Constraint satisfaction with homogeneous templates

Applications of model theory and Ramsey theory
in theoretical computer science

Michael Pinsker

Université Denis Diderot - Paris 7 (60%)
Technische Universität Wien (30%)
Hebrew University of Jerusalem (10%)

MALOA Training Workshop, 2011

Outline

Constraint satisfaction problems (CSPs)

Constraint satisfaction - general

An *instance* of a *constraint satisfaction problem (CSP)* consists of:

- A finite set of variables
- Constraints for these variables

(the kind of constraints and possible values of the variables being defined by the CSP)

A *solution* to the CSP is an assignment of values to the variables such that all constraints are satisfied.

Examples:

- Sudokus
- Equations

We will consider a special kind of CSPs, namely Homomorphism problems.

Interested in the *complexity* of solving the problem.

Relational Structures

Let τ be a relational signature, i.e., a set of relation symbols R_i, each associated with a finite arity k_i.

Relational Structures

Let τ be a relational signature, i.e., a set of relation symbols R_i, each associated with a finite arity k_i.

A τ-structure $\Gamma = (D; R_1^\Gamma, R_2^\Gamma, \dots)$ is a set D together with a relation $R_i^\Gamma \subseteq D^{k_i}$ for each relation symbol R_i of arity k_i in τ.

Relational Structures

Let τ be a relational signature, i.e., a set of relation symbols R_i, each associated with a finite arity k_i.

A τ-structure $\Gamma = (D; R_1^\Gamma, R_2^\Gamma, \dots)$ is a set D together with a relation $R_i^\Gamma \subseteq D^{k_i}$ for each relation symbol R_i of arity k_i in τ.

Examples:

- Graphs $G = (V; E)$, Digraphs
- Vertex-colored graphs
- Graphs with different types of edges
- Hypergraphs
- Databases
- Mathematical structures: $(\mathbb{N}; \neq)$, $(\mathbb{Q}; <, \leq, \neq, =)$, $(\mathbb{R}; \{(x, y)\ x^2 + y^2 \leq 1\})$, $\dots$

Homomorphisms

Let Δ and Γ be structures with the same relational signature τ.

Definition

A function $f : \Delta \to \Gamma$ is called homomorphism iff for each k-ary relation symbol R of τ

$$(a_1, \ldots, a_k) \in R^\Delta \;\to\; (f(a_1), \ldots, f(a_k)) \in R^\Gamma \; .$$

The Constraint Satisfaction Problem

Let Γ be a structure with a finite relational signature τ.
Γ also called the template.

Definition

CSP(Γ) is the computational problem to decide whether a given finite τ-structure Δ homomorphically maps to Γ.

Note: Γ need not be finite.

The Constraint Satisfaction Problem

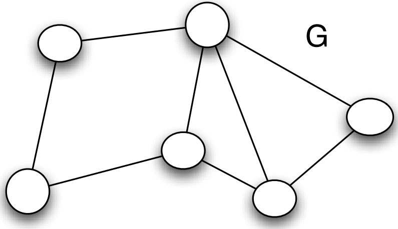
Let Γ be a structure with a finite relational signature τ.
Γ also called the template.

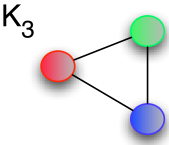
Definition

CSP(Γ) is the computational problem to decide whether a given finite τ-structure Δ homomorphically maps to Γ.

Note: Γ need not be finite.

Example: 3-colorability is CSP(K_3)





Examples of CSPs

Positive 1-in-3-3SAT

Input: A set of triples of variables (x, y, z)

Question: Is there a 0/1-assignment to the variables such that in each clause exactly one variable is true?

Examples of CSPs

Positive 1-in-3-3SAT

Input: A set of triples of variables (x, y, z)

Question: Is there a 0/1-assignment to the variables such that in each clause exactly one variable is true?

Is a CSP: template is $(\{0,1\};\{(0,0,1),(0,1,0),(1,0,0)\})$

Examples of CSPs

Positive 1-in-3-3SAT

Input: A set of triples of variables (x, y, z)

Question: Is there a 0/1-assignment to the variables such that in each clause exactly one variable is true?

Is a CSP: template is $(\{0,1\};\{(0,0,1),(0,1,0),(1,0,0)\})$

Digraph acyclicity

Input: A directed graph $(V;E)$

Question: Is $(V;E)$ acyclic?

Examples of CSPs

Positive 1-in-3-3SAT

Input: A set of triples of variables (x, y, z)

Question: Is there a 0/1-assignment to the variables such that in each clause exactly one variable is true?

Is a CSP: template is $(\{0,1\};\{(0,0,1),(0,1,0),(1,0,0)\})$

Digraph acyclicity

Input: A directed graph $(V; E)$

Question: Is $(V; E)$ acyclic?

Is CSP: template is $(\mathbb{Q}; <)$

More Examples of CSPs

Betweenness:

Input: A set of triples of variables (x, y, z)

Question: Is there a weak linear order on the variables such that for each triple either $x < y < z$ or $z < y < x$?

More Examples of CSPs

Betweenness:

Input: A set of triples of variables (x, y, z)

Question: Is there a weak linear order on the variables such that for each triple either $x < y < z$ or $z < y < x$?

Is a CSP: template is $(\mathbb{Q}; \{(x, y, z) \mid (x < y < z) \vee (z < y < x)\})$

More Examples of CSPs

Betweenness:

Input: A set of triples of variables (x, y, z)

Question: Is there a weak linear order on the variables such that for each triple either $x < y < z$ or $z < y < x$?

Is a CSP: template is $(\mathbb{Q}; \{(x, y, z) \mid (x < y < z) \vee (z < y < x)\})$

And/Or-Precedence-Constraints:

Input: A set of triples of variables (x, y, z)

Question: Is there a weak linear order on the variables such that for each triple x is strictly larger than the minimum of y and z?

More Examples of CSPs

Betweenness:

Input: A set of triples of variables (x, y, z)

Question: Is there a weak linear order on the variables such that for each triple either $x < y < z$ or $z < y < x$?

Is a CSP: template is $(\mathbb{Q}; \{(x, y, z) \mid (x < y < z) \vee (z < y < x)\})$

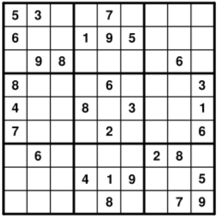
And/Or-Precedence-Constraints:

Input: A set of triples of variables (x, y, z)

Question: Is there a weak linear order on the variables such that for each triple x is strictly larger than the minimum of y and z?

Is a CSP: template is $(\mathbb{Q}; \{(x, y, z) \mid (x > y) \vee (x > z)\})$

Sudokus



5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Sudokus

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

A template for **Sudokus**:

$$\Gamma_S = (\{1, 2, \ldots, 9\}; R, P_1, \ldots, P_9)\ ,$$

where $R = \{(t_1, \ldots, t_9) \mid |\{t_1, \ldots, t_9\}| = 9\}$,
and $P_i = \{i\}$ for all $1 \leq i \leq 9$.

Sudokus

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

A template for **Sudokus**:

$$\Gamma_S = (\{1, 2, \ldots, 9\}; R, P_1, \ldots, P_9) \ ,$$

where $R = \{(t_1, \ldots, t_9) \mid |\{t_1, \ldots, t_9\}| = 9\}$,
and $P_i = \{i\}$ for all $1 \leq i \leq 9$.

- Every Sudoku can be formulated as an instance of $\mathrm{CSP}(\Gamma_S)$
- Not all instances of $\mathrm{CSP}(\Gamma_S)$ correspond to a Sudoku.

Diophantine equations

Diophantine:

Input: An equation using $=, +, \cdot, 1$

Question: Is there a solution to the equation in $\mathbb{Z}$?

Diophantine equations

Diophantine:

Input: An equation using $=, +, \cdot, 1$

Question: Is there a solution to the equation in $\mathbb{Z}$?

Is a CSP: template is $\Gamma := (\mathbb{Z}; 1, +, \cdot, =)$.

Which problems can be formulated as CSPs?

Let $\mathcal{C}$ be a class of τ-structures.

Definition

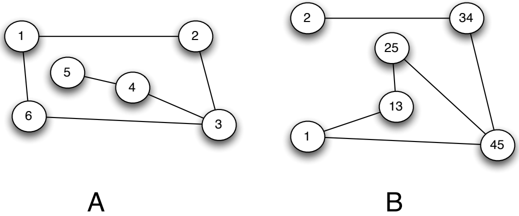
$\mathcal{C}$ is closed under disjoint unions iff whenever $A, B \in \mathcal{C}$ then $A \mathbin{\dot\cup} B \in \mathcal{C}$.

Which problems can be formulated as CSPs?

Let $\mathcal{C}$ be a class of τ-structures.

Definition

$\mathcal{C}$ is closed under disjoint unions iff whenever $A, B \in \mathcal{C}$ then $A \dot{\cup} B \in \mathcal{C}$.

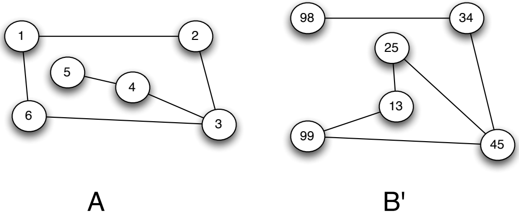


Which problems can be formulated as CSPs?

Let $\mathcal{C}$ be a class of τ-structures.

Definition

$\mathcal{C}$ is closed under disjoint unions iff whenever $A, B \in \mathcal{C}$ then $A \mathbin{\dot\cup} B \in \mathcal{C}$.

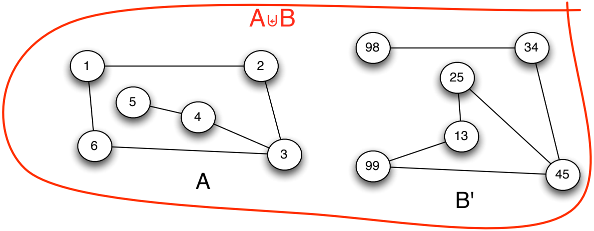


Which problems can be formulated as CSPs?

Let $\mathcal{C}$ be a class of τ-structures.

Definition

$\mathcal{C}$ is closed under disjoint unions iff whenever $A, B \in \mathcal{C}$ then $A \dot\cup B \in \mathcal{C}$.



Inverse homomorphisms

Let $\mathcal{C}$ be a class of τ-structures.

Definition

$\mathcal{C}$ is closed under inverse homomorphisms iff $B \in \mathcal{C}$ and $A \to B$ implies that $A \in \mathcal{C}$.

Inverse homomorphisms

Let $\mathcal{C}$ be a class of τ-structures.

Definition

$\mathcal{C}$ is closed under inverse homomorphisms iff $B \in \mathcal{C}$ and $A \to B$ implies that $A \in \mathcal{C}$.

Example: the set of all triangle-free graphs is closed under disjoint unions and inverse homomorphisms.

Inverse homomorphisms

Let $\mathcal{C}$ be a class of τ-structures.

Definition

$\mathcal{C}$ is closed under inverse homomorphisms iff $B \in \mathcal{C}$ and $A \to B$ implies that $A \in \mathcal{C}$.

Example: the set of all triangle-free graphs is closed under disjoint unions and inverse homomorphisms.

Observation:

CSP(Γ) can be viewed as a class of finite structures: all those structures that homomorphically map to Γ.

Inverse homomorphisms

Let $\mathcal{C}$ be a class of τ-structures.

Definition

$\mathcal{C}$ is closed under inverse homomorphisms iff $B \in \mathcal{C}$ and $A \to B$ implies that $A \in \mathcal{C}$.

Example: the set of all triangle-free graphs is closed under disjoint unions and inverse homomorphisms.

Observation:

CSP(Γ) can be viewed as a class of finite structures: all those structures that homomorphically map to Γ.

CSP(Γ) is closed under disjoint unions.

Inverse homomorphisms

Let $\mathcal{C}$ be a class of τ-structures.

Definition

$\mathcal{C}$ is closed under inverse homomorphisms iff $B \in \mathcal{C}$ and $A \rightarrow B$ implies that $A \in \mathcal{C}$.

Example: the set of all triangle-free graphs is closed under disjoint unions and inverse homomorphisms.

Observation:

CSP(Γ) can be viewed as a class of finite structures: all those structures that homomorphically map to Γ.

CSP(Γ) is closed under disjoint unions.

CSP(Γ) is closed under inverse homomorphisms.

Fundamental Lemma

Lemma (Feder ’93)

$\mathcal{C} =$CSP(Γ) for some relational structure Γ if and only if $\mathcal{C}$ is closed under disjoint unions and inverse homomorphisms.

Proof. It remains to show the ‘if’-part of the statement.

Fundamental Lemma

Lemma (Feder '93)

$\mathcal{C} =$CSP(Γ) for some relational structure Γ if and only if $\mathcal{C}$ is closed under disjoint unions and inverse homomorphisms.

Proof. It remains to show the 'if'-part of the statement.

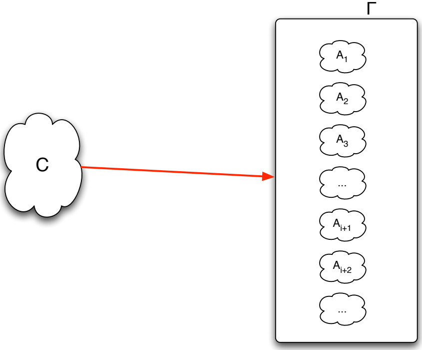
Choose Γ as $\dot{\bigcup}_{A \in \mathcal{C}} A$

Fundamental Lemma

Lemma (Feder '93)

$\mathcal{C}$ $=$CSP(Γ) for some relational structure Γ if and only if $\mathcal{C}$ is closed under disjoint unions and inverse homomorphisms.

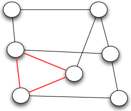
Proof. It remains to show the 'if'-part of the statement. Choose Γ as $\dot{\bigcup}_{A\in\mathcal{C}} A$



☐

Examples of CSPs

Triangle-Freeness:

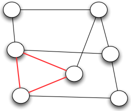


Input: A graph *G*

Question: Is *G* triangle-free?

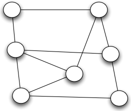
Examples of CSPs

Triangle-Freeness:



Input: A graph G

Question: Is G triangle-free?



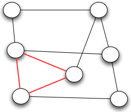
No-Mono-Tri:

Input: A graph G

Question: Can we partition $V(G) = V_1 \uplus V_2$ such that $G[V_1]$ and $G[V_2]$ are triangle-free?

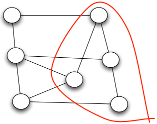
Examples of CSPs

Triangle-Freeness:



Input: A graph G

Question: Is G triangle-free?



No-Mono-Tri:

Input: A graph G

Question: Can we partition $V(G) = V_1 \uplus V_2$ such that $G[V_1]$ and $G[V_2]$ are triangle-free?

Exercises

Prove that the following two problems can be formulated as CSPs with an infinite template. For each problem, give two proofs: one using the previous lemma, and one by direct construction of the template.

Acylic Bipartition:

Input: A digraph G

Question: Can we partition $V(G) = V_1 \uplus V_2$ such that $G[V_1]$ and $G[V_2]$ are acyclic?

Cyclic Embedding:

Input: A digraph G

Question: Can we map $V(G)$ to the plane such that all arcs in $E(G)$ have the origin on the left side?

Logical Perspective on CSPs

Logical Perspective on CSPs

Let τ be a relational signature.

Definition

A primitive positive τ-formula is a first-order τ-formula of the form

$$\exists x_1, \ldots, x_n . \psi_1 \wedge \cdots \wedge \psi_m \ ,$$

where $\psi_1, \ldots, \psi_m$ are atomic formulas, i.e., formulas of the form $x = y$ or of the form $R(x_{i_1}, \ldots, x_{i_k})$ for $R \in \tau$.

Logical Perspective on CSPs

Let τ be a relational signature.

Definition

A primitive positive τ-formula is a first-order τ-formula of the form

$$\exists x_1, \ldots, x_n . \psi_1 \wedge \cdots \wedge \psi_m \, ,$$

where $\psi_1, \ldots, \psi_m$ are atomic formulas, i.e., formulas of the form $x = y$ or of the form $R(x_{i_1}, \ldots, x_{i_k})$ for $R \in \tau$.

A sentence is a formula without free variables.

Logical Perspective on CSPs

Let τ be a relational signature.

Definition

A primitive positive τ-formula is a first-order τ-formula of the form

$$\exists x_1, \ldots, x_n . \psi_1 \wedge \cdots \wedge \psi_m \ ,$$

where $\psi_1, \ldots, \psi_m$ are atomic formulas, i.e., formulas of the form $x = y$ or of the form $R(x_{i_1}, \ldots, x_{i_k})$ for $R \in \tau$.

A sentence is a formula without free variables.

Alternative definition of CSP(Γ):

Input: a primitive positive sentence Φ

Question: Is Φ true in Γ?

The Canonical Query

Let Δ be a finite τ-structure with domain D.

Definition

The canonical query $\Phi(\Delta)$ of Δ is the primitive positive formula with existentially quantified variables D that contains a conjunct $R(a_1, \ldots, a_n)$ iff $(a_1, \ldots, a_n) \in R^\Delta$.

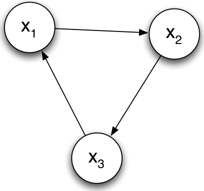
The Canonical Query

Let Δ be a finite τ-structure with domain D.

Definition

The canonical query $\Phi(\Delta)$ of Δ is the primitive positive formula with existentially quantified variables D that contains a conjunct $R(a_1, \ldots, a_n)$ iff $(a_1, \ldots, a_n) \in R^\Delta$.

Example:



$$\Phi(\Delta) := \exists x_1, x_2, x_3.\ E(x_1, x_2) \wedge E(x_2, x_3) \wedge E(x_3, x_1)$$

Homomorphisms vs Logic

Let Γ be a τ-structure.

Lemma

For any finite τ-structure Δ the following are equivalent.

- *There is a homomorphism from Δ to Γ.*
- *$\Phi(\Delta)$ is true in Γ.*

Homomorphisms vs Logic

Let Γ be a τ-structure.

Lemma

For any finite τ-structure Δ the following are equivalent.

- *There is a homomorphism from Δ to Γ.*
- $\Phi(\Delta)$ *is true in* Γ.

Proof. trivial.

Canonical Database

Definition

For each primitive positive sentence Φ over signature τ, the canonical database $\Delta(\Phi)$ is the τ-structure defined on the variables of Φ such that $(x_1, \ldots, x_n) \in R^\Delta$ iff $R(x_1, \ldots, x_n)$ is a conjunct in Φ.

Example.

$\Phi := \exists x_1, x_2, x_3.\ E(x_1, x_2) \wedge E(x_2, x_3) \wedge E(x_3, x_1)$

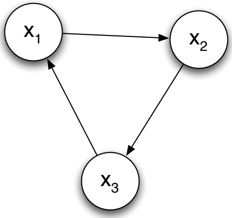
Canonical Database

Definition

For each primitive positive sentence Φ over signature τ, the canonical database $\Delta(\Phi)$ is the τ-structure defined on the variables of Φ such that $(x_1, \ldots, x_n) \in R^\Delta$ iff $R(x_1, \ldots, x_n)$ is a conjunct in Φ.

Example.

$\Phi := \exists x_1, x_2, x_3.\ E(x_1, x_2) \wedge E(x_2, x_3) \wedge E(x_3, x_1)$



Logic vs. Homomorphisms

Lemma

Let Γ be a τ-structure. For any primitive positive sentence Φ the following are equivalent.

- *Γ satisfies Φ.*
- *There is a homomorphism from $\Delta(\Phi)$ to Γ.*

Logic vs. Homomorphisms

Lemma

Let Γ be a τ-structure. For any primitive positive sentence Φ the following are equivalent.

- *Γ satisfies Φ.*
- *There is a homomorphism from $\Delta(\Phi)$ to Γ.*

Proof. trivial.

Computational Complexity of CSPs

Computational Complexity of CSPs

Basic observations:

- If Γ is finite, then $\mathrm{CSP}(\Gamma)$ is in NP.

Computational Complexity of CSPs

Basic observations:

- If Γ is finite, then $\mathrm{CSP}(\Gamma)$ is in NP.
- $\mathrm{CSP}(\Gamma)$ might be in P: e.g. $\mathrm{CSP}(\mathbb{Q};<)$.

Computational Complexity of CSPs

Basic observations:

- If Γ is finite, then CSP(Γ) is in NP.
- CSP(Γ) might be in P: e.g. CSP$(\mathbb{Q}; <)$.
- CSP(Γ) might be NP-complete: e.g. 1-in-3-3SAT.

Computational Complexity of CSPs

Basic observations:

- If Γ is finite, then CSP(Γ) is in NP.
- CSP(Γ) might be in P: e.g. CSP$(\mathbb{Q}; <)$.
- CSP(Γ) might be NP-complete: e.g. 1-in-3-3SAT.
- CSP(Γ) might be undecidable: CSP$(\mathbb{Z}; \{(x, y, z) \in \mathbb{Z}^3 \mid x + y = z\}, \{(x, y, z) \in \mathbb{Z}^3 \mid x * y = z\}, \{1\})$

Computational Complexity of CSPs

Basic observations:

- If Γ is finite, then CSP(Γ) is in NP.
- CSP(Γ) might be in P: e.g. CSP$(\mathbb{Q}; <)$.
- CSP(Γ) might be NP-complete: e.g. 1-in-3-3SAT.
- CSP(Γ) might be undecidable: CSP$(\mathbb{Z}; \{(x, y, z) \in \mathbb{Z}^3 \mid x + y = z\}, \{(x, y, z) \in \mathbb{Z}^3 \mid x * y = z\}, \{1\})$

is polynomial-time equivalent to the problem of deciding whether a given polynomial equation has an integer solution (solving *diophantine* equations; ‘Hilberts 10th problem’). This problem was shown to be undecidable by Matiyasevich in 1970.

CSPs of All Complexities

Theorem

For every $L \subseteq \{a, b\}^*$ there is a relational structure Γ such that L is polynomial-time equivalent to $\mathrm{CSP}(\Gamma)$.

CSPs of All Complexities

Theorem

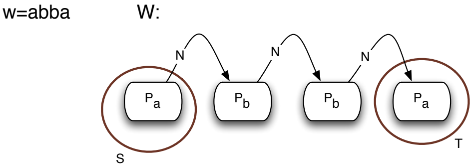
For every $L \subseteq \{a, b\}^*$ there is a relational structure Γ such that L is polynomial-time equivalent to CSP(Γ).

Reminder: Turing reduction. Write $L_1 \leq_t^p L_2$ if there is a deterministic polynomial-time Turing machine that decides L_1 with an oracle for L_2.

L_1 and L_2 are polynomial-time (Turing) equivalent if $L_1 \leq_t^p L_2$ and $L_2 \leq_t^p L_1$.

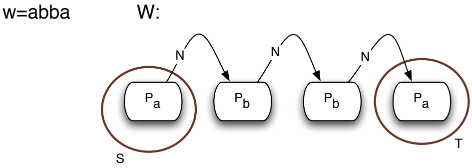
Proof

Proof idea. Encode words w from $\{a, b\}^*$ by structures W with signature $\{N, P_a, P_b, S, T\}$ as follows.



Proof

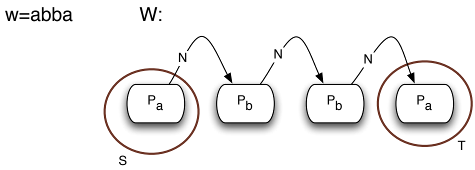
Proof idea. Encode words w from $\{a, b\}^*$ by structures W with signature $\{N, P_a, P_b, S, T\}$ as follows.



Let $\mathcal{X}$ be the set of all τ-structures encoding words as before, but
with an unlabeled element, or S is empty, or T is empty.
Let Γ be the disjoint union over all structures in $\{W \mid w \in L\} \cup \mathcal{X}$.

Proof

Proof idea. Encode words w from $\{a, b\}^*$ by structures W with signature $\{N, P_a, P_b, S, T\}$ as follows.



Let $\mathcal{X}$ be the set of all τ-structures encoding words as before, but
with an unlabeled element, or S is empty, or T is empty.
Let Γ be the disjoint union over all structures in $\{W \mid w \in L\} \cup \mathcal{X}$.

Claim: L is polynomial-time Turing equivalent to $\mathrm{CSP}(\Gamma)$.

First Reduction

Reduction from CSP(Γ) to L.

First Reduction

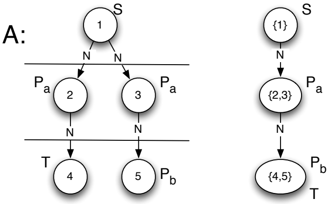
Reduction from CSP(Γ) to L.

Suppose A instance of CSP(Γ) (wlog A is connected wrt N)

First Reduction

Reduction from CSP(Γ) to L.

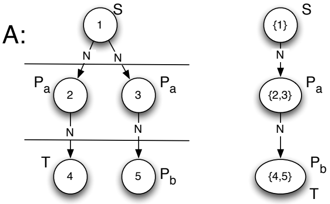
Suppose A instance of CSP(Γ) (wlog A is connected wrt N)



First Reduction

Reduction from CSP(Γ) to L.

Suppose A instance of CSP(Γ) (wlog A is connected wrt N)

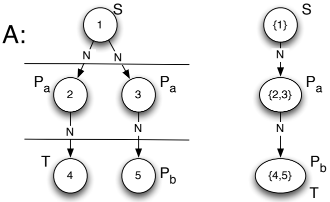


reject if the N-reduct is not homomorphic to a path, if a vertex from P_a is contracted with a vertex from P_b, a vertex in S has predecessor, or vertex in T has successor.

First Reduction

Reduction from CSP(Γ) to L.

Suppose A instance of CSP(Γ) (wlog A is connected wrt N)



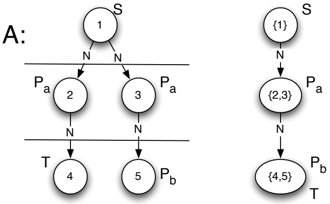
reject if the N-reduct is not homomorphic to a path, if a vertex from P_a is contracted with a vertex from P_b, a vertex in S has predecessor, or vertex in T has successor.

accept if in the resulting graph there is a vertex neither in P_a nor in P_b, or S is empty, or T is empty.

First Reduction

Reduction from CSP(Γ) to L.

Suppose A instance of CSP(Γ) (wlog A is connected wrt N)



reject if the N-reduct is not homomorphic to a path, if a vertex from P_a is contracted with a vertex from P_b, a vertex in S has predecessor, or vertex in T has successor.

accept if in the resulting graph there is a vertex neither in P_a nor in P_b, or S is empty, or T is empty.

otherwise have the word-structure of a word $w \in \{a, b\}^*$, and accept iff $w \in L$.

Second Reduction

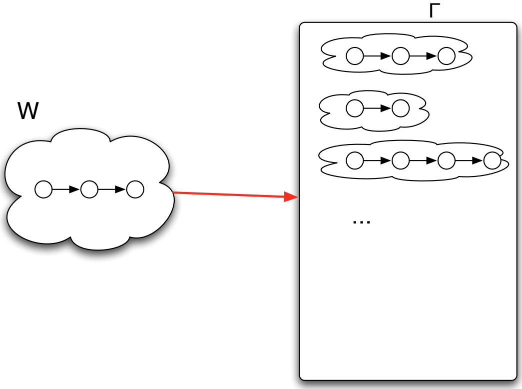
Reduction from L to CSP(Γ).

Second Reduction

Reduction from L to CSP(Γ).
Given a word w, **accept** if and only if
the word-structure for w homomorphically maps to Γ.

Second Reduction

Reduction from L to CSP(Γ).
Given a word w, **accept** if and only if
the word-structure for w homomorphically maps to Γ.



Tractability

One of the main questions in this course:

Which CSPs are tractable (=can be solved efficiently)?

Tractability

One of the main questions in this course:

Which CSPs are tractable (=can be solved efficiently)?

Often: tractable = ‘can be solved in deterministic polynomial time’ (P)

Tractability

One of the main questions in this course:

Which CSPs are tractable (=can be solved efficiently)?

Often: tractable = ‘can be solved in deterministic polynomial time’ (P)

Criticism:

- Is worst-case complexity really the right concept (rather than e.g. ‘average-case complexity’)?
- Is a $O(n^{100})$ algorithm really better than an $O(1.01^n)$ algorithm?

Why Polynomial Time?

P is still a well-accepted mathematical model of tractability:

Why Polynomial Time?

P is still a well-accepted mathematical model of tractability:

- If guaranteed bounded running time is essential for the application, there is no way around worst-case complexity.

Why Polynomial Time?

P is still a well-accepted mathematical model of tractability:

- If guaranteed bounded running time is essential for the application, there is no way around worst-case complexity.
- ‘practical’ and ‘theoretical’ complexity often match.

Why Polynomial Time?

P is still a well-accepted mathematical model of tractability:

- If guaranteed bounded running time is essential for the application, there is no way around worst-case complexity.
- ‘practical’ and ‘theoretical’ complexity often match.
- the fastest algorithms for relevant problems in P usually have a running time in $O(n^3)$, but not $O(n^{10})$.

Why Polynomial Time?

P is still a well-accepted mathematical model of tractability:

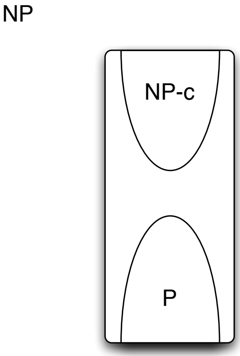
- If guaranteed bounded running time is essential for the application, there is no way around worst-case complexity.
- ‘practical’ and ‘theoretical’ complexity often match.
- the fastest algorithms for relevant problems in P usually have a running time in $O(n^3)$, but not $O(n^{10})$.
- P is robust: it is largely independent from the machine model

Why Polynomial Time?

P is still a well-accepted mathematical model of tractability:

- If guaranteed bounded running time is essential for the application, there is no way around worst-case complexity.
- ‘practical’ and ‘theoretical’ complexity often match.
- the fastest algorithms for relevant problems in P usually have a running time in $O(n^3)$, but not $O(n^{10})$.
- P is robust: it is largely independent from the machine model
- ‘Classical’ complexity theory is mathematically rich, deep, and beautiful

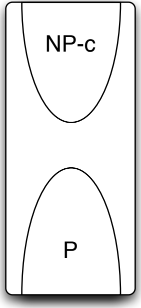
Computational Complexity



Computational Complexity

NP: class of computational problems decidable in non-deterministic polynomial time.

NP

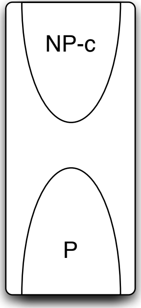


Computational Complexity

NP: class of computational problems decidable in non-deterministic polynomial time.

P: class of computational problems decidable in polynomial time.

NP

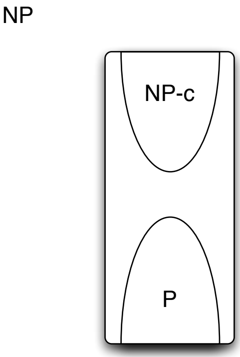


Computational Complexity

NP: class of computational problems decidable in non-deterministic polynomial time.

P: class of computational problems decidable in polynomial time.

NP-c: class of problems L such that every problem in NP can be reduced in polynomial time to L.



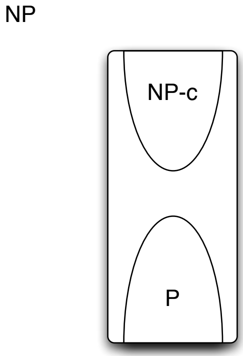
Computational Complexity

NP: class of computational problems decidable in non-deterministic polynomial time.

P: class of computational problems decidable in polynomial time.

NP-c: class of problems L such that every problem in NP can be reduced in polynomial time to L.

Ladner 1975: Unless P=NP, there are NP-intermediate problems: problems in NP that are neither in P nor NP-c.



The Quest for Tractable CSPs

For which Γ is CSP(Γ) in P?

The Quest for Tractable CSPs

For which Γ is CSP(Γ) in P?

Important open problem:

Conjecture (Feder, Vardi '93)

For finite relational structures Γ, CSP(Γ) is either in P or NP-hard.

The Quest for Tractable CSPs

For which Γ is CSP(Γ) in P?

Important open problem:

Conjecture (Feder, Vardi '93)

For finite relational structures Γ, CSP(Γ) is either in P or NP-hard.

- Remarkable: no NP-intermediate finite domain CSP

The Quest for Tractable CSPs

For which Γ is CSP(Γ) in P?

Important open problem:

Conjecture (Feder, Vardi '93)

For finite relational structures Γ, CSP(Γ) is either in P or NP-hard.

- Remarkable: no NP-intermediate finite domain CSP
- As we have seen: false when Γ might have infinite domain.

The Quest for Tractable CSPs

For which Γ is CSP(Γ) in P?

Important open problem:

Conjecture (Feder, Vardi ’93)

For finite relational structures Γ, CSP(Γ) is either in P or NP-hard.

- Remarkable: no NP-intermediate finite domain CSP
- As we have seen: false when Γ might have infinite domain.

Theorem (Feder, Vardi ’93)

For every finite Γ, there is a directed graph H such that CSP(Γ) and CSP(H) are polynomial-time equivalent.

The algebraic approach

Primitive positive (pp) definability

Fix a domain D.

All functions, relations, structures will be on D.

Primitive positive (pp) definability

Fix a domain D.

All functions, relations, structures will be on D.

For structures Γ and Δ on D, set $\Gamma \leq_{pp} \Delta$ iff every relation of Γ has a pp-definition from Δ.

Primitive positive (pp) definability

Fix a domain D.

All functions, relations, structures will be on D.

For structures Γ and Δ on D, set $\Gamma \leq_{pp} \Delta$ iff every relation of Γ has a pp-definition from Δ.

Fundamental observation.
If $\Gamma \leq_{pp} \Delta$, then CSP(Γ) has a polynomial-time reduction to CSP(Δ).

Primitive positive (pp) definability

Fix a domain D.

All functions, relations, structures will be on D.

For structures Γ and Δ on D, set $\Gamma \leq_{pp} \Delta$ iff
every relation of Γ has a pp-definition from Δ.

Fundamental observation.
If $\Gamma \leq_{pp} \Delta$, then CSP(Γ) has a polynomial-time reduction to CSP(Δ).

In particular: If $\Gamma \leq_{pp} \Delta$ and $\Delta \leq_{pp} \Gamma$,
then CSP(Γ) are polynomial-time equivalent.

Primitive positive (pp) definability

Fix a domain D.

All functions, relations, structures will be on D.

For structures Γ and Δ on D, set $\Gamma \leq_{pp} \Delta$ iff every relation of Γ has a pp-definition from Δ.

Fundamental observation.
If $\Gamma \leq_{pp} \Delta$, then CSP(Γ) has a polynomial-time reduction to CSP(Δ).

In particular: If $\Gamma \leq_{pp} \Delta$ and $\Delta \leq_{pp} \Gamma$, then CSP(Γ) are polynomial-time equivalent.

We therefore identify such structures and call them *pp-interdefinable* or *pp-equivalent*.

Polymorphisms

Polymorphisms

A function $f : D^n \to D$ *preserves* a relation R on D iff for all $r_1, \ldots, r_n \in R$ we have $f(r_1, \ldots, r_n) \in R$.

$f(r_1, \ldots, r_n)$ is calculated componentwise.

Polymorphisms

A function $f : D^n \to D$ *preserves* a relation R on D iff for all $r_1, \ldots, r_n \in R$ we have $f(r_1, \ldots, r_n) \in R$.

$f(r_1, \ldots, r_n)$ is calculated componentwise.

A function $f : D^n \to D$ is a *polymorphism* of Γ iff it preserves all relations of Γ.

Generalization of endomorphism, automorphism.

Polymorphisms

A function $f : D^n \to D$ *preserves* a relation R on D iff for all $r_1, \ldots, r_n \in R$ we have $f(r_1, \ldots, r_n) \in R$.

$f(r_1, \ldots, r_n)$ is calculated componentwise.

A function $f : D^n \to D$ is a *polymorphism* of Γ iff it preserves all relations of Γ.

Generalization of endomorphism, automorphism.

We write $\mathrm{Pol}(\Gamma)$ for the set of polymorphisms of Γ.

"Polymorphism clone of Γ".

Polymorphisms

A function $f : D^n \to D$ *preserves* a relation R on D iff for all $r_1, \ldots, r_n \in R$ we have $f(r_1, \ldots, r_n) \in R$.

$f(r_1, \ldots, r_n)$ is calculated componentwise.

A function $f : D^n \to D$ is a *polymorphism* of Γ iff it preserves all relations of Γ.

Generalization of endomorphism, automorphism.

We write $\mathrm{Pol}(\Gamma)$ for the set of polymorphisms of Γ.

"Polymorphism clone of Γ*"*.

A *clone* is a set of finitary operations on D which

- contains the projections and
- is closed under composition.

Invariant relations

Invariant relations

A relation R is *invariant* under a function f iff f preserves R.

Invariant relations

A relation R is *invariant* under a function f iff
f preserves R.

We write $\mathrm{Inv}(F)$ for the set of invariant relations of a set of functions F.

Invariant relations

A relation R is *invariant* under a function f iff
f preserves R.

We write $\mathrm{Inv}(F)$ for the set of invariant relations of a set of functions F.

- More relations in $\Gamma \rightarrow$ less functions in $\mathrm{Pol}(\Gamma)$.
- More functions in $F \rightarrow$ less relations in $\mathrm{Inv}(F)$.

Invariant relations

A relation R is *invariant* under a function f iff
f preserves R.

We write $\mathrm{Inv}(F)$ for the set of invariant relations of a set of functions F.

- More relations in $\Gamma \rightarrow$ less functions in $\mathrm{Pol}(\Gamma)$.
- More functions in $F \rightarrow$ less relations in $\mathrm{Inv}(F)$.

The operators Pol and Inv define a *Galois connection*, i.e.,

- Pol and Inv are antitone, and
- Pol Inv and Inv Pol are closure operators.

Polymorphisms and pp-definability

Polymorphisms and pp-definability

Let $\langle\Gamma\rangle_{pp}$ be the expansion of Γ by all pp definable relations.

Polymorphisms and pp-definability

Let $\langle\Gamma\rangle_{pp}$ be the expansion of Γ by all pp definable relations.

Theorem

Let Γ be finite or ω-categorical.
Then $\langle\Gamma\rangle_{pp} = \mathrm{Inv}\,\mathrm{Pol}(\Gamma)$.

Polymorphisms and pp-definability

Let $\langle \Gamma \rangle_{pp}$ be the expansion of Γ by all pp definable relations.

Theorem

Let Γ be finite or ω-categorical.
Then $\langle \Gamma \rangle_{pp} = \operatorname{Inv} \operatorname{Pol}(\Gamma)$.

Therefore, if Γ and Δ have the same polymorphisms, then their CSPs are polynomial-time equivalent.

Polymorphisms and pp-definability

Let $\langle\Gamma\rangle_{pp}$ be the expansion of Γ by all pp definable relations.

Theorem

Let Γ be finite or ω-categorical.
Then $\langle\Gamma\rangle_{pp} = \mathrm{Inv}\,\mathrm{Pol}(\Gamma)$.

Therefore, if Γ and Δ have the same polymorphisms, then their CSPs are polynomial-time equivalent.

Can define complexity of sets of functions F (algebras) on D to be the compexity of $\mathrm{Inv}(F)$.

Polymorphisms and pp-definability

Let $\langle\Gamma\rangle_{pp}$ be the expansion of Γ by all pp definable relations.

Theorem

Let Γ be finite or ω-categorical.
Then $\langle\Gamma\rangle_{pp} = \operatorname{Inv}\operatorname{Pol}(\Gamma)$.

Therefore, if Γ and Δ have the same polymorphisms, then their CSPs are polynomial-time equivalent.

Can define complexity of sets of functions F (algebras) on D to be the compexity of $\operatorname{Inv}(F)$.

Problem of infinite signature.

The polymorphism strategy

The polymorphism strategy

Larger structures → harder CSP

$\Gamma \leq_{pp} \Delta \quad \rightarrow \quad \mathrm{CSP}(\Gamma) \leq_t^p \mathrm{CSP}(\Delta)$

The polymorphism strategy

Larger structures → harder CSP

$\Gamma \leq_{pp} \Delta \quad \rightarrow \quad \mathrm{CSP}(\Gamma) \leq_t^p \mathrm{CSP}(\Delta)$

Larger clones → easier CSP

$\mathrm{Pol}(\Gamma) \subseteq \mathrm{Pol}(\Delta) \quad \rightarrow \quad \mathrm{CSP}(\Delta) \leq_t^p \mathrm{CSP}(\Gamma)$

The polymorphism strategy

Larger structures → harder CSP

$\Gamma \leq_{pp} \Delta \quad \rightarrow \quad \mathrm{CSP}(\Gamma) \leq_t^p \mathrm{CSP}(\Delta)$

Larger clones → easier CSP

$\mathrm{Pol}(\Gamma) \subseteq \mathrm{Pol}(\Delta) \quad \rightarrow \quad \mathrm{CSP}(\Delta) \leq_t^p \mathrm{CSP}(\Gamma)$

Strategy:

(i) Prove hardness for certain relations

(ii) Prove tractability for certain functions

(iii) Hope that this is exhaustive

The polymorphism strategy

Larger structures → harder CSP

$\Gamma \leq_{pp} \Delta \quad \rightarrow \quad \mathrm{CSP}(\Gamma) \leq_t^p \mathrm{CSP}(\Delta)$

Larger clones → easier CSP

$\mathrm{Pol}(\Gamma) \subseteq \mathrm{Pol}(\Delta) \quad \rightarrow \quad \mathrm{CSP}(\Delta) \leq_t^p \mathrm{CSP}(\Gamma)$

Strategy:

(i) Prove hardness for certain relations
(ii) Prove tractability for certain functions
(iii) Hope that this is exhaustive

Structures which do not pp-define hard relations have polymorphisms violating them.

Some polymorphisms implying tractability

Theorem

On a finite domain, the following polymorphisms imply tractability of the CSP.

Some polymorphisms implying tractability

Theorem

On a finite domain, the following polymorphisms imply tractability of the CSP.

- $\vee$-semilattice

Some polymorphisms implying tractability

Theorem

On a finite domain, the following polymorphisms imply tractability of the CSP.

- $\vee$-semilattice
- Majority

Some polymorphisms implying tractability

Theorem

On a finite domain, the following polymorphisms imply tractability of the CSP.

- $\vee$-semilattice
- Majority
- Minority

Some polymorphisms implying tractability

Theorem

On a finite domain, the following polymorphisms imply tractability of the CSP.

- $\vee$-semilattice
- Majority
- Minority
- Mal’tsev

Some polymorphisms implying tractability

Theorem

On a finite domain, the following polymorphisms imply tractability of the CSP.

- $\vee$-semilattice
- Majority
- Minority
- Mal'tsev
- Constant

Invariant relations and clones

Invariant relations and clones

For a set F of functions on D, write $\langle F \rangle$ for the smallest clone containing F.

“The clone generated by F”.

Invariant relations and clones

For a set F of functions on D, write $\langle F\rangle$ for the smallest clone containing F.

“The clone generated by F”.

$\langle F\rangle$ is obtained by building all terms over F.

Invariant relations and clones

For a set F of functions on D, write $\langle F \rangle$ for the smallest clone containing F.

"The clone generated by F".

$\langle F \rangle$ is obtained by building all terms over F.

Theorem

For finite D we have $\langle F \rangle = \mathrm{Pol}\,\mathrm{Inv}(F)$.

Invariant relations and clones

For a set F of functions on D, write $\langle F \rangle$ for the smallest clone containing F.

"The clone generated by F".

$\langle F \rangle$ is obtained by building all terms over F.

Theorem

For finite D we have $\langle F \rangle = \mathrm{Pol}\,\mathrm{Inv}(F)$.

Therefore, if two sets F, G of functions generate the same clone, then they have the same complexity.

Invariant relations and clones

For a set F of functions on D, write $\langle F \rangle$ for the smallest clone containing F.

“The clone generated by F”.

$\langle F \rangle$ is obtained by building all terms over F.

Theorem

For finite D we have $\langle F \rangle = \mathrm{Pol}\,\mathrm{Inv}(F)$.

Therefore, if two sets F, G of functions generate the same clone, then they have the same complexity.

Sample application: If Γ has a polymorphism which generates a tractable polymorphism, then CSP(Γ) is tractable.

Clones on infinite sets

What about infinite domains?

Clones on infinite sets

What about infinite domains?

For a set F of functions on D, write $\langle F\rangle_{loc}$ for the *topological closure* of $\langle F\rangle$ in the natural topology on the space of all operations on D.

Clones on infinite sets

What about infinite domains?

For a set F of functions on D, write $\langle F \rangle_{loc}$ for the *topological closure* of $\langle F \rangle$ in the natural topology on the space of all operations on D.

$\langle F \rangle_{loc}$ is called the *local clone* generated by F.

Clones on infinite sets

What about infinite domains?

For a set F of functions on D, write $\langle F \rangle_{loc}$ for the *topological closure* of $\langle F \rangle$ in the natural topology on the space of all operations on D.

$\langle F \rangle_{loc}$ is called the *local clone* generated by F.

A function $f : D^n \to D$ is in $\langle F \rangle_{loc}$ iff for all finite subsets S of D^n there is a function in $\langle F \rangle$ which agrees with f on S.

“f can be interpolated by functions from $\langle F \rangle$ on finite sets.”

Clones on infinite sets

What about infinite domains?

For a set F of functions on D, write $\langle F\rangle_{loc}$ for the *topological closure* of $\langle F\rangle$ in the natural topology on the space of all operations on D.

$\langle F\rangle_{loc}$ is called the *local clone* generated by F.

A function $f : D^n \to D$ is in $\langle F\rangle_{loc}$ iff for all finite subsets S of D^n there is a function in $\langle F\rangle$ which agrees with f on S.

“f can be interpolated by functions from $\langle F\rangle$ on finite sets.”

Theorem

For any D we have $\langle F\rangle_{loc} = \mathrm{Pol}\,\mathrm{Inv}(F)$.

Term conditions

Many properties of an algebra $(D; F)$ only depend on the clone $\langle F \rangle$ of the algebra.

Term conditions

Many properties of an algebra $(D; F)$ only depend on the clone $\langle F \rangle$ of the algebra.

Classical examples: Subalgebras, congruence relations.

Term conditions

Many properties of an algebra $(D; F)$ only depend on the clone $\langle F \rangle$ of the algebra.

Classical examples: Subalgebras, congruence relations.

New example: Complexity of the CSP of the algebra.

Term conditions

Many properties of an algebra $(D; F)$ only depend on the clone $\langle F \rangle$ of the algebra.

Classical examples: Subalgebras, congruence relations.

New example: Complexity of the CSP of the algebra.

Sample universal algebra theorem:

The congruences of an algebra permute iff the algebra has a term $t(x, y, z)$ which satisfies $t(x, x, y) = t(y, x, x) = y$.

Term conditions

Many properties of an algebra $(D; F)$ only depend on the clone $\langle F \rangle$ of the algebra.

Classical examples: Subalgebras, congruence relations.

New example: Complexity of the CSP of the algebra.

Sample universal algebra theorem:

The congruences of an algebra permute iff the algebra has a term $t(x, y, z)$ which satisfies
$t(x, x, y) = t(y, x, x) = y$.

Many properties depend only on equations satisfied by terms in the clone.

Term conditions

Many properties of an algebra $(D; F)$ only depend on the clone $\langle F \rangle$ of the algebra.

Classical examples: Subalgebras, congruence relations.

New example: Complexity of the CSP of the algebra.

Sample universal algebra theorem:

The congruences of an algebra permute iff the algebra has a term $t(x, y, z)$ which satisfies $t(x, x, y) = t(y, x, x) = y$.

Many properties depend only on equations satisfied by terms in the clone.

Also holds for the complexity of the CSP.

Some polymorphisms implying tractability

Theorem

On a finite domain, the following polymorphisms imply tractability of the CSP.

Some polymorphisms implying tractability

Theorem

On a finite domain, the following polymorphisms imply tractability of the CSP.

- $\vee$-semilattice

Some polymorphisms implying tractability

Theorem

On a finite domain, the following polymorphisms imply tractability of the CSP.

- $\bigvee$-semilattice
- Majority

Some polymorphisms implying tractability

Theorem

On a finite domain, the following polymorphisms imply tractability of the CSP.

- $\vee$-semilattice
- Majority
- Minority

Some polymorphisms implying tractability

Theorem

On a finite domain, the following polymorphisms imply tractability of the CSP.

- $\bigvee$-semilattice
- Majority
- Minority
- Mal'tsev

Some polymorphisms implying tractability

Theorem

On a finite domain, the following polymorphisms imply tractability of the CSP.

- $\vee$-semilattice
- Majority
- Minority
- Mal'tsev
- Constant

The tractability conjecture

Dichotomy conjecture

All finite domain CSPs are either tractable or NP-complete.

The tractability conjecture

Dichotomy conjecture

All finite domain CSPs are either tractable or NP-complete.

Tractability conjecture

For all structures Γ on a finite domain which are a core,

- either there is a poymorphism $f(x_1, x_2, x_3, x_4)$ satisfying $f(y, y, x, x) = f(x, x, x, y) = f(y, x, y, x)$, and CSP($\Gamma$) is tractable,
- or CSP(Γ) is NP-complete.

ω-categorical templates

Ryll-Nardzewski

The *type* of an tuple a of elements of a structure Γ is the set of first-order formulas satisfied by a in Γ.

Ryll-Nardzewski

The *type* of an tuple a of elements of a structure Γ is the set of first-order formulas satisfied by a in Γ.

Theorem (Ryll-Nardzewski)

The following are equivalent for a countable structure Γ.

- All countable models of the theory of Γ are isomorphic to Γ.
- Γ has finitely many types of n-tuples for every $n \geq 1$.

Ryll-Nardzewski

The *type* of an tuple a of elements of a structure Γ is the set of first-order formulas satisfied by a in Γ.

Theorem (Ryll-Nardzewski)

The following are equivalent for a countable structure Γ.

- All countable models of the theory of Γ are isomorphic to Γ.
- Γ has finitely many types of n-tuples for every $n \geq 1$.

The *orbit* of a tuple a in Γ is the set $\{\alpha(a) : \alpha \in \mathrm{Aut}(\Gamma)\}$.

Ryll-Nardzewski

The *type* of an tuple a of elements of a structure Γ is the set of first-order formulas satisfied by a in Γ.

Theorem (Ryll-Nardzewski)

The following are equivalent for a countable structure Γ.

- All countable models of the theory of Γ are isomorphic to Γ.
- Γ has finitely many types of n-tuples for every $n \geq 1$.

The *orbit* of a tuple a in Γ is the set $\{\alpha(a) : \alpha \in \mathrm{Aut}(\Gamma)\}$.

In ω-categorical structures,
orbits = maximal sets of tuples of the same type.

Ryll-Nardzewski

The *type* of an tuple a of elements of a structure Γ is the set of first-order formulas satisfied by a in Γ.

Theorem (Ryll-Nardzewski)

The following are equivalent for a countable structure Γ.

- All countable models of the theory of Γ are isomorphic to Γ.
- Γ has finitely many types of n-tuples for every $n \geq 1$.

The *orbit* of a tuple a in Γ is the set $\{\alpha(a) : \alpha \in \text{Aut}(\Gamma)\}$.

In ω-categorical structures,
orbits = maximal sets of tuples of the same type.

Thus, a relation R has a fo definition from Γ iff
it is preserved by all automorphisms of Γ.

Ryll-Nardzewski generalized

Theorem (Bodirsky and Nešetřil) '02

Let Γ be ω-categorical.

A relation *R* has a primitive positive (pp) definition from Γ iff it is preserved by all polymorphisms of Γ.

In other words, $\mathrm{Inv}\,\mathrm{Pol}(\Gamma) = \langle\Gamma\rangle_{pp}$.

Ryll-Nardzewski generalized

Theorem (Bodirsky and Nešetřil) ’02

Let Γ be ω-categorical.

A relation *R* has a primitive positive (pp) definition from Γ iff it is preserved by all polymorphisms of Γ.

In other words, $\mathrm{Inv}\,\mathrm{Pol}(\Gamma) = \langle \Gamma \rangle_{pp}$.

The corresponding first-order statement:

$\mathrm{Inv}\,\mathrm{Aut}(\Gamma) = \langle \Gamma \rangle_{fo}$

The Boolean satisfiability problem

The Boolean satisfiability problem

Let Ψ be a finite set of propositional formulas.

The Boolean satisfiability problem

Let Ψ be a finite set of propositional formulas.

Computational problem: Boolean-SAT(Ψ)

INPUT:

- A set W of propositional variables, and
- statements $\phi_1, \ldots, \phi_n$ about the variables in W, where each ϕ_i is taken from Ψ.

QUESTION: Is $\bigwedge_{1 \leq i \leq n} \phi_i$ satisfiable?

The Boolean satisfiability problem

Let Ψ be a finite set of propositional formulas.

Computational problem: Boolean-SAT(Ψ)

INPUT:

- A set W of propositional variables, and
- statements $\phi_1, \ldots, \phi_n$ about the variables in W, where each ϕ_i is taken from Ψ.

QUESTION: Is $\bigwedge_{1 \leq i \leq n} \phi_i$ satisfiable?

Computational complexity depends on Ψ. Always in NP.

The Boolean satisfiability problem

Let Ψ be a finite set of propositional formulas.

Computational problem: Boolean-SAT(Ψ)

INPUT:

- A set W of propositional variables, and
- statements $\phi_1, \ldots, \phi_n$ about the variables in W, where each ϕ_i is taken from Ψ.

QUESTION: Is $\bigwedge_{1 \leq i \leq n} \phi_i$ satisfiable?

Computational complexity depends on Ψ. Always in NP.

Question

For which Ψ is Boolean-SAT(Ψ) tractable?

Boolean formulas and Boolean structures

Boolean formulas and Boolean structures

For a Boolean formula $\psi(x_1, \ldots, x_n)$, define a relation

$$R_\psi := \{(a_1, \ldots, a_n) \in \{0,1\}^n : \psi(a_1, \ldots, a_n)\}.$$

Boolean formulas and Boolean structures

For a Boolean formula $\psi(x_1, \ldots, x_n)$, define a relation

$$R_\psi := \{(a_1, \ldots, a_n) \in \{0,1\}^n : \psi(a_1, \ldots, a_n)\}.$$

For a set Ψ of Boolean formulas, define a structure

$$\Gamma_\Psi := (\{0,1\}; (R_\psi : \psi \in \Psi)).$$

Boolean formulas and Boolean structures

For a Boolean formula $\psi(x_1, \ldots, x_n)$, define a relation

$$R_\psi := \{(a_1, \ldots, a_n) \in \{0,1\}^n : \psi(a_1, \ldots, a_n)\}.$$

For a set Ψ of Boolean formulas, define a structure

$$\Gamma_\Psi := (\{0,1\}; (R_\psi : \psi \in \Psi)).$$

Γ_Ψ is a Boolean structure.

Boolean-SAT as CSP

Boolean-SAT as CSP

An instance

- $W = \{w_1, \ldots, w_m\}$
- $\phi_1, \ldots, \phi_n$

of Boolean-SAT(Ψ) has a positive solution $\leftrightarrow$

Boolean-SAT as CSP

An instance

- $W = \{w_1, \ldots, w_m\}$
- $\phi_1, \ldots, \phi_n$

of Boolean-SAT(Ψ) has a positive solution $\leftrightarrow$

the sentence $\exists w_1, \ldots, w_m. \bigwedge_i \phi_i$ holds in Γ_Ψ.

Boolean-SAT as CSP

An instance

- $W = \{w_1, \ldots, w_m\}$
- $\phi_1, \ldots, \phi_n$

of Boolean-SAT(Ψ) has a positive solution $\leftrightarrow$

the sentence $\exists w_1, \ldots, w_m. \bigwedge_i \phi_i$ holds in Γ_Ψ.

The decision problem whether or not a given primitive positive sentence holds in Γ_Ψ is just CSP(Γ_Ψ).

Boolean-SAT as CSP

An instance

- $W = \{w_1, \ldots, w_m\}$
- $\phi_1, \ldots, \phi_n$

of Boolean-SAT(Ψ) has a positive solution $\leftrightarrow$

the sentence $\exists w_1, \ldots, w_m. \bigwedge_i \phi_i$ holds in Γ_Ψ.

The decision problem whether or not a given primitive positive sentence holds in Γ_Ψ is just CSP(Γ_Ψ).

So Boolean-SAT(Ψ) and CSP(Γ_Ψ) are one and the same problem.

The Graph Satisfiability Problem

The Graph Satisfiability Problem

Let E be a binary relation symbol.

(Imagine: edge relation of an undirected graph.)

Let Ψ be a finite set of quantifier-free $\{E\}$-formulas.

The Graph Satisfiability Problem

Let E be a binary relation symbol.

(Imagine: edge relation of an undirected graph.)

Let Ψ be a finite set of quantifier-free $\{E\}$-formulas.

Computational problem: Graph-SAT(Ψ)

INPUT:

- A set W of variables (vertices), and
- statements $\phi_1, \ldots, \phi_n$ about the elements of W, where each ϕ_i is taken from Ψ.

QUESTION: Is $\bigwedge_{1 \leq i \leq n} \phi_i$ satisfiable in a graph?

The Graph Satisfiability Problem

Let E be a binary relation symbol.

(Imagine: edge relation of an undirected graph.)

Let Ψ be a finite set of quantifier-free $\{E\}$-formulas.

Computational problem: Graph-SAT(Ψ)

INPUT:

- A set W of variables (vertices), and
- statements $\phi_1, \ldots, \phi_n$ about the elements of W, where each ϕ_i is taken from Ψ.

QUESTION: Is $\bigwedge_{1 \leq i \leq n} \phi_i$ satisfiable in a graph?

Computational complexity depends on Ψ. Always in NP.

The Graph Satisfiability Problem

Let E be a binary relation symbol.

(Imagine: edge relation of an undirected graph.)

Let Ψ be a finite set of quantifier-free $\{E\}$-formulas.

Computational problem: Graph-SAT(Ψ)

INPUT:

- A set W of variables (vertices), and
- statements $\phi_1, \ldots, \phi_n$ about the elements of W, where each ϕ_i is taken from Ψ.

QUESTION: Is $\bigwedge_{1 \leq i \leq n} \phi_i$ satisfiable in a graph?

Computational complexity depends on Ψ. Always in NP.

Question

For which Ψ is Graph-SAT(Ψ) tractable?

Graph-SAT: Examples

Graph-SAT: Examples

Example 1 Let Ψ_1 only contain

$$\begin{aligned}\psi_1(x,y,z) :=& (E(x,y) \wedge \neg E(y,z) \wedge \neg E(x,z)) \\ & \vee (\neg E(x,y) \wedge E(y,z) \wedge \neg E(x,z)) \\ & \vee (\neg E(x,y) \wedge \neg E(y,z) \wedge E(x,z)) \, .\end{aligned}$$

Graph-SAT: Examples

Example 1 Let Ψ_1 only contain

$$\begin{aligned}\psi_1(x,y,z) :=& (E(x,y) \wedge \neg E(y,z) \wedge \neg E(x,z)) \\ &\vee (\neg E(x,y) \wedge E(y,z) \wedge \neg E(x,z)) \\ &\vee (\neg E(x,y) \wedge \neg E(y,z) \wedge E(x,z))\ .\end{aligned}$$

Graph-SAT(Ψ_1) is NP-complete.

Graph-SAT: Examples

Example 1 Let Ψ_1 only contain

$$\begin{aligned}\psi_1(x,y,z) :=&(E(x,y) \wedge \neg E(y,z) \wedge \neg E(x,z))\\ &\vee (\neg E(x,y) \wedge E(y,z) \wedge \neg E(x,z))\\ &\vee (\neg E(x,y) \wedge \neg E(y,z) \wedge E(x,z)) \, .\end{aligned}$$

Graph-SAT(Ψ_1) is NP-complete.

Example 2 Let Ψ_2 only contain

$$\begin{aligned}\psi_2(x,y,z) :=&(E(x,y) \wedge \neg E(y,z) \wedge \neg E(x,z))\\ &\vee (\neg E(x,y) \wedge E(y,z) \wedge \neg E(x,z))\\ &\vee (\neg E(x,y) \wedge \neg E(y,z) \wedge E(x,z))\\ &\vee (E(x,y) \wedge E(y,z) \wedge E(x,z)) \, .\end{aligned}$$

Graph-SAT: Examples

Example 1 Let Ψ_1 only contain

$$\begin{aligned}\psi_1(x,y,z) :=& (E(x,y) \wedge \neg E(y,z) \wedge \neg E(x,z)) \\ &\vee (\neg E(x,y) \wedge E(y,z) \wedge \neg E(x,z)) \\ &\vee (\neg E(x,y) \wedge \neg E(y,z) \wedge E(x,z)) .\end{aligned}$$

Graph-SAT(Ψ_1) is NP-complete.

Example 2 Let Ψ_2 only contain

$$\begin{aligned}\psi_2(x,y,z) :=& (E(x,y) \wedge \neg E(y,z) \wedge \neg E(x,z)) \\ &\vee (\neg E(x,y) \wedge E(y,z) \wedge \neg E(x,z)) \\ &\vee (\neg E(x,y) \wedge \neg E(y,z) \wedge E(x,z)) \\ &\vee (E(x,y) \wedge E(y,z) \wedge E(x,z)) .\end{aligned}$$

Graph-SAT(Ψ_2) is in P.

Graph formulas and reducts of the random graph

Graph formulas and reducts of the random graph

Let $G = (V; E)$ denote the *random graph*, i.e., the unique countably infinite graph which

Graph formulas and reducts of the random graph

Let $G = (V; E)$ denote the *random graph*, i.e., the unique countably infinite graph which

- is (ultra-)homogeneous

Graph formulas and reducts of the random graph

Let $G = (V; E)$ denote the *random graph*, i.e., the unique countably infinite graph which

- is (ultra-)homogeneous
- contains all finite (even countable) graphs.

Graph formulas and reducts of the random graph

Let $G = (V; E)$ denote the *random graph*, i.e., the unique countably infinite graph which

- is (ultra-)homogeneous
- contains all finite (even countable) graphs.

For a graph formula $\psi(x_1, \ldots, x_n)$, define a relation

$$R_\psi := \{(a_1, \ldots, a_n) \in V^n : \psi(a_1, \ldots, a_n)\}.$$

Graph formulas and reducts of the random graph

Let $G = (V; E)$ denote the *random graph*, i.e., the unique countably infinite graph which

- is (ultra-)homogeneous
- contains all finite (even countable) graphs.

For a graph formula $\psi(x_1, \ldots, x_n)$, define a relation

$$R_\psi := \{(a_1, \ldots, a_n) \in V^n : \psi(a_1, \ldots, a_n)\}.$$

For a set Ψ of graph formulas, define a structure

$$\Gamma_\Psi := (V; (R_\psi : \psi \in \Psi)).$$

Graph formulas and reducts of the random graph

Let $G = (V; E)$ denote the *random graph*, i.e., the unique countably infinite graph which

- is (ultra-)homogeneous
- contains all finite (even countable) graphs.

For a graph formula $\psi(x_1, \ldots, x_n)$, define a relation

$$R_\psi := \{(a_1, \ldots, a_n) \in V^n : \psi(a_1, \ldots, a_n)\}.$$

For a set Ψ of graph formulas, define a structure

$$\Gamma_\Psi := (V; (R_\psi : \psi \in \Psi)).$$

Γ_Ψ is a *reduct of* the random graph, i.e., a structure with a first-order definition in G.

Graph-SAT as CSP

Graph-SAT as CSP

An instance

- $W = \{w_1, \ldots, w_m\}$
- $\phi_1, \ldots, \phi_n$

of Graph-SAT(Ψ) has a positive solution $\leftrightarrow$

Graph-SAT as CSP

An instance

- $W = \{w_1, \ldots, w_m\}$
- $\phi_1, \ldots, \phi_n$

of Graph-SAT(Ψ) has a positive solution $\leftrightarrow$

the sentence $\exists w_1, \ldots, w_m. \bigwedge_i \phi_i$ holds in Γ_Ψ.

Graph-SAT as CSP

An instance

- $W = \{w_1, \ldots, w_m\}$
- $\phi_1, \ldots, \phi_n$

of Graph-SAT(Ψ) has a positive solution $\leftrightarrow$

the sentence $\exists w_1, \ldots, w_m. \bigwedge_i \phi_i$ holds in Γ_Ψ.

So Graph-SAT(Ψ) and CSP(Γ_Ψ) are one and the same problem.

Graph-SAT as CSP

An instance

- $W = \{w_1, \ldots, w_m\}$
- $\phi_1, \ldots, \phi_n$

of Graph-SAT(Ψ) has a positive solution $\leftrightarrow$

the sentence $\exists w_1, \ldots, w_m. \bigwedge_i \phi_i$ holds in Γ_Ψ.

So Graph-SAT(Ψ) and CSP(Γ_Ψ) are one and the same problem.

Could have used any graph that contains all finite graphs.

Graph-SAT as CSP

An instance

- $W = \{w_1, \ldots, w_m\}$
- $\phi_1, \ldots, \phi_n$

of Graph-SAT(Ψ) has a positive solution $\leftrightarrow$

the sentence $\exists w_1, \ldots, w_m. \bigwedge_i \phi_i$ holds in Γ_Ψ.

So Graph-SAT(Ψ) and CSP(Γ_Ψ) are one and the same problem.

Could have used any graph that contains all finite graphs.

Classifying the complexity of all Graph-SAT problems is the same as classifying the complexity of CSPs of all reducts of the random graph.

Temporal constraints

Temporal constraints

Let $<$ be a binary relation symbol.

(Imagine: linear order relation.)

Let Ψ be a finite set of quantifier-free $\{<\}$-formulas.

Temporal constraints

Let $<$ be a binary relation symbol.

(Imagine: linear order relation.)

Let Ψ be a finite set of quantifier-free $\{<\}$-formulas.

Computational problem: Temp-SAT(Ψ)

INPUT:

- A set W of variables (vertices), and
- statements $\phi_1, \ldots, \phi_n$ about the elements of W, where each ϕ_i is taken from Ψ.

QUESTION: Is $\bigwedge_{1 \leq i \leq n} \phi_i$ satisfiable in a linear order?

Temporal constraints

Let $<$ be a binary relation symbol.

(Imagine: linear order relation.)

Let Ψ be a finite set of quantifier-free $\{<\}$-formulas.

Computational problem: Temp-SAT(Ψ)

INPUT:

- A set W of variables (vertices), and
- statements $\phi_1, \ldots, \phi_n$ about the elements of W, where each ϕ_i is taken from Ψ.

QUESTION: Is $\bigwedge_{1 \leq i \leq n} \phi_i$ satisfiable in a linear order?

Computational complexity depends on Ψ. Always in NP.

Temporal constraints

Let $<$ be a binary relation symbol.

(Imagine: linear order relation.)

Let Ψ be a finite set of quantifier-free $\{<\}$-formulas.

Computational problem: Temp-SAT(Ψ)

INPUT:

- A set *W* of variables (vertices), and
- statements $\phi_1, \ldots, \phi_n$ about the elements of *W*, where each ϕ_i is taken from Ψ.

QUESTION: Is $\bigwedge_{1 \leq i \leq n} \phi_i$ satisfiable in a linear order?

Computational complexity depends on Ψ. Always in NP.

Question

For which Ψ is Temp-SAT(Ψ) tractable?

Temporal formulas and reducts of $(\mathbb{Q}; <)$

Temporal formulas and reducts of $(\mathbb{Q};<)$

Let $(\mathbb{Q};<)$ denote the order of the rationals.

Temporal formulas and reducts of $(\mathbb{Q}; <)$

Let $(\mathbb{Q}; <)$ denote the order of the rationals.

Temporal formulas and reducts of $(\mathbb{Q}; <)$

Let $(\mathbb{Q}; <)$ denote the order of the rationals.

Temporal formulas and reducts of $(\mathbb{Q};<)$

Let $(\mathbb{Q};<)$ denote the order of the rationals.

For a $\{<\}$-formula $\psi(x_1,\ldots,x_n)$, define a relation

$$R_\psi := \{(a_1,\ldots,a_n) \in V^n : \psi(a_1,\ldots,a_n)\}.$$

Temporal formulas and reducts of $(\mathbb{Q};<)$

Let $(\mathbb{Q};<)$ denote the order of the rationals.

For a $\{<\}$-formula $\psi(x_1,\dots,x_n)$, define a relation

$$R_\psi := \{(a_1,\dots,a_n) \in V^n : \psi(a_1,\dots,a_n)\}.$$

For a set Ψ of $\{<\}$-formulas, define a structure

$$\Gamma_\Psi := (V; (R_\psi : \psi \in \Psi)).$$

Temporal formulas and reducts of $(\mathbb{Q};<)$

Let $(\mathbb{Q};<)$ denote the order of the rationals.

For a $\{<\}$-formula $\psi(x_1,\ldots,x_n)$, define a relation

$$R_\psi := \{(a_1,\ldots,a_n) \in V^n : \psi(a_1,\ldots,a_n)\}.$$

For a set Ψ of $\{<\}$-formulas, define a structure

$$\Gamma_\Psi := (V;(R_\psi : \psi \in \Psi)).$$

Γ_Ψ is a reduct the dense linear order.

Temporal formulas and reducts of $(\mathbb{Q};<)$

Let $(\mathbb{Q};<)$ denote the order of the rationals.

For a $\{<\}$-formula $\psi(x_1,\dots,x_n)$, define a relation

$$R_\psi := \{(a_1,\dots,a_n) \in V^n : \psi(a_1,\dots,a_n)\}.$$

For a set Ψ of $\{<\}$-formulas, define a structure

$$\Gamma_\Psi := (V;(R_\psi : \psi \in \Psi)).$$

Γ_Ψ is a reduct the dense linear order.

Temp-SAT(Ψ) and CSP(Γ_Ψ) are one and the same problem.

Temporal formulas and reducts of $(\mathbb{Q};<)$

Let $(\mathbb{Q};<)$ denote the order of the rationals.

For a $\{<\}$-formula $\psi(x_1,\dots,x_n)$, define a relation

$$R_\psi := \{(a_1,\dots,a_n) \in V^n : \psi(a_1,\dots,a_n)\}.$$

For a set Ψ of $\{<\}$-formulas, define a structure

$$\Gamma_\Psi := (V;(R_\psi : \psi \in \Psi)).$$

Γ_Ψ is a reduct the dense linear order.

Temp-SAT(Ψ) and CSP(Γ_Ψ) are one and the same problem.

Could have used any linear order that contains all finite linear orders.

Three classification theorems

All problems Boolean-SAT(Ψ), Graph-SAT(Ψ), and Temp-SAT(Ψ) are either in P or NP-complete.

Given Ψ, we can decide in which class the problem falls.

Boolean-SAT(Ψ): Schaefer (1978).

Temp-SAT(Ψ): Bodirsky and Kara (2007).

Graph-SAT(Ψ): Bodirsky and Pinsker (2010).

Remark: Complexity of CSPs for 3-element domains classified by Bulatov in ’03.

Ramsey structures

Reducts of homogeneous structures

Let Γ be a countable relational structure in a finite language

Reducts of homogeneous structures

Let Γ be a countable relational structure in a finite language
which is *homogeneous*, i.e.,

For all $A, B \subseteq \Gamma$ finite, for all isomorphisms $i : A \to B$ there exists $\alpha \in \text{Aut}(\Gamma)$ extending i.

Reducts of homogeneous structures

Let Γ be a countable relational structure in a finite language
which is *homogeneous*, i.e.,

For all $A, B \subseteq \Gamma$ finite, for all isomorphisms $i : A \to B$ there exists $\alpha \in \mathrm{Aut}(\Gamma)$ extending i.

Γ is the Fraïssé limit of its *age*, i.e., its class of finite induced substructures.

Reducts of homogeneous structures

Let Γ be a countable relational structure in a finite language
which is *homogeneous*, i.e.,

For all $A, B \subseteq \Gamma$ finite, for all isomorphisms $i : A \to B$ there exists $\alpha \in \mathrm{Aut}(\Gamma)$ extending i.

Γ is the Fraïssé limit of its *age*, i.e., its class of finite induced substructures.

Definition

A *reduct* of Γ is a structure with a first-order (f.o.) definition in Γ.

Reducts of homogeneous structures

Let Γ be a countable relational structure in a finite language
which is *homogeneous*, i.e.,

For all $A, B \subseteq \Gamma$ finite, for all isomorphisms $i : A \to B$ there exists $\alpha \in \mathrm{Aut}(\Gamma)$ extending i.

Γ is the Fraïssé limit of its *age*, i.e., its class of finite induced substructures.

Definition

A *reduct* of Γ is a structure with a first-order (f.o.) definition in Γ.

Problem

Classify the reducts of Γ.

Possible classifications

Consider two reducts Δ, Δ' of Γ *equivalent* iff Δ has a fo definition from Δ' and vice-versa.

Possible classifications

Consider two reducts Δ, Δ' of Γ *equivalent* iff Δ has a fo definition from Δ' and vice-versa.

We say that Δ and Δ' are *first-order interdefinable*.

Possible classifications

Consider two reducts Δ, Δ' of Γ *equivalent* iff Δ has a fo definition from Δ' and vice-versa.

We say that Δ and Δ' are *first-order interdefinable*.

"Δ has a fo definition from Δ'" is a *quasiorder* on relational structures over the same domain.

Possible classifications

Consider two reducts Δ, Δ' of Γ *equivalent* iff Δ has a fo definition from Δ' and vice-versa.

We say that Δ and Δ' are *first-order interdefinable*.

"Δ has a fo definition from Δ'" is a *quasiorder* on relational structures over the same domain.

This quasiorder, factored by f.o.-interdefinability, becomes a *complete lattice*.

Possible classifications

Consider two reducts Δ, Δ' of Γ *equivalent* iff Δ has a fo definition from Δ' and vice-versa.

We say that Δ and Δ' are *first-order interdefinable*.

"Δ has a fo definition from Δ'" is a *quasiorder* on relational structures over the same domain.

This quasiorder, factored by f.o.-interdefinability, becomes a *complete lattice*.

Finer classifications of the reducts of Γ, e.g. up to

Possible classifications

Consider two reducts Δ, Δ' of Γ *equivalent* iff Δ has a fo definition from Δ' and vice-versa.

We say that Δ and Δ' are *first-order interdefinable*.

"Δ has a fo definition from Δ'" is a *quasiorder* on relational structures over the same domain.

This quasiorder, factored by f.o.-interdefinability, becomes a *complete lattice*.

Finer classifications of the reducts of Γ, e.g. up to

- Existential interdefinability

Possible classifications

Consider two reducts Δ, Δ' of Γ *equivalent* iff Δ has a fo definition from Δ' and vice-versa.

We say that Δ and Δ' are *first-order interdefinable*.

“Δ has a fo definition from Δ'” is a *quasiorder* on relational structures over the same domain.

This quasiorder, factored by f.o.-interdefinability, becomes a *complete lattice*.

Finer classifications of the reducts of Γ, e.g. up to

- Existential interdefinability
- Existential positive interdefinability

Possible classifications

Consider two reducts Δ, Δ' of Γ *equivalent* iff Δ has a fo definition from Δ' and vice-versa.

We say that Δ and Δ' are *first-order interdefinable*.

"Δ has a fo definition from Δ'" is a *quasiorder* on relational structures over the same domain.

This quasiorder, factored by f.o.-interdefinability, becomes a *complete lattice*.

Finer classifications of the reducts of Γ, e.g. up to

- Existential interdefinability
- Existential positive interdefinability
- Primitive positive interdefinability

Example: The dense linear order

Example: The dense linear order

Denote by $(\mathbb{Q};<)$ be the dense linear order, and set

$$\begin{aligned}\mathrm{betw}(x,y,z) &:= \{(x,y,z) \in \mathbb{Q}^3 : x<y<z \text{ or } z<y<x\}\\ \mathrm{cycl}(x,y,z) &:= \{(x,y,z) \in \mathbb{Q}^3 : x<y<z \text{ or } z<x<y\\ &\qquad \text{ or } y<z<x\}\\ \mathrm{sep}(x,y,z,w) &:= \{(x,y,z,w) \in \mathbb{Q}^4 : \ldots\}\end{aligned}$$

Example: The dense linear order

Denote by $(\mathbb{Q};<)$ be the dense linear order, and set

$$\begin{aligned}\text{betw}(x,y,z) :=& \{(x,y,z) \in \mathbb{Q}^3 : x < y < z \text{ or } z < y < x\}\\ \text{cycl}(x,y,z) :=& \{(x,y,z) \in \mathbb{Q}^3 : x < y < z \text{ or } z < x < y\\ & \text{or } y < z < x\}\\ \text{sep}(x,y,z,w) :=& \{(x,y,z,w) \in \mathbb{Q}^4 : \dots\}\end{aligned}$$

Theorem (Cameron '76)

Let Γ be a reduct of $(\mathbb{Q};<)$. Then:

Example: The dense linear order

Denote by $(\mathbb{Q};<)$ be the dense linear order, and set

$$\begin{aligned}
\text{betw}(x,y,z) &:= \{(x,y,z) \in \mathbb{Q}^3 : x<y<z \text{ or } z<y<x\} \\
\text{cycl}(x,y,z) &:= \{(x,y,z) \in \mathbb{Q}^3 : x<y<z \text{ or } z<x<y \\
&\qquad \text{or } y<z<x\} \\
\text{sep}(x,y,z,w) &:= \{(x,y,z,w) \in \mathbb{Q}^4 : \ldots\}
\end{aligned}$$

Theorem (Cameron '76)

Let Γ be a reduct of $(\mathbb{Q};<)$. Then:

1. Γ is first-order interdefinable with $(\mathbb{Q};<)$, or

Example: The dense linear order

Denote by $(\mathbb{Q}; <)$ be the dense linear order, and set

$$\begin{aligned} \text{betw}(x,y,z) :=& \{(x,y,z) \in \mathbb{Q}^3 : x < y < z \text{ or } z < y < x\} \\ \text{cycl}(x,y,z) :=& \{(x,y,z) \in \mathbb{Q}^3 : x < y < z \text{ or } z < x < y \\ & \text{ or } y < z < x\} \\ \text{sep}(x,y,z,w) :=& \{(x,y,z,w) \in \mathbb{Q}^4 : \dots\} \end{aligned}$$

Theorem (Cameron '76)

Let Γ be a reduct of $(\mathbb{Q}; <)$. Then:

1. Γ is first-order interdefinable with $(\mathbb{Q}; <)$, or
2. Γ is first-order interdefinable with $(\mathbb{Q}; \text{betw})$, or

Example: The dense linear order

Denote by $(\mathbb{Q}; <)$ be the dense linear order, and set

$$\begin{aligned} \mathrm{betw}(x,y,z) &:= \{(x,y,z) \in \mathbb{Q}^3 : x < y < z \text{ or } z < y < x\} \\ \mathrm{cycl}(x,y,z) &:= \{(x,y,z) \in \mathbb{Q}^3 : x < y < z \text{ or } z < x < y \\ &\qquad \text{ or } y < z < x\} \\ \mathrm{sep}(x,y,z,w) &:= \{(x,y,z,w) \in \mathbb{Q}^4 : \ldots\} \end{aligned}$$

Theorem (Cameron '76)

Let Γ be a reduct of $(\mathbb{Q}; <)$. Then:

1. Γ is first-order interdefinable with $(\mathbb{Q}; <)$, or
2. Γ is first-order interdefinable with $(\mathbb{Q}; \mathrm{betw})$, or
3. Γ is first-order interdefinable with $(\mathbb{Q}; \mathrm{cycl})$, or

Example: The dense linear order

Denote by $(\mathbb{Q}; <)$ be the dense linear order, and set

$$\begin{aligned}
\text{betw}(x, y, z) :=& \{(x, y, z) \in \mathbb{Q}^3 : x < y < z \text{ or } z < y < x\} \\
\text{cycl}(x, y, z) :=& \{(x, y, z) \in \mathbb{Q}^3 : x < y < z \text{ or } z < x < y \\
& \text{ or } y < z < x\} \\
\text{sep}(x, y, z, w) :=& \{(x, y, z, w) \in \mathbb{Q}^4 : \ldots\}
\end{aligned}$$

Theorem (Cameron '76)

Let Γ be a reduct of $(\mathbb{Q}; <)$. Then:

1. Γ is first-order interdefinable with $(\mathbb{Q}; <)$, or
2. Γ is first-order interdefinable with $(\mathbb{Q}; \text{betw})$, or
3. Γ is first-order interdefinable with $(\mathbb{Q}; \text{cycl})$, or
4. Γ is first-order interdefinable with $(\mathbb{Q}; \text{sep})$, or

Example: The dense linear order

Denote by $(\mathbb{Q}; <)$ be the dense linear order, and set

$$\begin{aligned} \text{betw}(x,y,z) &:= \{(x,y,z) \in \mathbb{Q}^3 : x < y < z \text{ or } z < y < x\} \\ \text{cycl}(x,y,z) &:= \{(x,y,z) \in \mathbb{Q}^3 : x < y < z \text{ or } z < x < y \\ &\qquad \text{or } y < z < x\} \\ \text{sep}(x,y,z,w) &:= \{(x,y,z,w) \in \mathbb{Q}^4 : \ldots\} \end{aligned}$$

Theorem (Cameron '76)

Let Γ be a reduct of $(\mathbb{Q}; <)$. Then:

1. Γ is first-order interdefinable with $(\mathbb{Q}; <)$, or
2. Γ is first-order interdefinable with $(\mathbb{Q}; \text{betw})$, or
3. Γ is first-order interdefinable with $(\mathbb{Q}; \text{cycl})$, or
4. Γ is first-order interdefinable with $(\mathbb{Q}; \text{sep})$, or
5. Γ is first-order interdefinable with $(\mathbb{Q}; =)$.

Example: The random graph

Example: The random graph

Let $G = (V; E)$ be the random graph, and set for all $k \geq 2$

$R^{(k)} := \{(x_1, \ldots, x_k) \subseteq V^k : x_i$ distinct, number of edges odd$\}$.

Example: The random graph

Let $G = (V; E)$ be the random graph, and set for all $k \geq 2$

$R^{(k)} := \{(x_1, \ldots, x_k) \subseteq V^k : x_i \text{ distinct, number of edges odd}\}.$

Theorem (Thomas '91)

Let Γ be a reduct of G. Then:

Example: The random graph

Let $G = (V; E)$ be the random graph, and set for all $k \geq 2$

$R^{(k)} := \{(x_1, \ldots, x_k) \subseteq V^k : x_i \text{ distinct, number of edges odd}\}.$

Theorem (Thomas '91)

Let Γ be a reduct of G. Then:

1. Γ is first-order interdefinable with $(V; E)$, or

Example: The random graph

Let $G = (V; E)$ be the random graph, and set for all $k \geq 2$

$$R^{(k)} := \{(x_1, \ldots, x_k) \subseteq V^k : x_i \text{ distinct, number of edges odd}\}.$$

Theorem (Thomas '91)

Let Γ be a reduct of G. Then:

1. Γ is first-order interdefinable with $(V; E)$, or
2. Γ is first-order interdefinable with $(V; R^{(3)})$, or

Example: The random graph

Let $G = (V; E)$ be the random graph, and set for all $k \geq 2$

$R^{(k)} := \{(x_1, \ldots, x_k) \subseteq V^k : x_i$ distinct, number of edges odd$\}$.

Theorem (Thomas '91)

Let Γ be a reduct of G. Then:

1. Γ is first-order interdefinable with $(V; E)$, or
2. Γ is first-order interdefinable with $(V; R^{(3)})$, or
3. Γ is first-order interdefinable with $(V; R^{(4)})$, or

Example: The random graph

Let $G = (V; E)$ be the random graph, and set for all $k \geq 2$

$R^{(k)} := \{(x_1, \ldots, x_k) \subseteq V^k : x_i$ distinct, number of edges odd$\}$.

Theorem (Thomas ’91)

Let Γ be a reduct of G. Then:

1. Γ is first-order interdefinable with $(V; E)$, or
2. Γ is first-order interdefinable with $(V; R^{(3)})$, or
3. Γ is first-order interdefinable with $(V; R^{(4)})$, or
4. Γ is first-order interdefinable with $(V; R^{(5)})$, or

Example: The random graph

Let $G = (V; E)$ be the random graph, and set for all $k \geq 2$

$R^{(k)} := \{(x_1, \ldots, x_k) \subseteq V^k : x_i$ distinct, number of edges odd$\}$.

Theorem (Thomas '91)

Let Γ be a reduct of G. Then:

1. Γ is first-order interdefinable with $(V; E)$, or
2. Γ is first-order interdefinable with $(V; R^{(3)})$, or
3. Γ is first-order interdefinable with $(V; R^{(4)})$, or
4. Γ is first-order interdefinable with $(V; R^{(5)})$, or
5. Γ is first-order interdefinable with $(V; =)$.

Further examples

Further examples

Theorem (Thomas ’91)

The homogeneous K_n-free graph has 2 reducts, up to f.o.-interdefinability.

Further examples

Theorem (Thomas '91)

The homogeneous K_n-free graph has 2 reducts, up to f.o.-interdefinability.

Theorem (Thomas '96)

The homogeneous k-graph has $2^k + 1$ reducts, up to f.o.-interdefinability.

Further examples

Theorem (Thomas '91)

The homogeneous K_n-free graph has 2 reducts, up to f.o.-interdefinability.

Theorem (Thomas '96)

The homogeneous k-graph has $2^k + 1$ reducts, up to f.o.-interdefinability.

Theorem (Junker, Ziegler '08)

$(\mathbb{Q}; <, 0)$ has 116 reducts, up to f.o.-interdefinability.

Thomas' conjecture

Conjecture (Thomas '91)

Let Γ be homogeneous in a finite language.

Then Γ has finitely many reducts up to f.o.-interdefinability.

Finer classifications

Finer classifications

A formula is *existential* iff
it is of the form $\exists x_1, \ldots, x_n . \psi$, where ψ is quantifier-free.

Finer classifications

A formula is *existential* iff
it is of the form $\exists x_1, \dots, x_n . \psi$, where ψ is quantifier-free.

A formula is *existential positive* iff
it is existential and does not contain negations.

Finer classifications

A formula is *existential* iff
it is of the form $\exists x_1, \ldots, x_n.\psi$, where ψ is quantifier-free.

A formula is *existential positive* iff
it is existential and does not contain negations.

A formula is *primitive positive* iff
it is existential positive and does not contain disjunctions.

Finer classifications

A formula is *existential* iff
it is of the form $\exists x_1, \ldots, x_n.\psi$, where ψ is quantifier-free.

A formula is *existential positive* iff
it is existential and does not contain negations.

A formula is *primitive positive* iff
it is existential positive and does not contain disjunctions.

Theorem (Bodirsky, Chen, P. '08)

For the structure $\Gamma := (X; =)$, there exist:

Finer classifications

A formula is *existential* iff
it is of the form $\exists x_1, \ldots, x_n.\psi$, where ψ is quantifier-free.

A formula is *existential positive* iff
it is existential and does not contain negations.

A formula is *primitive positive* iff
it is existential positive and does not contain disjunctions.

Theorem (Bodirsky, Chen, P. ’08)

For the structure $\Gamma := (X; =)$, there exist:

- 1 reduct up to first order / existential interdefinability

Finer classifications

A formula is *existential* iff
it is of the form $\exists x_1, \ldots, x_n.\psi$, where ψ is quantifier-free.

A formula is *existential positive* iff
it is existential and does not contain negations.

A formula is *primitive positive* iff
it is existential positive and does not contain disjunctions.

Theorem (Bodirsky, Chen, P. ’08)

For the structure $\Gamma := (X; =)$, there exist:

- 1 reduct up to first order / existential interdefinability
- $\aleph_0$ reducts up to existential positive interdefinability

Finer classifications

A formula is *existential* iff
it is of the form $\exists x_1, \ldots, x_n . \psi$, where ψ is quantifier-free.

A formula is *existential positive* iff
it is existential and does not contain negations.

A formula is *primitive positive* iff
it is existential positive and does not contain disjunctions.

Theorem (Bodirsky, Chen, P. '08)

For the structure $\Gamma := (X; =)$, there exist:

- 1 reduct up to first order / existential interdefinability
- $\aleph_0$ reducts up to existential positive interdefinability
- $2^{\aleph_0}$ reducts up to primitive positive interdefinability

Groups, Monoids, Clones

Groups, Monoids, Clones

Theorem

Let Γ be ω-categorical.

- The mapping $\Delta \mapsto \text{Aut}(\Delta)$ is a one-to-one correspondence between the first-order closed reducts of Γ and the closed supergroups of $\text{Aut}(\Gamma)$.

Groups, Monoids, Clones

Theorem

Let Γ be ω-categorical.

- The mapping $\Delta \mapsto \text{Aut}(\Delta)$ is a one-to-one correspondence between the first-order closed reducts of Γ and the closed supergroups of $\text{Aut}(\Gamma)$.
- The mapping $\Delta \mapsto \text{End}(\Delta)$ is a one-to-one correspondence between the existential positive closed reducts of Γ and the closed supermonoids of $\text{Aut}(\Gamma)$.

Groups, Monoids, Clones

Theorem

Let Γ be ω-categorical.

- The mapping $\Delta \mapsto \mathrm{Aut}(\Delta)$ is a one-to-one correspondence between the first-order closed reducts of Γ and the closed supergroups of $\mathrm{Aut}(\Gamma)$.
- The mapping $\Delta \mapsto \mathrm{End}(\Delta)$ is a one-to-one correspondence between the existential positive closed reducts of Γ and the closed supermonoids of $\mathrm{Aut}(\Gamma)$.
- The mapping $\Delta \mapsto \mathrm{Pol}(\Delta)$ is a one-to-one correspondence between the primitive positive closed reducts of Γ and the closed superclones of $\mathrm{Aut}(\Gamma)$.

The reducts of the random graph, revisited

Let $G := (V; E)$ be the random graph.

The reducts of the random graph, revisited

Let $G := (V; E)$ be the random graph.

Let $\bar{G}$ be the graph that arises by switching edges and non-edges.

The reducts of the random graph, revisited

Let $G := (V; E)$ be the random graph.

Let $\bar{G}$ be the graph that arises by switching edges and non-edges.

Let $- : V \to V$ be an isomorphism between G and $\bar{G}$.

The reducts of the random graph, revisited

Let $G := (V; E)$ be the random graph.

Let $\bar{G}$ be the graph that arises by switching edges and non-edges.

Let $- : V \to V$ be an isomorphism between G and $\bar{G}$.

For $c \in V$, let G_c be the graph that arises by switching all edges and non-edges from c.

The reducts of the random graph, revisited

Let $G := (V; E)$ be the random graph.

Let $\bar{G}$ be the graph that arises by switching edges and non-edges.

Let $- : V \to V$ be an isomorphism between G and $\bar{G}$.

For $c \in V$, let G_c be the graph that arises by switching all edges and non-edges from c.

Let $\mathrm{sw}_c : V \to V$ be an isomorphism between G and G_c.

The reducts of the random graph, revisited

Let $G := (V; E)$ be the random graph.

Let $\bar{G}$ be the graph that arises by switching edges and non-edges.

Let $- : V \to V$ be an isomorphism between G and $\bar{G}$.

For $c \in V$, let G_c be the graph that arises by switching all edges and non-edges from c.

Let $\mathrm{sw}_c : V \to V$ be an isomorphism between G and G_c.

Theorem (Thomas '91)

The closed groups containing $\mathrm{Aut}(G)$ are the following:

The reducts of the random graph, revisited

Let $G := (V; E)$ be the random graph.

Let $\bar{G}$ be the graph that arises by switching edges and non-edges.

Let $- : V \to V$ be an isomorphism between G and $\bar{G}$.

For $c \in V$, let G_c be the graph that arises by switching all edges and non-edges from c.

Let $\mathrm{sw}_c : V \to V$ be an isomorphism between G and G_c.

Theorem (Thomas '91)

The closed groups containing $\mathrm{Aut}(G)$ are the following:

1. $\mathrm{Aut}(G)$

The reducts of the random graph, revisited

Let $G := (V; E)$ be the random graph.

Let $\bar{G}$ be the graph that arises by switching edges and non-edges.

Let $- : V \to V$ be an isomorphism between G and $\bar{G}$.

For $c \in V$, let G_c be the graph that arises by switching all edges and non-edges from c.

Let $\mathrm{sw}_c : V \to V$ be an isomorphism between G and G_c.

Theorem (Thomas '91)

The closed groups containing $\mathrm{Aut}(G)$ are the following:

1. $\mathrm{Aut}(G)$
2. $\langle \{-\} \cup \mathrm{Aut}(G) \rangle$

The reducts of the random graph, revisited

Let $G := (V; E)$ be the random graph.

Let $\bar{G}$ be the graph that arises by switching edges and non-edges.

Let $- : V \to V$ be an isomorphism between G and $\bar{G}$.

For $c \in V$, let G_c be the graph that arises by switching all edges and non-edges from c.

Let $\mathrm{sw}_c : V \to V$ be an isomorphism between G and G_c.

Theorem (Thomas '91)

The closed groups containing $\mathrm{Aut}(G)$ are the following:

1. $\mathrm{Aut}(G)$
2. $\langle \{-\} \cup \mathrm{Aut}(G) \rangle$
3. $\langle \{\mathrm{sw}_c\} \cup \mathrm{Aut}(G) \rangle$

The reducts of the random graph, revisited

Let $G := (V; E)$ be the random graph.

Let $\bar{G}$ be the graph that arises by switching edges and non-edges.

Let $- : V \to V$ be an isomorphism between G and $\bar{G}$.

For $c \in V$, let G_c be the graph that arises by switching all edges and non-edges from c.

Let $\mathrm{sw}_c : V \to V$ be an isomorphism between G and G_c.

Theorem (Thomas '91)

The closed groups containing $\mathrm{Aut}(G)$ are the following:

1. $\mathrm{Aut}(G)$
2. $\langle \{-\} \cup \mathrm{Aut}(G) \rangle$
3. $\langle \{\mathrm{sw}_c\} \cup \mathrm{Aut}(G) \rangle$
4. $\langle \{-, \mathrm{sw}_c\} \cup \mathrm{Aut}(G) \rangle$

The reducts of the random graph, revisited

Let $G := (V; E)$ be the random graph.

Let $\bar{G}$ be the graph that arises by switching edges and non-edges.

Let $- : V \to V$ be an isomorphism between G and $\bar{G}$.

For $c \in V$, let G_c be the graph that arises by switching all edges and non-edges from c.

Let $\mathrm{sw}_c : V \to V$ be an isomorphism between G and G_c.

Theorem (Thomas '91)

The closed groups containing $\mathrm{Aut}(G)$ are the following:

1. $\mathrm{Aut}(G)$
2. $\langle \{-\} \cup \mathrm{Aut}(G) \rangle$
3. $\langle \{\mathrm{sw}_c\} \cup \mathrm{Aut}(G) \rangle$
4. $\langle \{-, \mathrm{sw}_c\} \cup \mathrm{Aut}(G) \rangle$
5. The full symmetric group S_V.

How to find all reducts up to …-interdefinability?

Climb up the lattice!

Canonical functions on the Random graph

Definition. $f : V \to V$ is *canonical* iff

Canonical functions on the Random graph

Definition. $f : V \to V$ is *canonical* iff

for all $x, y, u, v \in V$,

if (x, y) and (u, v) have the same type,

Canonical functions on the Random graph

Definition. $f : V \to V$ is *canonical* iff
for all $x, y, u, v \in V$,
if (x, y) and (u, v) have the same type,
then so do $(f(x), f(y))$ and $(f(u), f(v))$.

Canonical functions on the Random graph

Definition. $f : V \to V$ is *canonical* iff
for all $x, y, u, v \in V$,
if (x, y) and (u, v) have the same type,
then so do $(f(x), f(y))$ and $(f(u), f(v))$.

Examples.

Canonical functions on the Random graph

Definition. $f : V \to V$ is *canonical* iff
for all $x, y, u, v \in V$,
if (x, y) and (u, v) have the same type,
then so do $(f(x), f(y))$ and $(f(u), f(v))$.

Examples.
The identity is canonical.

Canonical functions on the Random graph

Definition. $f : V \to V$ is *canonical* iff
for all $x, y, u, v \in V$,
if (x, y) and (u, v) have the same type,
then so do $(f(x), f(y))$ and $(f(u), f(v))$.

Examples.

The identity is canonical.

$-$ is canonical on V.

Canonical functions on the Random graph

Definition. $f : V \to V$ is *canonical* iff
for all $x, y, u, v \in V$,
if (x, y) and (u, v) have the same type,
then so do $(f(x), f(y))$ and $(f(u), f(v))$.

Examples.

The identity is canonical.

$-$ is canonical on V.

sw_c is canonical on any $F \subseteq V \setminus \{c\}$.

Canonical functions on the Random graph

Definition. $f : V \to V$ is *canonical* iff
for all $x, y, u, v \in V$,
if (x, y) and (u, v) have the same type,
then so do $(f(x), f(y))$ and $(f(u), f(v))$.

Examples.

The identity is canonical.

$-$ is canonical on V.

sw_c is canonical on any $F \subseteq V \setminus \{c\}$.

$f : V \to V$ is *canonical on* $F \subseteq V$ iff its restriction to F is canonical.

Finding canonical behaviour

The class of finite graphs has the following Ramsey property:

Finding canonical behaviour

The class of finite graphs has the following Ramsey property:

For all graphs H
there exists a graph S such that

Finding canonical behaviour

The class of finite graphs has the following Ramsey property:

For all graphs H
there exists a graph S such that
if the edges of S are colored with 2 colors,

Finding canonical behaviour

The class of finite graphs has the following Ramsey property:

For all graphs H
there exists a graph S such that
if the edges of S are colored with 2 colors,
then there is a copy of H in S
on which the coloring is constant.

Finding canonical behaviour

The class of finite graphs has the following Ramsey property:

For all graphs H
there exists a graph S such that
if the edges of S are colored with 2 colors,
then there is a copy of H in S
on which the coloring is constant.

Given $f : V \to V$, color an edge according to the type of its image (3 possibilities).
Same for non-edges.

Finding canonical behaviour

The class of finite graphs has the following Ramsey property:

For all graphs H
there exists a graph S such that
if the edges of S are colored with 2 colors,
then there is a copy of H in S
on which the coloring is constant.

Given $f : V \to V$, color an edge according to the type of its image (3 possibilities).
Same for non-edges.

Conclusion: Every finite graph has a copy in G on which f is canonical.

Patterns in functions on the random graph

Being canonical means:

Patterns in functions on the random graph

Being canonical means:

Turning everything into edges (e_E), or

Patterns in functions on the random graph

Being canonical means:

Turning everything into edges (e_E), or
turning everything into non-edges (e_N), or

Patterns in functions on the random graph

Being canonical means:

Turning everything into edges (e_E), or
turning everything into non-edges (e_N), or
behaving like $-$, or

Patterns in functions on the random graph

Being canonical means:

Turning everything into edges (e_E), or
turning everything into non-edges (e_N), or
behaving like $-$, or
being constant, or

Patterns in functions on the random graph

Being canonical means:

Turning everything into edges (e_E), or
turning everything into non-edges (e_N), or
behaving like $-$, or
being constant, or
behaving like the identity.

Patterns in functions on the random graph

Being canonical means:

Turning everything into edges (e_E), or
turning everything into non-edges (e_N), or
behaving like $-$, or
being constant, or
behaving like the identity.

Let $f : V \to V$.
If $f \notin \mathrm{Aut}(G)$, then there are $c, d \in V$ witnessing this.

Patterns in functions on the random graph

Being canonical means:

Turning everything into edges (e_E), or
turning everything into non-edges (e_N), or
behaving like $-$, or
being constant, or
behaving like the identity.

Let $f : V \to V$.
If $f \notin \mathrm{Aut}(G)$, then there are $c, d \in V$ witnessing this.

The structure $(V; E, c, d)$ has similar Ramsey properties as $(V; E)$.

The minimal monoids on the random graph

Theorem (Thomas '96)

Let $f : V \to V,\ f \notin \mathrm{Aut}(G)$.
Then f generates one of the following:

- A constant operation
- e_E
- e_N
- $-$
- sw_c

The minimal monoids on the random graph

Theorem (Thomas ’96)

Let $f : V \to V,\ f \notin \mathrm{Aut}(G)$.
Then f generates one of the following:

- A constant operation
- e_E
- e_N
- $-$
- sw_c

We thus know the *minimal closed monoids* containing $\mathrm{Aut}(G)$.

The minimal clones on the random graph

Theorem (Bodirsky, P. ’09)

Let $f : V^n \to V,\ f \notin \mathrm{Aut}(G)$.
Then f generates one of the following:

- One of the five minimal unary functions of Thomas’ theorem;
- One of 9 canonical binary injections.

The minimal clones on the random graph

Theorem (Bodirsky, P. ’09)

Let $f : V^n \to V,\ f \notin \mathrm{Aut}(G)$.
Then f generates one of the following:

- One of the five minimal unary functions of Thomas’ theorem;
- One of 9 canonical binary injections.

We thus know the *minimal closed clones* containing $\mathrm{Aut}(G)$.

Ramsey classes

Ramsey classes

Let S, H, P be structures in the same signature τ.

$$S \rightarrow (H)^P$$

means:

Ramsey classes

Let S, H, P be structures in the same signature τ.

$$S \to (H)^P$$

means:

For any coloring of the copies of P in S with 2 colors
there exists a copy of H in S
such that the copies of P in H all have the same color.

Ramsey classes

Let S, H, P be structures in the same signature τ.

$$S \to (H)^P$$

means:

For any coloring of the copies of P in S with 2 colors

there exists a copy of H in S

such that the copies of P in H all have the same color.

Definition

A class $\mathcal{C}$ of structures of the same signature τ is called a *Ramsey class* iff
for all $H, P \in \mathcal{C}$ there is S in $\mathcal{C}$ such that $S \to (H)^P$.

Canonical functions on Ramsey structures

Let Γ now be an arbitrary structure.

Canonical functions on Ramsey structures

Let Γ now be an arbitrary structure.

Definition

$f : \Gamma \to \Gamma$ is *canonical* iff

for all tuples $(x_1, \ldots, x_n), (y_1, \ldots, y_n)$ of the same type $(f(x_1), \ldots, f(x_n))$ and $(f(y_1), \ldots, f(y_n))$ have the same type too.

Canonical functions on Ramsey structures

Let Γ now be an arbitrary structure.

Definition

$f : \Gamma \to \Gamma$ is *canonical* iff
for all tuples $(x_1, \ldots, x_n), (y_1, \ldots, y_n)$ of the same type $(f(x_1), \ldots, f(x_n))$ and $(f(y_1), \ldots, f(y_n))$ have the same type too.

Observation. Let Γ be Ramsey, ordered, and ω-categorical.
Let H be a finite structure in the age of Γ.
Then there is a copy of H in Γ on which f is canonical.

Canonical functions on Ramsey structures

Let Γ now be an arbitrary structure.

Definition

$f : \Gamma \to \Gamma$ is *canonical* iff
for all tuples $(x_1, \ldots, x_n), (y_1, \ldots, y_n)$ of the same type $(f(x_1), \ldots, f(x_n))$ and $(f(y_1), \ldots, f(y_n))$ have the same type too.

Observation. Let Γ be Ramsey, ordered, and ω-categorical.
Let H be a finite structure in the age of Γ.
Then there is a copy of H in Γ on which f is canonical.

Thus: If Γ is in addition homogeneous in a finite language, then any $f : V \to V$ generates a canonical function,

Canonical functions on Ramsey structures

Let Γ now be an arbitrary structure.

Definition

$f : \Gamma \to \Gamma$ is *canonical* iff
for all tuples $(x_1, \ldots, x_n), (y_1, \ldots, y_n)$ of the same type $(f(x_1), \ldots, f(x_n))$ and $(f(y_1), \ldots, f(y_n))$ have the same type too.

Observation. Let Γ be Ramsey, ordered, and ω-categorical.
Let H be a finite structure in the age of Γ.
Then there is a copy of H in Γ on which f is canonical.

Thus: If Γ is in addition homogeneous in a finite language, then any $f : V \to V$ generates a canonical function, but it could be the identity.

Canonical functions on Ramsey structures

Let Γ now be an arbitrary structure.

Definition

$f : \Gamma \to \Gamma$ is *canonical* iff
for all tuples $(x_1, \ldots, x_n), (y_1, \ldots, y_n)$ of the same type $(f(x_1), \ldots, f(x_n))$ and $(f(y_1), \ldots, f(y_n))$ have the same type too.

Observation. Let Γ be Ramsey, ordered, and ω-categorical.
Let H be a finite structure in the age of Γ.
Then there is a copy of H in Γ on which f is canonical.

Thus: If Γ is in addition homogeneous in a finite language, then any $f : V \to V$ generates a canonical function, but it could be the identity.

We would like to fix $c_1, \ldots, c_n$ witnessing $f \notin \mathrm{Aut}(\Gamma)$, and have canonical behavior on $(\Gamma, c_1, \ldots, c_n)$.

Adding constants to Ramsey classes

Problem

If Γ is Ramsey, is $(\Gamma, c_1, \ldots, c_n)$ still Ramsey?

Adding constants to Ramsey classes

Problem

If Γ is Ramsey, is $(\Gamma, c_1, \ldots, c_n)$ still Ramsey?

Theorem (Kechris, Pestov, Todorcevic '05)

An ordered homogeneous structure Δ is Ramsey iff its automorphism group is *extremely amenable*, i.e., it has a fixed point whenever it acts on a compact topological space.

Adding constants to Ramsey classes

Problem

If Γ is Ramsey, is $(\Gamma, c_1, \ldots, c_n)$ still Ramsey?

Theorem (Kechris, Pestov, Todorcevic '05)

An ordered homogeneous structure Δ is Ramsey iff its automorphism group is *extremely amenable*, i.e., it has a fixed point whenever it acts on a compact topological space.

Easy observation (Tsankov '10)

Every open subgroup of an extremely amenable group is extremely amenable.

Adding constants to Ramsey classes

Problem

If Γ is Ramsey, is $(\Gamma, c_1, \ldots, c_n)$ still Ramsey?

Theorem (Kechris, Pestov, Todorcevic '05)

An ordered homogeneous structure Δ is Ramsey iff its automorphism group is *extremely amenable*, i.e., it has a fixed point whenever it acts on a compact topological space.

Easy observation (Tsankov '10)

Every open subgroup of an extremely amenable group is extremely amenable.

Corollary

If Γ is ordered, homogeneous, and Ramsey, then so is $(\Gamma, c_1, \ldots, c_n)$.

Minimal monoids above Ramsey structures

Thus:

If Γ is ordered Ramsey, $f : \Gamma \to \Gamma$, and $c_1, \ldots, c_n \in \Gamma$, then f generates a function canonical for $(\Gamma, c_1, \ldots, c_n)$ which behaves like f on $\{c_1, \ldots, c_n\}$.

Minimal monoids above Ramsey structures

Thus:

If Γ is ordered Ramsey, $f : \Gamma \to \Gamma$, and $c_1, \ldots, c_n \in \Gamma$, then f generates a function canonical for $(\Gamma, c_1, \ldots, c_n)$ which behaves like f on $\{c_1, \ldots, c_n\}$.

Theorem (Bodirsky, P., Tsankov '10)

Let Γ be a finite language reduct of a finite language homogeneous ordered Ramsey structure. Then:

Minimal monoids above Ramsey structures

Thus:

If Γ is ordered Ramsey, $f : \Gamma \to \Gamma$, and $c_1, \ldots, c_n \in \Gamma$, then f generates a function canonical for $(\Gamma, c_1, \ldots, c_n)$ which behaves like f on $\{c_1, \ldots, c_n\}$.

Theorem (Bodirsky, P., Tsankov '10)

Let Γ be a finite language reduct of a finite language homogeneous ordered Ramsey structure. Then:

- There are finitely many minimal closed supermonoids of $\text{End}(\Gamma)$.

Minimal monoids above Ramsey structures

Thus:

If Γ is ordered Ramsey, $f : \Gamma \to \Gamma$, and $c_1, \ldots, c_n \in \Gamma$, then f generates a function canonical for $(\Gamma, c_1, \ldots, c_n)$ which behaves like f on $\{c_1, \ldots, c_n\}$.

Theorem (Bodirsky, P., Tsankov '10)

Let Γ be a finite language reduct of a finite language homogeneous ordered Ramsey structure. Then:

- There are finitely many minimal closed supermonoids of $\mathrm{End}(\Gamma)$.
- Every closed supermonoid of $\mathrm{End}(\Gamma)$ contains a minimal closed supermonoid of $\mathrm{End}(\Gamma)$.

Minimal clones above Ramsey structures

Going to products of Γ, we get:

Minimal clones above Ramsey structures

Going to products of Γ, we get:

Theorem (Bodirsky, P., Tsankov '10)

Let Γ be a finite language reduct of a finite language homogeneous ordered Ramsey structure. Then:

Minimal clones above Ramsey structures

Going to products of Γ, we get:

Theorem (Bodirsky, P., Tsankov '10)

Let Γ be a finite language reduct of a finite language homogeneous ordered Ramsey structure. Then:

- There are finitely many minimal closed clones containing $\mathrm{Pol}(\Gamma)$. (Arity bound: $|S_2(\Gamma)|$.)

Minimal clones above Ramsey structures

Going to products of Γ, we get:

Theorem (Bodirsky, P., Tsankov '10)

Let Γ be a finite language reduct of a finite language homogeneous ordered Ramsey structure. Then:

- There are finitely many minimal closed clones containing $\mathrm{Pol}(\Gamma)$. (Arity bound: $|S_2(\Gamma)|$.)
- Every closed clone above $\mathrm{Pol}(\Gamma)$ contains a minimal one.

Schaefer's theorem for graphs

Theorem (Bodirsky, P. '10)

Let Γ be a reduct of the random graph. Then CSP(Γ) is either in P or NP-complete.

Method: Prove hardness for certain relations, and tractability for certain polymorphisms.

If a reduct of G does not pp define any of the hard relations, then it has polymorphisms violating them.

These polymorphisms can be assumed to be canonical.

Thus they can easily be handled, and one can show that they produce one of the tractable polymorphisms.

The theorem in more detail

The theorem in more detail

Theorem

Let Γ be a reduct of the random graph. Then:

- Either Γ has one out of 17 canonical polymorphisms, and CSP(Γ) is tractable,
- or CSP(Γ) is NP-complete.

The theorem in more detail

Theorem

Let Γ be a reduct of the random graph. Then:

- Either Γ has one out of 17 canonical polymorphisms, and CSP(Γ) is tractable,
- or CSP(Γ) is NP-complete.

Theorem

Let Γ be a reduct of the random graph. Then:

- Either Γ pp-defines one out of 4 hard relations, and CSP(Γ) is NP-complete,
- or CSP(Γ) is tractable.

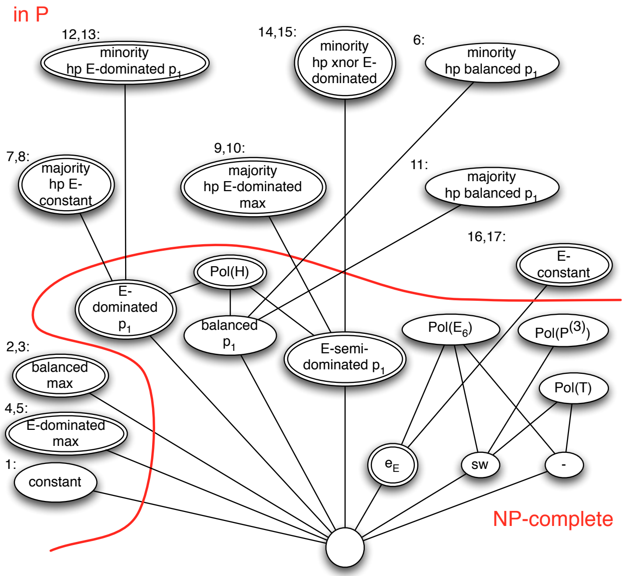
Examples of tractable polymorphisms

Theorem

The following 17 distinct clones are precisely the minimal tractable local clones containing $\mathrm{Aut}(G)$:

1. The clone generated by a constant operation.
2. The clone generated by a balanced binary injection of type max.
3. The clone generated by a balanced binary injection of type min.
4. The clone generated by an *E*-dominated binary injection of type max.
5. The clone generated by an *N*-dominated binary injection of type min.
6. The clone generated by a function of type majority which is hyperplanely balanced and of type projection.
7. The clone generated by a function of type majority which is hyperplanely *E*-constant.
8. The clone generated by a function of type majority which is hyperplanely *N*-constant.
9. The clone generated by a function of type majority which is

Classification



The Meta Problem

The Meta Problem

Meta-Problem of Graph-SAT(Ψ)

INPUT: A finite set Ψ of graph formulas.

QUESTION: Is Graph-SAT(Ψ) in P?

The Meta Problem

Meta-Problem of Graph-SAT(Ψ)

INPUT: A finite set Ψ of graph formulas.

QUESTION: Is Graph-SAT(Ψ) in P?

Theorem

The Meta-Problem of Graph-SAT(Ψ) is decidable.

Temp-SAT

Theorem (Bodirsky and Kara ’08)

Let Γ be a reduct of the order of the rationals. Then Γ either has one out of 9 binary canonical polymorphisms, and CSP(Γ) is in P, or CSP(Γ) is NP-complete.

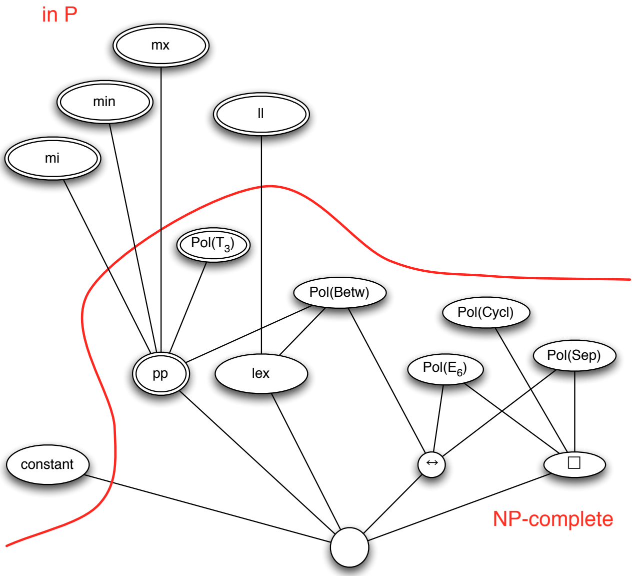
Method: Prove hardness for certain relations, and tractability for certain polymorphisms.

If a reduct of the order does not pp define any of the hard relations, then it has polymorphisms violating them.

These polymorphisms can be assumed to be canonical.

Thus they can easily be handled, and one can show that they produce one of the tractable polymorphisms.

Classification



Boolean-SAT

Theorem (Schaefer '78)

Let Γ be a structure on a Boolean domain. Then Γ either has one of the polymorphisms listed below, and CSP(Γ) is in P, or CSP(Γ) is NP-complete.

- Constant
- Max
- Min
- Majority
- Minority

Proof: Any operation which depends on at least two variables generates Max, Min, Majority, or Minority. If all polymorphisms of Γ depend on at most one variable, and no polymorphism is constant, then the polymorphisms preserve

$\{(1,0,0),(0,1,0),(0,0,1),(1,1,0),(1,0,1),(0,1,1)\}$,

Summary

- CSPs model many real computational problems from theoretical computer science.

Summary

- CSPs model many real computational problems from theoretical computer science.
- Universal algebra useful in classifications since Pol(Γ) captures complexity.

Summary

- CSPs model many real computational problems from theoretical computer science.
- Universal algebra useful in classifications since $\text{Pol}(\Gamma)$ captures complexity.
- Many real computational problems are infinite domain CSPs.

Summary

- CSPs model many real computational problems from theoretical computer science.
- Universal algebra useful in classifications since $\mathrm{Pol}(\Gamma)$ captures complexity.
- Many real computational problems are infinite domain CSPs.
- On infinite domains, add model theory + Ramsey theory to study polymorphisms. They then become functions on types, hence finite.