},$$

   - relations

$$R_i^{\mathcal{A}_I} = \{((R, a_1, \ldots, a_k), a_i) : (a_1, \ldots, a_k) \in R^{\mathcal{A}}\}$$

   for all k-ary $R \in \tau$ and $i \in [k]$.

Example

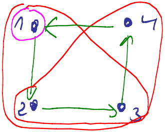
$\{R_3, E_2, P_1\}$ - structure $\mathcal{A}$:

$A = \{1, 2, 3, 4\}$

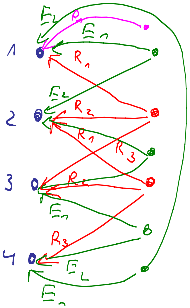
$R^{\mathcal{A}} = \{(1,2,3), (2,3,4)\}$

$E^{\mathcal{A}} = \{(1,2), (2,3), (3,4), (4,1)\}$

$P^{\mathcal{A}} = \{(1)\}$



The structure $\mathcal{A}_I$



Definition

The incidence width $\text{iw}(\mathcal{A})$ of a relational structure $\mathcal{A}$ is the tree width of its incidence structure.

Incidence width

Definition
The incidence width $\mathrm{iw}(\mathcal{A})$ of a relational structure $\mathcal{A}$ is the tree width of its incidence structure.

Remark
For all structures $\mathcal{A}$ of arity k:

$$\mathrm{iw}(\mathcal{A}) - 1 \leq \mathrm{tw}(\mathcal{A}) \leq (\mathrm{iw}(\mathcal{A}) + 1) \cdot (k - 1).$$

Theorem

Assume that FPT $\neq$ W[1].

For every recursively enumerable class **K** *of structures of bounded arity, the following are equivalent:*

1. $\mathrm{CSP}_{\mathrm{GDNF}}(\mathbf{K}, -)$ *is tractable.*
2. **K** *has bounded incidence width modulo homomorphic equivalence.*

Constraint language restrictions

Theorem

Let $\mathbf{L}_k$ *be the class of all relational structures having a near-unanimity polymorphism of arity* k*. For each* $k \geq 3$*, the problem* $\mathrm{CSP}_{\mathrm{GDNF}}(-, \mathbf{L}_k)$ *is tractable.*

Constraint language restrictions

Theorem

Let $\mathbf{L}_k$ be the class of all relational structures having a near-unanimity polymorphism of arity k. For each $k \geq 3$, the problem $\mathrm{CSP}_{\mathrm{GDNF}}(-, \mathbf{L}_k)$ is tractable.

Theorem

Let $\mathbf{L}$ be the set of all relational structures invariant under a set function. The problem $\mathrm{CSP}_{\mathrm{GDNF}}(-, \mathbf{L})$ is tractable.

- Ordered binary decision diagrams (OBDDs) are another representation of Boolean relations with nice algorithmic properties. In particular, the satisfiability problem is decidable.

Other representations

- Ordered binary decision diagrams (OBDDs) are another representation of Boolean relations with nice algorithmic properties. In particular, the satisfiability problem is decidable.
- We considered the natural generalization of OBDDs to arbitrary finite domains and obtained initial results.