The Complexity of Constraint Satisfaction Problems

Andrei Krokhin
Durham University

Tutorial, Part III - Here and Now

Recap from Previous Lectures

- Three forms of CSP: Variable-Value, Sat, and Hom
- Parameterisation: CSP(Γ), CSP($\mathcal{B}$)
- Translation into universal algebra, CSP($\mathbf{A}$)
- Algebraic Dichotomy Conjecture
- Hardness results
- Tractability via Few Subpowers

Today

1. Constraints and Their Complexity: An introduction
2. Universal Algebra for CSP: A general theory
3. UA (and a bit of logic) for CSP: A bigger picture
    - Datalog and some fragments
    - Finer complexity classification
    - Some Tame Congruence Theory
    - Hardness/non-expressibility results
    - A private suspicion

Datalog

- A Datalog Program consists of rules, and takes as input a relational structure.
- a typical Datalog rule might look like this one:

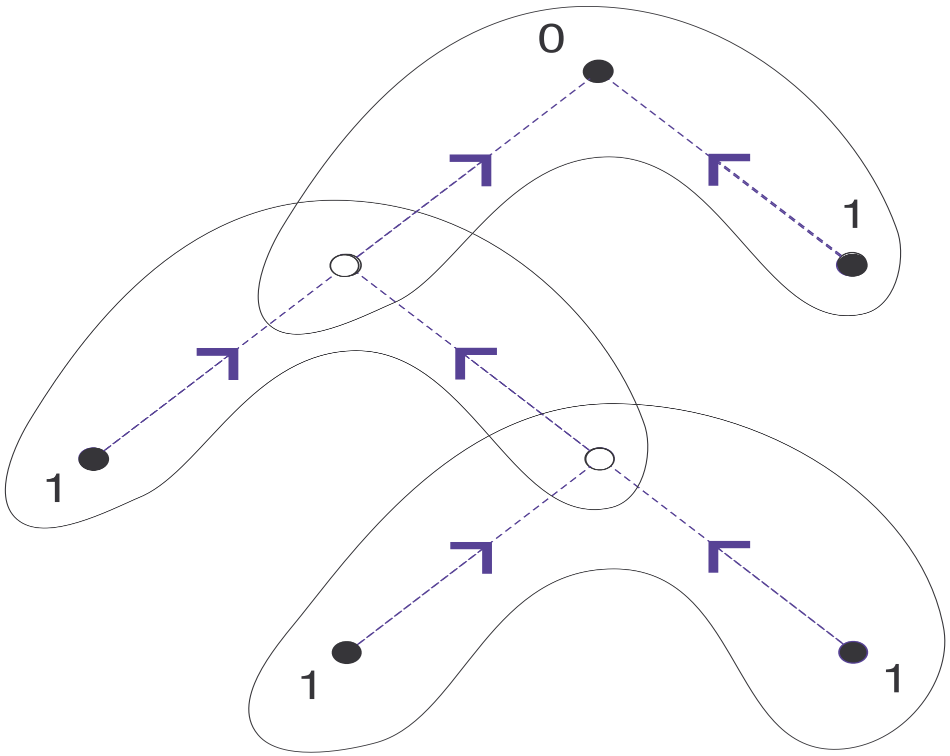
$$\theta_1(x,y) \leftarrow \theta_2(w,u,x), \theta_3(x), R_1(x,y,z), R_2(x,w)$$

- the relations R_1 and R_2 are basic relations of the input structures (EDB's);
- the relations θ_i are auxiliary relations (IDB's);
- the rule stipulates that if the condition on the righthand side (the body of the rule) holds, then the condition of the left (the head) should also hold.

HORN 3-SAT

Recall: HORN 3-SAT is CSP($\mathcal{B}$) where
$\mathcal{B} = (\{0,1\}; R, \{0\}, \{1\})$ with $R = \{(x,y,z) : (y \wedge z) \to x\}$.

Here's an unsatisfiable instance:



Datalog Program for Horn 3-Sat

A Datalog program recursively computes the auxiliary relations (IDBs).

Intuition: locally derive new constraints, trying to get a contradiction (to certify that there's no solution).

$$\begin{array}{rcl} \lambda(x) & \leftarrow & 1(x) \\ \lambda(x) & \leftarrow & \lambda(y), \lambda(z), R(x,y,z) \\ \gamma & \leftarrow & \lambda(x), 0(x) \end{array}$$

The 0-ary relation γ is the goal predicate of the program: it "lights up" precisely if the input structure admits NO homomorphism to the target structure $\mathcal{B}$.

Definability in Datalog

We say that co-CSP($\mathcal{B}$) is definable in Datalog if there exists a Datalog program that accepts precisely those structures that do not admit a homomorphism to $\mathcal{B}$.

In this case, $\mathcal{B}$ is also said to have bounded width.

Theorem 1 *If* co-CSP($\mathcal{B}$) *is definable in Datalog then* CSP($\mathcal{B}$) *is in* **P**.

Idea: IDBs have bounded arity, so the program can do only polynomially many steps before stabilising.

Theorem 2 (Feder,Vardi '98)
If $\mathcal{B} = (Z_p; R, \{1\})$ *where* $R = \{(x, y, z) : x + y = z\}$ *then* co-CSP($\mathcal{B}$) *is not definable in Datalog.*

A Fragment: Linear Datalog

A Datalog program is said to be linear if each rule contains at most one occurrence of an IDB in the body.

In other words, each rule looks like this

$$\theta_1(x,y) \leftarrow \theta_2(w,u,x), R_1(x,y,z), R_2(x,w)$$

where θ_i's are the only IDBs in it, or like this

$$\theta_1(x,y) \leftarrow R_1(x,y,z), R_2(x,w).$$

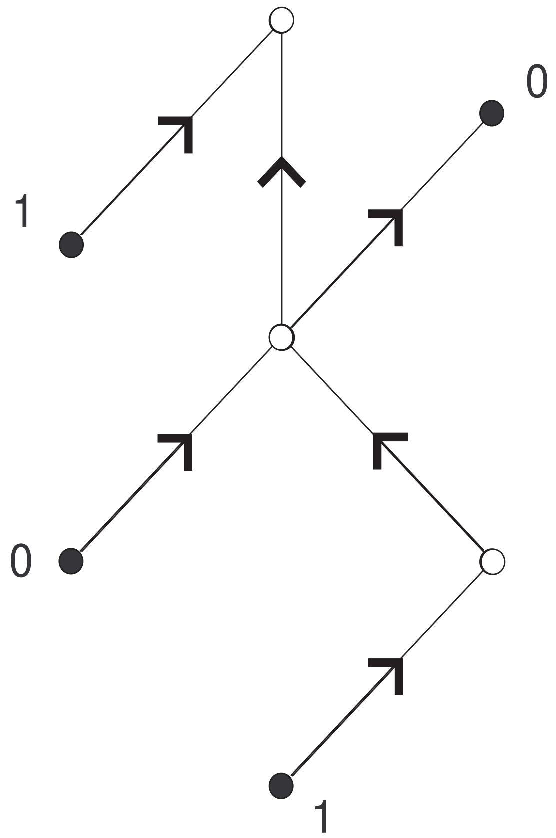
Our program for Horn 3-Sat is non-linear.

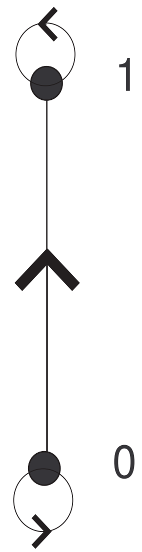
Theorem 3 (Afrati,Cosmodakis '89)
Horn 3-Sat *is not definable in Linear Datalog.*

Path or Directed Reachability

Path is co-CSP($\mathcal{B}$) where $\mathcal{B} = (\{0, 1\}; \leq, \{0\}, \{1\})$.

Here's an unsatisfiable instance (and the target) :





A Linear Program for PATH

$$
\begin{aligned}
\lambda(x) &\leftarrow 1(x) \\
\lambda(y) &\leftarrow \lambda(x), R_{\leq}(x,y) \\
\gamma &\leftarrow \lambda(x), 0(x)
\end{aligned}
$$

Expressibility in Linear Datalog

Recall that PATH is **NL**-complete.

Theorem 4 *If* co-CSP($\mathcal{B}$) *is definable in Linear Datalog then* CSP($\mathcal{B}$) *is in* **NL**.

Idea: by the linearity, the program accepts if and only if there is a derivation path that ends in the goal predicate: this amounts to directed reachability.

Coincidence? For each CSP($\mathcal{B}$) currently known to be in **NL**, co-CSP($\mathcal{B}$) is definable in Linear Datalog.

A Fragment: Symmetric Datalog

A Datalog program is said to be symmetric if (i) it is linear and (ii) it is invariant under symmetry of rules.

In other words, if the program contains the rule

$$\theta_1(x, y) \leftarrow \theta_2(w, u, x), R_1(x, y, z), R_2(x, w)$$

then it must also contain its symmetric:

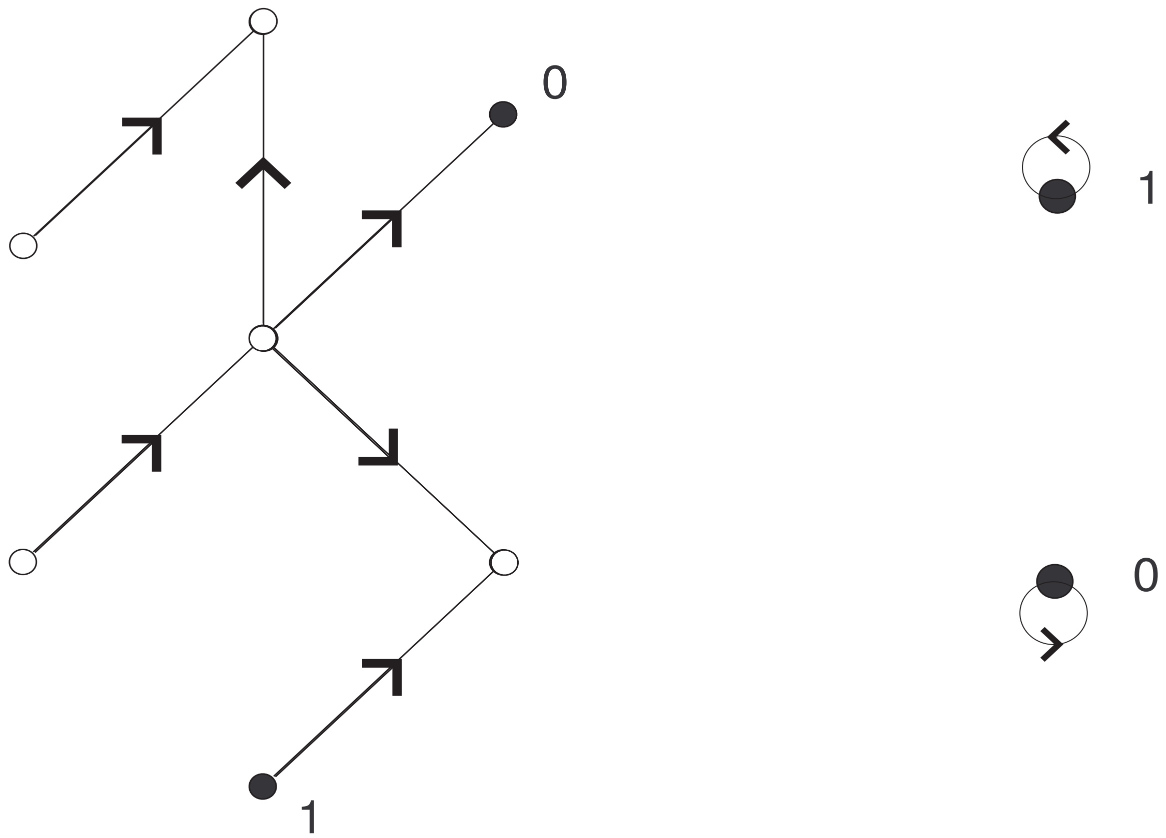
$$\theta_2(w, u, x) \leftarrow \theta_1(x, y), R_1(x, y, z), R_2(x, w).$$

Our program for Path is linear, but not symmetric.

Theorem 5 (Egri,Larose,Tesson '08)
Path *is not definable in Symmetric Datalog.*

Undirected Reachability

This is co-CSP($\mathcal{B}$) where $\mathcal{B} = (\{0,1\}; =, \{0\}, \{1\})$.
Here's an unsatisfiable instance (and the target) :



A Symmetric Program for Undirected Reachability

$$
\begin{aligned}
\lambda(x) &\leftarrow 1(x) \\
\lambda(y) &\leftarrow \lambda(x), R_{=}(x, y) \\
\lambda(x) &\leftarrow \lambda(y), R_{=}(x, y) \\
\gamma &\leftarrow \lambda(x), 0(x)
\end{aligned}
$$

Expressibility in Symmetric Datalog

Theorem 6 (Reingold'06)
Undirected Reachability *is in* **L**.

Theorem 7 (Egri,Larose,Tesson'07) *If* co-CSP($\mathcal{B}$) *is definable in Symmetric Datalog then* CSP($\mathcal{B}$) *is in* **L**.

Idea: the program accepts if and only if there is a derivation path that ends in the goal predicate: by the symmetry, this amounts to undirected reachability.

Coincidence? For each CSP($\mathcal{B}$) currently known to be in **L**, co-CSP($\mathcal{B}$) is definable in Symmetric Datalog.

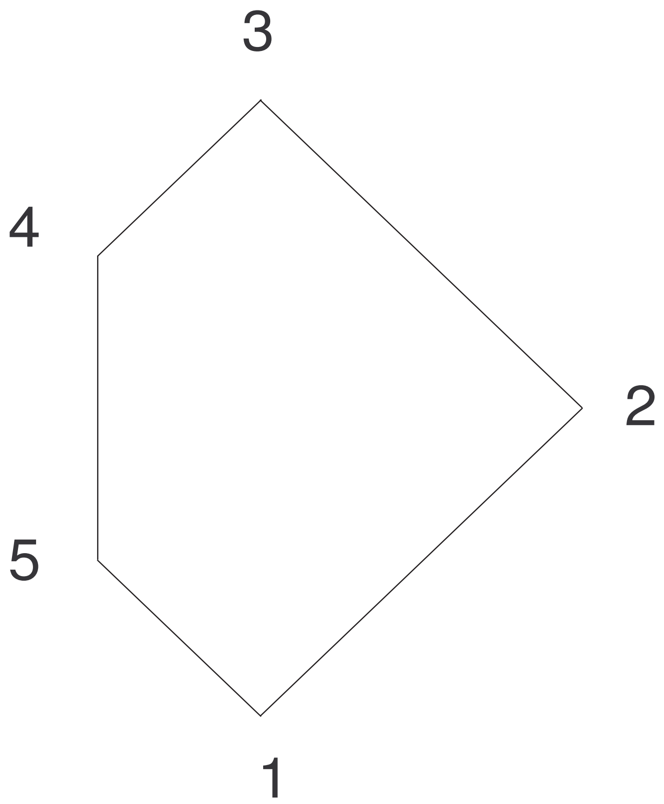
Back to Algebra: TCT Types, Vaguely

- to each (finite) algebra $\mathbf{A}$ is associated a set of types describing the basic "local behaviours" of $\mathbf{A}$;
- the possible types are:
  - the unary type, or type $\mathbf{1}$;
  - the affine type, or type $\mathbf{2}$;
  - the Boolean type, or type $\mathbf{3}$;
  - the lattice type, or type $\mathbf{4}$;
  - the semilattice type, or type $\mathbf{5}$.
- the typeset of the variety $\mathrm{var}(\mathbf{A})$ is the union of all typesets of all finite algebras in it.

The Ordering of the Types

We shall refer later to the following ordering of types:

$$\mathbf{1} < \mathbf{2} < \mathbf{3} > \mathbf{4} > \mathbf{5} > \mathbf{1}$$



Strictly Simple Algebras

A factor of an algebra is a homomorphic image of a subalgebra.

An algebra is strictly simple if it has no proper factors, i.e. it is simple and has no non-trivial subalgebras.

Every strictly simple idempotent algebra has a unique type associated to it.

Lemma 1 (Valeriote '07) *Let* $\mathbf{A}$ *be an idempotent algebra, and suppose type* $\mathbf{i}$ *is in the typeset of* $\mathrm{var}(\mathbf{A})$*. Then* $\mathbf{A}$ *has a strictly simple factor of type* $\leq \mathbf{i}$*.*

Strictly Simple Algebras and Types

Á. Szendrei classified strictly simple algebras by types.
We need the following four consequences:

Lemma 2 (unary type 1) *Let* $\mathbf{A}$ *be a strictly simple idempotent algebra of unary type. Then it is a 2-element algebra, and its basic operations preserve the relation* $\{0,1\}^3 \setminus \{(0,0,0),(1,1,1)\}$.

This smells of NAE SAT.

Lemma 3 (affine type 2) *Let* $\mathbf{A}$ *be a strictly simple idempotent algebra of affine type. Then there exists an Abelian group structure on* D *such that the basic operations of* $\mathbf{A}$ *preserve the relation* $\{(x,y,z) : x+y=z\}$.

This smells of LINEAR EQ'S.

Strictly Simple Algebras and Types

Lemma 4 (lattice type 4) *Let* $\mathbf{A}$ *be a strictly simple idempotent algebra of lattice type. Then it is a 2-element algebra, and its basic operations preserve the usual ordering* $\leq$ *on* $\{0, 1\}$.

This smells of PATH.

Lemma 5 (semilattice type 5) *Let* $\mathbf{A}$ *be a strictly simple idempotent algebra of semilattice type. Then it is isomorphic to a 2-element algebra whose basic operations preserve the relation* $\{(x, y, z) : (y \wedge z) \to x\}$.

This smells of HORN 3-SAT.

A Reduction Lemma

Lemma 6 (Larose,Tesson '07)
Let $\mathcal{B}$ be a core and let $\mathbf{A}$ be the (idempotent) algebra associated to $\mathcal{B}$.
Let $\mathbf{A}'$ be a factor of $\mathbf{A}$, and let $\mathcal{B}'$ be a structure whose basic relations are invariant under the operations of $\mathbf{A}'$.
Then

- *there is a logspace reduction from* $\mathrm{CSP}(\mathcal{B}')$ *to* $\mathrm{CSP}(\mathcal{B})$*;*
- *if* $\text{co-CSP}(\mathcal{B})$ *is definable in (Linear, Symmetric) Datalog then so is* $\text{co-CSP}(\mathcal{B}')$*.*

Hardness and Non-Definability

Corollary 1 (LT'08) *For a core structure $\mathcal{B}$ with associated idempotent algebra $\mathbf{A}$, the following holds.*

var($\mathbf{A}$)		CSP($\mathcal{B}$)	co-CSP($\mathcal{B}$)
omits	*admits*	*complexity*	*definability*
	1	**NP**-*complete*	*not Datalog*
1	**2**	*mod*$_p$**L**-*hard* ($\exists p$)	*not Datalog*
1,2	**5**	**P**-*hard*	*not Linear Datalog*
1,2,5	**4**	**NL**-*hard*	*not Symmetric Datalog*

Two Conjectures and A Private Suspicion

Strictly for those who believe in beauty ...

Suspicion 1 *Let $\mathcal{B}$ be a core and let* $\mathbf{A}$ *be the idempotent algebra associated to it.*

- *(BJK) If* $\mathrm{var}(\mathbf{A})$ *omits type* $\mathbf{1}$ *then* $\mathrm{CSP}(\mathcal{B})$ *is in* $\mathbf{P}$*;*
- *(Larose,Zádori)* $\mathrm{var}(\mathbf{A})$ *omits types* $\mathbf{1}$ *and* $\mathbf{2}$ *iff* co-$\mathrm{CSP}(\mathcal{B})$ *is in Datalog;*
- $\mathrm{var}(\mathbf{A})$ *omits types* $\mathbf{1}$, $\mathbf{2}$ *and* $\mathbf{5}$ *iff* co-$\mathrm{CSP}(\mathcal{B})$ *is in Linear Datalog* *(bonus: iff* $\mathrm{CSP}(\mathcal{B})$ *in* $\mathbf{NL}$*);*
- $\mathrm{var}(\mathbf{A})$ *omits types* $\mathbf{1}$, $\mathbf{2}$, $\mathbf{4}$ *and* $\mathbf{5}$ *iff* co-$\mathrm{CSP}(\mathcal{B})$ *is in Symmetric Datalog* *(bonus: iff* $\mathrm{CSP}(\mathcal{B})$ *in* $\mathbf{L}$*).*

Some Evidence

Theorem 8 (Larose,Tesson '07)
For two-element structures, everything is as suspected.

Theorem 9 (Bulatov'02-06)
The Larose-Zádori (bounded width) conjecture holds

1. *for all three-element algebras, and*

2. *for all conservative algebras.*

Some More Evidence

Let $\mathcal{B}$ be a core and let $\mathbf{A}$ be its idempotent algebra.

	nec. cond. for var($\mathbf{A}$)	sufficient condition (polymorphism for $\mathcal{B}$)
Datalog	no types **1** and **2**	Totally symmetric idemp. (FV'98)
		2-semilattice (Bulatov'06)
		NU (FV'98; JCC'98)
		Jónsson operations (Barto,Kozik'08)
Lin Dat	no **1**,**2**,**5**	majority (3-ary NU) (Dalmau,K'08)
Sym Dat	no **1**,**2**,**4**,**5**	maj. + Mal'tsev (Dalmau,Larose'08)

How to Put a CSP in Datalog, Vaguely

For a CSP($\mathcal{B}$) instance $\mathcal{A}$, a partial solution with domain $A' \subseteq A$ is a partial map $A' \to B$ that satisfies all constraints involving only elements in A'.

Say that a winning strategy for $(\mathcal{A}, \mathcal{B})$ is a "consistent" family of partial solutions with domains of bounded size.

Fact. For any $\mathcal{B}$, co-CSP($\mathcal{B}$) is definable in Datalog iff we have $\mathcal{A} \to \mathcal{B}$ for each pair $(\mathcal{A}, \mathcal{B})$ with a winning strategy.

Show: if $\mathcal{B}$ has property X (e.g., a given polymorphism) then a winning strategy for $(\mathcal{A}, \mathcal{B})$ implies $\mathcal{A} \to \mathcal{B}$.

Summary I: What We've Seen

1. Forms of CSP: Var-Val, Sat, and Hom.
2. Constraint languages, CSP(Γ), CSP($\mathcal{B}$)
3. Feder-Vardi (Dichotomy) Conjecture
4. Approaches: Graphs, Logic, Algebra
5. Reduction to classification of algebras
6. Algebraic Dichotomy Conjecture
7. Tractability via Few Subpowers and Datalog
8. Fragments of Datalog, TCT types, **L** and **NL**

Summary II: What We Didn't See

- Apologies: There is a lot of very nice (relevant) results that I didn't have time to mention.
- Challenge for UA: further investigate properties of subpowers that might be algorithmically usable (e.g., marry few subpowers and bounded width).
- A look beyond: There exist CSP-related problems where other kind of maths (including other kind of algebra) is naturally used.