Constraints, Symmetry, and Complexity

Andrei Krokhin

Message

Q.: What kind of mathematical structure makes a problem easy or hard?

Message

Q.: What kind of mathematical structure makes a problem easy or hard?

- The Constraint Satisfaction Problem paradigm
  - General enough to include many interesting problems
  - Structured enough to make significant progress (on all problems within a class, not just a single problem)

Message

Q.: What kind of mathematical structure makes a problem easy or hard?

- The Constraint Satisfaction Problem paradigm
- Main Achievement: better understanding of common structure in easy (or hard) problems
  - — focus on appropriate symmetries of solution spaces!
  - — lack of symmetries $\Rightarrow$ hardness
  - — symmetries $\Rightarrow$ efficient algorithms

Message

Q.: What kind of mathematical structure makes a problem easy or hard?

- The Constraint Satisfaction Problem paradigm
- Main Achievement: better understanding of common structure in easy (or hard) problems
  - focus on appropriate symmetries of solution spaces!
  - lack of symmetries $\Rightarrow$ hardness
  - symmetries $\Rightarrow$ efficient algorithms
  - symmetries of higher dimension/arity are important (not just auto- or endomorphisms) $\rightarrow$ **universal algebra**

Message

Q.: What kind of mathematical structure makes a problem easy or hard?

- The Constraint Satisfaction Problem paradigm
- Main Achievement: better understanding of common structure in easy (or hard) problems
  - focus on appropriate symmetries of solution spaces!
  - lack of symmetries $\Rightarrow$ hardness
  - symmetries $\Rightarrow$ efficient algorithms
  - symmetries of higher dimension/arity are important (not just auto- or endomorphisms) $\rightarrow$ **universal algebra**
  - useful way to measure/compare symmetries

Message

Q.: What kind of mathematical structure makes a problem easy or hard?

- The Constraint Satisfaction Problem paradigm
- Main Achievement: better understanding of common structure in easy (or hard) problems
  - focus on appropriate symmetries of solution spaces!
  - lack of symmetries $\Rightarrow$ hardness
  - symmetries $\Rightarrow$ efficient algorithms
  - symmetries of higher dimension/arity are important (not just auto- or endomorphisms) $\rightarrow$ **universal algebra**
  - useful way to measure/compare symmetries
- Long term goal: go beyond CSPs

Constraint Satisfaction Problem (CSP)

Fix finite relational structure $\mathbb{A} = (A; R_1, \ldots, R_n)$ (aka constraint language) where each $R_i \subseteq A^{k_i}$ or $R_i : A^{k_i} \to \{\text{true}, \text{false}\}$.

Constraint Satisfaction Problem (CSP)

Fix finite relational structure $\mathbb{A} = (A; R_1, \ldots, R_n)$ (aka constraint language) where each $R_i \subseteq A^{k_i}$ or $R_i : A^{k_i} \to \{\text{true}, \text{false}\}$.

Definition

An instance of $\mathrm{CSP}(\mathbb{A})$ is a list of constraints over vars V, e.g.

$$R_1(x, y, z), R_1(z, y, w), R_2(z), R_3(x, w), R_3(y, y)$$

where each R_i is from $\mathbb{A}$.

Question: Is there $s : V \to A$ satisfying all constraints?

Constraint Satisfaction Problem (CSP)

Fix finite relational structure $\mathbb{A} = (A; R_1, \ldots, R_n)$ (aka constraint language) where each $R_i \subseteq A^{k_i}$ or $R_i : A^{k_i} \to \{\text{true}, \text{false}\}$.

Definition

An instance of $\mathrm{CSP}(\mathbb{A})$ is a list of constraints over vars V, e.g.

$$R_1(x, y, z), R_1(z, y, w), R_2(z), R_3(x, w), R_3(y, y)$$

where each R_i is from $\mathbb{A}$.

Question: Is there $s : V \to A$ satisfying all constraints?

Other variants, e.g.:

- infinite A (but the instance is still finite)
- nothing (or something other than relations) is fixed
- real-valued functions instead of relations (for optimisation)

Examples and a conjecture

- k-Col: $\mathbb{A} = ([k]; \{\neq\})$

Examples and a conjecture

- k-Col: $\mathbb{A} = ([k]; \{\neq\})$
- k-NAE: $\mathbb{A} = ([k]; \{(a, b, c) \in [k]^3 \mid a \neq b \vee a \neq c \vee b \neq c\})$
  - (essentially) k-colouring for 3-uniform hypergraphs

Examples and a conjecture

- k-Col: $\mathbb{A} = ([k]; \{\neq\})$
- k-NAE: $\mathbb{A} = ([k]; \{(a, b, c) \in [k]^3 \mid a \neq b \vee a \neq c \vee b \neq c\})$
  - — (essentially) k-colouring for 3-uniform hypergraphs
- 3-Sat: $\mathbb{A} = (\{0, 1\}; (x \vee y \vee z), (x \vee y \vee \neg z), \ldots)$
- Horn 3-Sat: as above, each clause has ≤ 1 unneg var

Examples and a conjecture

- k-Col: $\mathbb{A} = ([k]; \{\neq\})$
- k-NAE: $\mathbb{A} = ([k]; \{(a, b, c) \in [k]^3 \mid a \neq b \vee a \neq c \vee b \neq c\})$
  - — (essentially) k-colouring for 3-uniform hypergraphs
- 3-Sat: $\mathbb{A} = (\{0, 1\}; (x \vee y \vee z), (x \vee y \vee \neg z), \ldots)$
- Horn 3-Sat: as above, each clause has ≤ 1 unneg var
- 3-Lin$p$: $\mathbb{A} = (\mathbb{Z}_p;\ x + y + z = 0,\ x + 2y + 3z = 7, \ldots)$

Examples and a conjecture

- k-Col: $\mathbb{A} = ([k]; \{\neq\})$
- k-NAE: $\mathbb{A} = ([k]; \{(a, b, c) \in [k]^3 \mid a \neq b \vee a \neq c \vee b \neq c\})$
  - — (essentially) k-colouring for 3-uniform hypergraphs
- 3-Sat: $\mathbb{A} = (\{0, 1\}; (x \vee y \vee z), (x \vee y \vee \neg z), \ldots)$
- Horn 3-Sat: as above, each clause has ≤ 1 unneg var
- 3-Lin$p$: $\mathbb{A} = (\mathbb{Z}_p;\ x + y + z = 0,\ x + 2y + 3z = 7, \ldots)$
- Unique Games-k: $\mathbb{A} = ([k]; \{(a, \pi(a)) \mid a \in [k]\})$ where π runs through all permutations on $[k]$.

Examples and a conjecture

- k-Col: $\mathbb{A} = ([k]; \{\neq\})$
- k-NAE: $\mathbb{A} = ([k]; \{(a,b,c) \in [k]^3 \mid a \neq b \vee a \neq c \vee b \neq c\})$
  - — (essentially) k-colouring for 3-uniform hypergraphs
- 3-Sat: $\mathbb{A} = (\{0,1\}; (x \vee y \vee z), (x \vee y \vee \neg z), \ldots)$
- Horn 3-Sat: as above, each clause has ≤ 1 unneg var
- 3-Lin$p$: $\mathbb{A} = (\mathbb{Z}_p;\ x+y+z=0,\ x+2y+3z=7, \ldots)$
- Unique Games-k: $\mathbb{A} = ([k]; \{(a, \pi(a)) \mid a \in [k]\})$ where π runs through all permutations on $[k]$.

Conjecture (CSP Dichotomy Conjecture, Feder-Vardi'98)

Every $\mathrm{CSP}(\mathbb{A})$ *is either in* **P** *or* **NP**-*c.*

Some algorithmic tasks about CSPs

Some algorithmic tasks about CSPs

Programme: Classify the complexity of these tasks wrt $\mathbb{A}$

1. **Decision CSP**: Can all constraints be satisfied?
   — Done [Bulatov'17, Zhuk'17]; finer classification — open

Some algorithmic tasks about CSPs

Programme: Classify the complexity of these tasks wrt $\mathbb{A}$

1. **Decision CSP**: Can all constraints be satisfied?
   — Done [Bulatov'17, Zhuk'17]; finer classification — open
2. **Counting CSP**: Count the number of solutions
   — Done [Bulatov'08-13, Dyer, Richerby'10-13, Cai, Chen'12-17]

Some algorithmic tasks about CSPs

Programme: Classify the complexity of these tasks wrt $\mathbb{A}$

1. **Decision CSP**: Can all constraints be satisfied?
   — Done [Bulatov'17, Zhuk'17]; finer classification — open
2. **Counting CSP**: Count the number of solutions
   — Done [Bulatov'08-13, Dyer, Richerby'10-13, Cai, Chen'12-17]
3. **Max CSP**: Find a map satisfying max number of constraints
   — Done [Thapper, Živný'16], [Kolmogorov, AK, Rolínek'17]

Some algorithmic tasks about CSPs

Programme: Classify the complexity of these tasks wrt $\mathbb{A}$

1. **Decision CSP**: Can all constraints be satisfied?
   — Done [Bulatov'17, Zhuk'17]; finer classification — open
2. **Counting CSP**: Count the number of solutions
   — Done [Bulatov'08-13, Dyer, Richerby'10-13, Cai, Chen'12-17]
3. **Max CSP**: Find a map satisfying max number of constraints
   — Done [Thapper, Živný'16], [Kolmogorov, AK, Rolínek'17]
4. **Approx Max CSP**: Satisfy $c \times \mathrm{Opt}$ number of constraints
   — (Essentially) Done [Raghavendra'08]

Some algorithmic tasks about CSPs

Programme: Classify the complexity of these tasks wrt $\mathbb{A}$

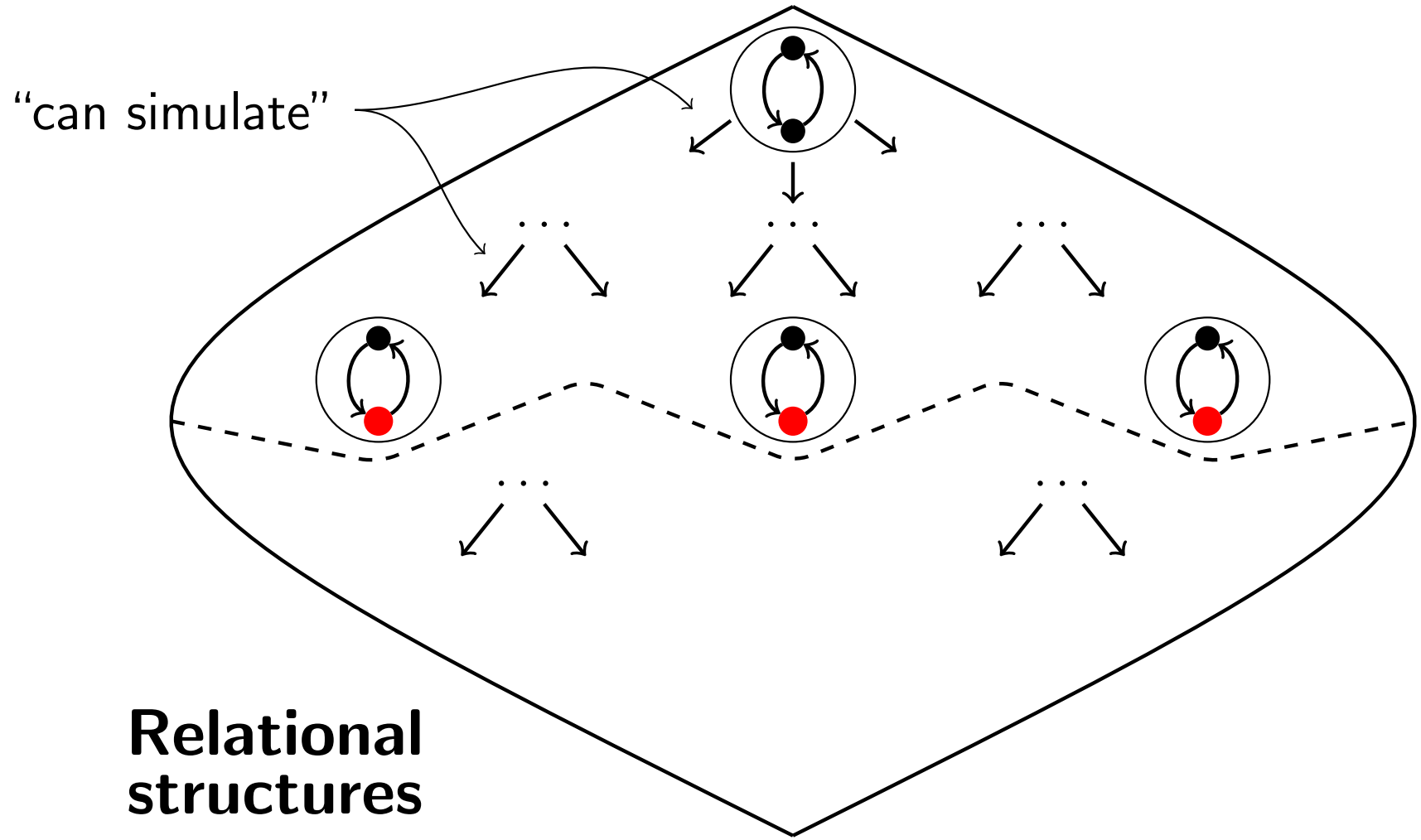
1. **Decision CSP**: Can all constraints be satisfied?
   — Done [Bulatov'17, Zhuk'17]; finer classification — open
2. **Counting CSP**: Count the number of solutions
   — Done [Bulatov'08-13, Dyer, Richerby'10-13, Cai, Chen'12-17]
3. **Max CSP**: Find a map satisfying max number of constraints
   — Done [Thapper, Živný'16], [Kolmogorov, AK, Rolínek'17]
4. **Approx Max CSP**: Satisfy $c \times \mathrm{Opt}$ number of constraints
   — (Essentially) Done [Raghavendra'08]
5. (α, β)-**Approx Max CSP**: assuming β fraction of constraints can be satisfied, find a map satisfying $\geq \alpha$ fraction.
   — Strong results for some (α, β) and $\mathbb{A}$, wide open in general

Some algorithmic tasks about CSPs

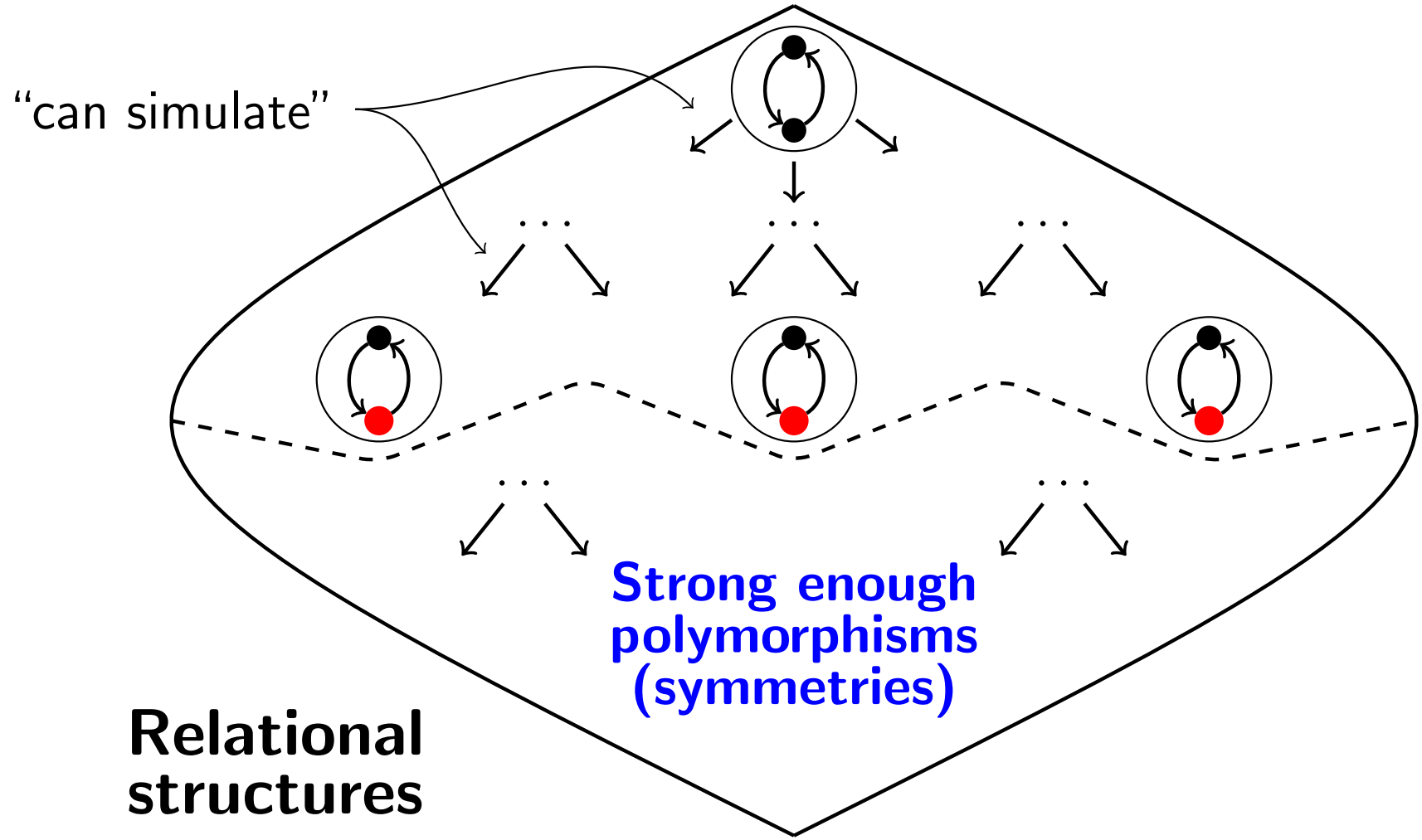
Programme: Classify the complexity of these tasks wrt $\mathbb{A}$

1. **Decision CSP**: Can all constraints be satisfied?
   — Done [Bulatov'17, Zhuk'17]; finer classification — open
2. **Counting CSP**: Count the number of solutions
   — Done [Bulatov'08-13, Dyer, Richerby'10-13, Cai, Chen'12-17]
3. **Max CSP**: Find a map satisfying max number of constraints
   — Done [Thapper, Živný'16], [Kolmogorov, AK, Rolínek'17]
4. **Approx Max CSP**: Satisfy $c \times \mathrm{Opt}$ number of constraints
   — (Essentially) Done [Raghavendra'08]
5. (α, β)-**Approx Max CSP**: assuming β fraction of constraints can be satisfied, find a map satisfying $\geq \alpha$ fraction.
   — Strong results for some (α, β) and $\mathbb{A}$, wide open in general
6. **Promise CSP**: wide open — tomorrow

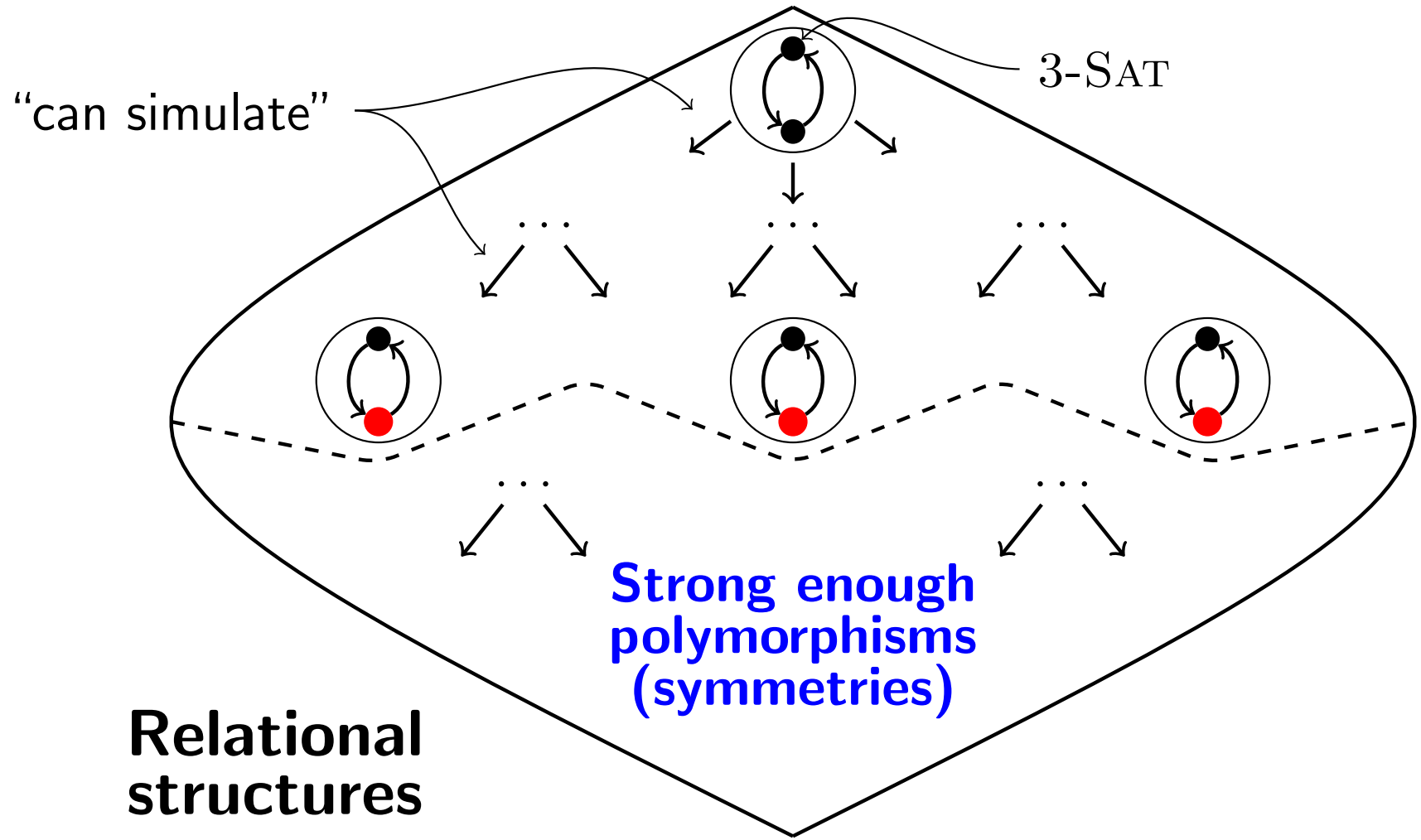
Algebraic approach to CSP (very high-level view)



Algebraic approach to CSP (very high-level view)



Algebraic approach to CSP (very high-level view)



Simulation by example

Three (increasingly more general) levels of simulation:

Simulation by example

Three (increasingly more general) levels of simulation:

1. pp-definitions (= gadgets, same domain)

Simulation by example

Three (increasingly more general) levels of simulation:

1. pp-definitions (= gadgets, same domain) Ex.: Let $\mathbb{A}_1 = (A; R)$, R ternary, and $\mathbb{A}_2 = (A; T, S)$ be s.t.

$$T(x) = \exists z\ R(x, z, x), \quad S(x, y) = \exists z\ R(x, y, z) \wedge R(y, y, x).$$

Simulation by example

Three (increasingly more general) levels of simulation:

1. pp-definitions (= gadgets, same domain) Ex.: Let $\mathbb{A}_1 = (A; R)$, R ternary, and $\mathbb{A}_2 = (A; T, S)$ be s.t.

$$T(x) = \exists z\ R(x, z, x), \quad S(x, y) = \exists z\ R(x, y, z) \wedge R(y, y, x).$$

Then an instance of $\mathrm{CSP}(\mathbb{A}_2)$, say

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

Simulation by example

Three (increasingly more general) levels of simulation:

1. pp-definitions (= gadgets, same domain) Ex.: Let $\mathbb{A}_1 = (A; R)$, R ternary, and $\mathbb{A}_2 = (A; T, S)$ be s.t.

$$T(x) = \exists z\ R(x, z, x), \quad S(x, y) = \exists z\ R(x, y, z) \wedge R(y, y, x).$$

Then an instance of $\mathrm{CSP}(\mathbb{A}_2)$, say

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

can be re-written as an instance of $\mathrm{CSP}(\mathbb{A}_1)$

$$R(y, z, y), R(x, y, w), R(y, y, x)$$

Simulation by example

Three (increasingly more general) levels of simulation:

1. pp-definitions (= gadgets, same domain) Ex.: Let $\mathbb{A}_1 = (A; R)$, R ternary, and $\mathbb{A}_2 = (A; T, S)$ be s.t.

   $$T(x) = \exists z\ R(x, z, x), \quad S(x, y) = \exists z\ R(x, y, z) \wedge R(y, y, x).$$

   Then an instance of $\mathrm{CSP}(\mathbb{A}_2)$, say

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

   can be re-written as an instance of $\mathrm{CSP}(\mathbb{A}_1)$

   $$R(y, z, y), R(x, y, w), R(y, y, x)$$

2. pp-interpretations (gadgets, possibly diff domains)
3. pp-constructions (gadgets + more)

Polymorphisms by example

- Take any 2-SAT instance $(x \vee \overline{y}) \wedge (y \vee \overline{z}) \wedge (y \vee \overline{u}) \wedge (x \vee u)$
- Take any three solutions $\mathbf{a}, \mathbf{b}, \mathbf{c}$ to this instance

Polymorphisms by example

- Take any 2-SAT instance $(x \vee \overline{y}) \wedge (y \vee \overline{z}) \wedge (y \vee \overline{u}) \wedge (x \vee u)$
- Take any three solutions $\mathbf{a}, \mathbf{b}, \mathbf{c}$ to this instance
- Apply the ternary *majority* operation m to $\mathbf{a}, \mathbf{b}, \mathbf{c}$ coordinate-wise (variables ordered here as x, y, z, u)

Polymorphisms by example

- Take any 2-SAT instance $(x \vee \overline{y}) \wedge (y \vee \overline{z}) \wedge (y \vee \overline{u}) \wedge (x \vee u)$
- Take any three solutions $\mathbf{a}, \mathbf{b}, \mathbf{c}$ to this instance
- Apply the ternary *majority* operation m to $\mathbf{a}, \mathbf{b}, \mathbf{c}$ coordinate-wise (variables ordered here as x, y, z, u)

$$
\begin{array}{rcccccc}
 & & m & m & m & m & \\
 & & \downarrow & \downarrow & \downarrow & \downarrow & \\
\mathbf{a} = & (& 1 & 1 & 1 & 0 &) \quad \text{sat} \\
\mathbf{b} = & (& 1 & 1 & 0 & 1 &) \quad \text{sat} \\
\mathbf{c} = & (& 1 & 0 & 0 & 0 &) \quad \text{sat} \\
\hline
m(\mathbf{a}, \mathbf{b}, \mathbf{c}) = & (& 1 & 1 & 0 & 0 &) \quad \text{sat}
\end{array}
$$

Polymorphisms

An operation $f : A^m \to A$ is called a polymorphism of a k-ary relation $R \subseteq A^k$ if for any k-tuples

$$\begin{array}{ccccccc}
 & f & & f & & f & \\
 & \downarrow & & \downarrow & & \downarrow & \\
(& a_{11} & , \ \ldots \ , & & a_{1k} &) & \in R \\
 & \vdots & & \vdots & & \vdots & \vdots \\
(& a_{m1} & , \ \ldots \ , & & a_{mk} &) & \in R \\
\hline
 & & & & & \Downarrow & \\
(& f(a_{11}, \ldots, a_{m1}) & , \ \ldots \ , & & f(a_{1k}, \ldots, a_{mk}) &) & \in R
\end{array}$$

Polymorphisms

An operation $f : A^m \to A$ is called a polymorphism of a k-ary relation $R \subseteq A^k$ if for any k-tuples

$$\begin{array}{ccccccc}
 & f & & f & & f & \\
 & \downarrow & & \downarrow & & \downarrow & \\
(& a_{11} & , \ \ldots \ , & & a_{1k} &) & \in R \\
 & \vdots & & \vdots & & \vdots & \vdots \\
(& a_{m1} & , \ \ldots \ , & & a_{mk} &) & \in R \\
\hline
 & & & & & \Downarrow & \\
(& f(a_{11}, \ldots, a_{m1}) & , \ \ldots \ , & & f(a_{1k}, \ldots, a_{mk}) &) & \in R
\end{array}$$

Call f a polymorphism of $\mathbb{A}$ if it is such for all R in $\mathbb{A}$.
Notation: $\mathrm{Pol}(\mathbb{A})$.

More examples of polymorphisms

1. Let $R = \{\mathbf{r} \in \mathbb{Z}_p^n \mid A\mathbf{r} = \mathbf{b}\}$ be an affine subspace of $\mathbb{Z}_p^n$.

More examples of polymorphisms

1. Let $R = \{\mathbf{r} \in \mathbb{Z}_p^n \mid A\mathbf{r} = \mathbf{b}\}$ be an affine subspace of $\mathbb{Z}_p^n$.
   Then $f(x_1, x_2, x_3) = x_1 - x_2 + x_3$ is a polymorphism of R:

More examples of polymorphisms

1. Let $R = \{\mathbf{r} \in \mathbb{Z}_p^n \mid A\mathbf{r} = \mathbf{b}\}$ be an affine subspace of $\mathbb{Z}_p^n$.
   Then $f(x_1, x_2, x_3) = x_1 - x_2 + x_3$ is a polymorphism of R:
   If $\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3 \in R$, i.e. $A\mathbf{r}_1 = \mathbf{b}, A\mathbf{r}_2 = \mathbf{b}, A\mathbf{r}_3 = \mathbf{b}$, then

More examples of polymorphisms

1. Let $R = \{\mathbf{r} \in \mathbb{Z}_p^n \mid A\mathbf{r} = \mathbf{b}\}$ be an affine subspace of $\mathbb{Z}_p^n$.
   Then $f(x_1, x_2, x_3) = x_1 - x_2 + x_3$ is a polymorphism of R:
   If $\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3 \in R$, i.e. $A\mathbf{r}_1 = \mathbf{b}, A\mathbf{r}_2 = \mathbf{b}, A\mathbf{r}_3 = \mathbf{b}$, then

$$A(\mathbf{r}_1 - \mathbf{r}_2 + \mathbf{r}_3) = A\mathbf{r}_1 - A\mathbf{r}_2 + A\mathbf{r}_3$$

More examples of polymorphisms

1. Let $R = \{\mathbf{r} \in \mathbb{Z}_p^n \mid A\mathbf{r} = \mathbf{b}\}$ be an affine subspace of $\mathbb{Z}_p^n$.

   Then $f(x_1, x_2, x_3) = x_1 - x_2 + x_3$ is a polymorphism of R:

   If $\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3 \in R$, i.e. $A\mathbf{r}_1 = \mathbf{b}, A\mathbf{r}_2 = \mathbf{b}, A\mathbf{r}_3 = \mathbf{b}$, then

   $$A(\mathbf{r}_1 - \mathbf{r}_2 + \mathbf{r}_3) = A\mathbf{r}_1 - A\mathbf{r}_2 + A\mathbf{r}_3 = \mathbf{b} - \mathbf{b} + \mathbf{b} = \mathbf{b}.$$

More examples of polymorphisms

1. Let $R = \{\mathbf{r} \in \mathbb{Z}_p^n \mid A\mathbf{r} = \mathbf{b}\}$ be an affine subspace of $\mathbb{Z}_p^n$.
   Then $f(x_1, x_2, x_3) = x_1 - x_2 + x_3$ is a polymorphism of R:
   If $\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3 \in R$, i.e. $A\mathbf{r}_1 = \mathbf{b}, A\mathbf{r}_2 = \mathbf{b}, A\mathbf{r}_3 = \mathbf{b}$, then

$$A(\mathbf{r}_1 - \mathbf{r}_2 + \mathbf{r}_3) = A\mathbf{r}_1 - A\mathbf{r}_2 + A\mathbf{r}_3 = \mathbf{b} - \mathbf{b} + \mathbf{b} = \mathbf{b}.$$

2. If $\mathbb{A} = (A, E)$ is a digraph then f is a polymorphism of $\mathbb{A}$ if

$$(a_1, b_1), \ldots, (a_n, b_n) \in E \Rightarrow (f(a_1, \ldots, a_n), f(b_1, \ldots, b_n)) \in E.$$

More examples of polymorphisms

1. Let $R = \{\mathbf{r} \in \mathbb{Z}_p^n \mid A\mathbf{r} = \mathbf{b}\}$ be an affine subspace of $\mathbb{Z}_p^n$.
   Then $f(x_1, x_2, x_3) = x_1 - x_2 + x_3$ is a polymorphism of R:
   If $\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3 \in R$, i.e. $A\mathbf{r}_1 = \mathbf{b}, A\mathbf{r}_2 = \mathbf{b}, A\mathbf{r}_3 = \mathbf{b}$, then

$$A(\mathbf{r}_1 - \mathbf{r}_2 + \mathbf{r}_3) = A\mathbf{r}_1 - A\mathbf{r}_2 + A\mathbf{r}_3 = \mathbf{b} - \mathbf{b} + \mathbf{b} = \mathbf{b}.$$

2. If $\mathbb{A} = (A, E)$ is a digraph then f is a polymorphism of $\mathbb{A}$ if

   $(a_1, b_1), \ldots, (a_n, b_n) \in E \Rightarrow (f(a_1, \ldots, a_n), f(b_1, \ldots, b_n)) \in E$.

3. Every polymorphism of 3-SAT is a projection (aka dictator),
   i.e. $f(x_1, \ldots, x_n) = x_i$ for some i.

More examples of polymorphisms

1. Let $R = \{\mathbf{r} \in \mathbb{Z}_p^n \mid A\mathbf{r} = \mathbf{b}\}$ be an affine subspace of $\mathbb{Z}_p^n$.
   Then $f(x_1, x_2, x_3) = x_1 - x_2 + x_3$ is a polymorphism of R:
   If $\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3 \in R$, i.e. $A\mathbf{r}_1 = \mathbf{b}, A\mathbf{r}_2 = \mathbf{b}, A\mathbf{r}_3 = \mathbf{b}$, then

   $$A(\mathbf{r}_1 - \mathbf{r}_2 + \mathbf{r}_3) = A\mathbf{r}_1 - A\mathbf{r}_2 + A\mathbf{r}_3 = \mathbf{b} - \mathbf{b} + \mathbf{b} = \mathbf{b}.$$

2. If $\mathbb{A} = (A, E)$ is a digraph then f is a polymorphism of $\mathbb{A}$ if

   $$(a_1, b_1), \ldots, (a_n, b_n) \in E \Rightarrow (f(a_1, \ldots, a_n), f(b_1, \ldots, b_n)) \in E.$$

3. Every polymorphism of 3-SAT is a projection (aka dictator),
   i.e. $f(x_1, \ldots, x_n) = x_i$ for some i.
4. Every polymorphism of 3-COL is of the form
   $f(x_1, \ldots, x_n) = \pi(x_i)$ for some $i \leq n$ and permutation π

Modern proof of Schaefer's classification for Boolean CSPs

Problem	Polymorphism	Complexity
0-valid	constant 0	Easy
1-valid	constant 1	Easy
2-SAT	majority	Easy
HORN-SAT	min	Easy
DUAL H-SAT	max	Easy
3-LIN2	$x - y + z$	Easy
3-SAT	only projections	Hard

Polymorphisms as symmetries

Objects capturing the symmetry of $\mathrm{CSP}(\mathbb{A})$:

- $\mathrm{Aut}(\mathbb{A}) = \{f : \mathbb{A} \to \mathbb{A}$ automorph$\}$ automorphism group
- $\mathrm{End}(\mathbb{A}) = \{f : \mathbb{A} \to \mathbb{A}$ homomorph$\}$ endomorphism monoid
- $\mathrm{Pol}(\mathbb{A}) = \{f : \mathbb{A}^n \to \mathbb{A}$ homomorph$\}$ polymorphism clone

Polymorphisms as symmetries

Objects capturing the symmetry of $\mathrm{CSP}(\mathbb{A})$:

- $\mathrm{Aut}(\mathbb{A}) = \{f : \mathbb{A} \to \mathbb{A}$ automorph$\}$ automorphism group
- $\mathrm{End}(\mathbb{A}) = \{f : \mathbb{A} \to \mathbb{A}$ homomorph$\}$ endomorphism monoid
- $\mathrm{Pol}(\mathbb{A}) = \{f : \mathbb{A}^n \to \mathbb{A}$ homomorph$\}$ polymorphism clone

Clone = set of multivariate functions on A which

1. is closed under composition, and
2. contains all projections/dictators

Polymorphisms as symmetries

Objects capturing the symmetry of $\mathrm{CSP}(\mathbb{A})$:

- $\mathrm{Aut}(\mathbb{A}) = \{f : \mathbb{A} \to \mathbb{A} \text{ automorph}\}$ automorphism group
- $\mathrm{End}(\mathbb{A}) = \{f : \mathbb{A} \to \mathbb{A} \text{ homomorph}\}$ endomorphism monoid
- $\mathrm{Pol}(\mathbb{A}) = \{f : \mathbb{A}^n \to \mathbb{A} \text{ homomorph}\}$ polymorphism clone

Clone = set of multivariate functions on A which

1. is closed under composition, and
2. contains all projections/dictators

Example: *trivial* clone $\mathcal{T}$, consisting of all projections.

Polymorphisms as symmetries

Objects capturing the symmetry of $\mathrm{CSP}(\mathbb{A})$:

- $\mathrm{Aut}(\mathbb{A}) = \{f : \mathbb{A} \to \mathbb{A}$ automorph$\}$ automorphism group
- $\mathrm{End}(\mathbb{A}) = \{f : \mathbb{A} \to \mathbb{A}$ homomorph$\}$ endomorphism monoid
- $\mathrm{Pol}(\mathbb{A}) = \{f : \mathbb{A}^n \to \mathbb{A}$ homomorph$\}$ polymorphism clone

Clone = set of multivariate functions on A which

1. is closed under composition, and
2. contains all projections/dictators

Example: *trivial* clone $\mathcal{T}$, consisting of all projections.

- $\mathrm{Aut}(\mathbb{A})$, $\mathrm{End}(\mathbb{A})$ - no info about the complexity of $\mathrm{CSP}(\mathbb{A})$.
- $\mathrm{Pol}(\mathbb{A})$ - a lot of info.

Simulation vs. polymorphisms

Theorem (Birkhoff'35; Geiger'68; Bodnarchuk et al.'69; Bodirsky; Willard; Barto, Opršal,Pinsker'18)

Simulation vs. polymorphisms

Theorem (Birkhoff'35; Geiger'68; Bodnarchuk et al.'69; Bodirsky; Willard; Barto, Opršal,Pinsker'18)

- $\mathbb{A}$ *pp-defines* $\mathbb{B}$ *iff* $\mathrm{Pol}(\mathbb{A}) \subseteq \mathrm{Pol}(\mathbb{B})$.

Simulation vs. polymorphisms

Theorem (Birkhoff'35; Geiger'68; Bodnarchuk et al.'69; Bodirsky; Willard; Barto, Opršal,Pinsker'18)

- $\mathbb{A}$ *pp-defines* $\mathbb{B}$ *iff* $\mathrm{Pol}(\mathbb{A}) \subseteq \mathrm{Pol}(\mathbb{B})$.
- $\mathbb{A}$ *pp-interprets* $\mathbb{B}$ *iff* $\mathrm{Pol}(\mathbb{A}) \to \mathrm{Pol}(\mathbb{B})$ *(homomorphism).*

Simulation vs. polymorphisms

Theorem (Birkhoff'35; Geiger'68; Bodnarchuk et al.'69; Bodirsky; Willard; Barto, Opršal,Pinsker'18)

- $\mathbb{A}$ *pp-defines* $\mathbb{B}$ *iff* $\mathrm{Pol}(\mathbb{A}) \subseteq \mathrm{Pol}(\mathbb{B})$.
- $\mathbb{A}$ *pp-interprets* $\mathbb{B}$ *iff* $\mathrm{Pol}(\mathbb{A}) \to \mathrm{Pol}(\mathbb{B})$ *(homomorphism).*
- $\mathbb{A}$ *pp-constructs* $\mathbb{B}$ *iff* $\mathrm{Pol}(\mathbb{A}) \dashrightarrow \mathrm{Pol}(\mathbb{B})$ *(height-1 homo).*

Simulation vs. polymorphisms

Theorem (Birkhoff'35; Geiger'68; Bodnarchuk et al.'69; Bodirsky; Willard; Barto, Opršal,Pinsker'18)

- $\mathbb{A}$ *pp-defines* $\mathbb{B}$ *iff* $\mathrm{Pol}(\mathbb{A}) \subseteq \mathrm{Pol}(\mathbb{B})$.
- $\mathbb{A}$ *pp-interprets* $\mathbb{B}$ *iff* $\mathrm{Pol}(\mathbb{A}) \to \mathrm{Pol}(\mathbb{B})$ *(homomorphism).*
- $\mathbb{A}$ *pp-constructs* $\mathbb{B}$ *iff* $\mathrm{Pol}(\mathbb{A}) \dashrightarrow \mathrm{Pol}(\mathbb{B})$ *(height-1 homo).*

Remarks:

- Proof constructive $\Rightarrow$ generic reduction $\mathrm{CSP}(\mathbb{B}) \rightsquigarrow \mathrm{CSP}(\mathbb{A})$

Simulation vs. polymorphisms

Theorem (Birkhoff'35; Geiger'68; Bodnarchuk et al.'69; Bodirsky; Willard; Barto, Opršal,Pinsker'18)

- $\mathbb{A}$ *pp-defines* $\mathbb{B}$ *iff* $\mathrm{Pol}(\mathbb{A}) \subseteq \mathrm{Pol}(\mathbb{B})$.
- $\mathbb{A}$ *pp-interprets* $\mathbb{B}$ *iff* $\mathrm{Pol}(\mathbb{A}) \to \mathrm{Pol}(\mathbb{B})$ *(homomorphism).*
- $\mathbb{A}$ *pp-constructs* $\mathbb{B}$ *iff* $\mathrm{Pol}(\mathbb{A}) \dashrightarrow \mathrm{Pol}(\mathbb{B})$ *(height-1 homo).*

Remarks:

- Proof constructive $\Rightarrow$ generic reduction $\mathrm{CSP}(\mathbb{B}) \rightsquigarrow \mathrm{CSP}(\mathbb{A})$
- $\xi : \mathrm{Pol}(\mathbb{A}) \to \mathrm{Pol}(\mathbb{B})$ iff it "preserves equations/identities"
  - This allows applications of deep structural universal algebra

Simulation vs. polymorphisms

Theorem (Birkhoff'35; Geiger'68; Bodnarchuk et al.'69; Bodirsky; Willard; Barto, Opršal,Pinsker'18)

- $\mathbb{A}$ *pp-defines* $\mathbb{B}$ *iff* $\mathrm{Pol}(\mathbb{A}) \subseteq \mathrm{Pol}(\mathbb{B})$.
- $\mathbb{A}$ *pp-interprets* $\mathbb{B}$ *iff* $\mathrm{Pol}(\mathbb{A}) \to \mathrm{Pol}(\mathbb{B})$ *(homomorphism).*
- $\mathbb{A}$ *pp-constructs* $\mathbb{B}$ *iff* $\mathrm{Pol}(\mathbb{A}) \dashrightarrow \mathrm{Pol}(\mathbb{B})$ *(height-1 homo).*

Remarks:

- Proof constructive $\Rightarrow$ generic reduction $\mathrm{CSP}(\mathbb{B}) \rightsquigarrow \mathrm{CSP}(\mathbb{A})$
- $\xi : \mathrm{Pol}(\mathbb{A}) \to \mathrm{Pol}(\mathbb{B})$ iff it "preserves equations/identities"
  - This allows applications of deep structural universal algebra
- $\xi : \mathrm{Pol}(\mathbb{A}) \dashrightarrow \mathrm{Pol}(\mathbb{B})$ iff it "preserves ... of height 1"
  - Not used in resolving Dichotomy Conj, but very important

Equations/identities

- Identities = functional equations

Equations/identities

- Identities = functional equations
- Ex.: A (ternary) majority operation is one satisfying identities $m(x, x, y) = m(x, y, x) = m(y, x, x) = x$.

Equations/identities

- Identities = functional equations
- Ex.: A (ternary) majority operation is one satisfying identities

  $m(x, x, y) = m(x, y, x) = m(y, x, x) = x$.
- A random identity: $g(f(x, y), z, x)) = g(x, y, y))$

Equations/identities

- Identities = functional equations
- Ex.: A (ternary) majority operation is one satisfying identities $m(x,x,y) = m(x,y,x) = m(y,x,x) = x$.
- A random identity: $g(f(x,y),z,x)) = g(x,y,y))$
- Height 1 identity: exactly 1 function symbol on both sides, e.g. $f(x,y) = f(y,x)$, but not $f(x,f(y,z)) = f(f(x,y),z)$

Equations/identities

- Identities = functional equations
- Ex.: A (ternary) majority operation is one satisfying identities $m(x,x,y) = m(x,y,x) = m(y,x,x) = x$.
- A random identity: $g(f(x,y),z,x)) = g(x,y,y))$
- Height 1 identity: exactly 1 function symbol on both sides, e.g. $f(x,y) = f(y,x)$, but not $f(x,f(y,z)) = f(f(x,y),z)$
- Can speak about an identity being satisfied in $\mathrm{Pol}(\mathbb{A})$

Equations/identities

- Identities = functional equations
- Ex.: A (ternary) majority operation is one satisfying identities $m(x,x,y) = m(x,y,x) = m(y,x,x) = x$.
- A random identity: $g(f(x,y),z,x)) = g(x,y,y))$
- Height 1 identity: exactly 1 function symbol on both sides, e.g. $f(x,y) = f(y,x)$, but not $f(x,f(y,z)) = f(f(x,y),z)$
- Can speak about an identity being satisfied in $\mathrm{Pol}(\mathbb{A})$
- Can consider systems of identities

Equations/identities

- Identities = functional equations
- Ex.: A (ternary) majority operation is one satisfying identities
  $m(x,x,y) = m(x,y,x) = m(y,x,x) = x$.
- A random identity: $g(f(x,y),z,x)) = g(x,y,y))$
- Height 1 identity: exactly 1 function symbol on both sides,
  e.g. $f(x,y) = f(y,x)$, but not $f(x,f(y,z)) = f(f(x,y),z)$
- Can speak about an identity being satisfied in $\mathrm{Pol}(\mathbb{A})$
- Can consider systems of identities
- Compare
  - $f(x_1,x_2,x_3) = f(x_1,x_3,x_2)$ trivial symmetry
  - $f(x_1,x_2,x_3) = f(x_2,x_3,x_1)$ non-trivial symmetry

Preserving identities

- $\mathrm{Pol}(\mathbb{A}) \to \mathrm{Pol}(\mathbb{B})$ means
  - "each system of identities satisfied in $\mathrm{Pol}(\mathbb{A})$ is also satisfied in $\mathrm{Pol}(\mathbb{B})$", or

Preserving identities

- $\mathrm{Pol}(\mathbb{A}) \to \mathrm{Pol}(\mathbb{B})$ means
  - "each system of identities satisfied in $\mathrm{Pol}(\mathbb{A})$ is also satisfied in $\mathrm{Pol}(\mathbb{B})$", or
  - "$\mathbb{B}$ has more symmetries than $\mathbb{A}$" - in a qualitative way

Preserving identities

- $\mathrm{Pol}(\mathbb{A}) \to \mathrm{Pol}(\mathbb{B})$ means
  - "each system of identities satisfied in $\mathrm{Pol}(\mathbb{A})$ is also satisfied in $\mathrm{Pol}(\mathbb{B})$", or
  - "$\mathbb{B}$ has more symmetries than $\mathbb{A}$" - in a qualitative way
- Ex.: If $\mathrm{Pol}(\mathbb{A})$ has a commutative (or associate, or majority) operation then so does $\mathrm{Pol}(\mathbb{B})$

Preserving identities

- $\mathrm{Pol}(\mathbb{A}) \to \mathrm{Pol}(\mathbb{B})$ means
  - — "each system of identities satisfied in $\mathrm{Pol}(\mathbb{A})$ is also satisfied in $\mathrm{Pol}(\mathbb{B})$", or
  - — "$\mathbb{B}$ has more symmetries than $\mathbb{A}$" - in a qualitative way
- Ex.: If $\mathrm{Pol}(\mathbb{A})$ has a commutative (or associate, or majority) operation then so does $\mathrm{Pol}(\mathbb{B})$

Systems of (h1) identites sat in $\mathrm{Pol}(\mathbb{A})$ = measure of symmetry.

Tractability conjecture

Theorem

If $\mathrm{Pol}(\mathbb{A}) \to \mathcal{T}$ *then* $\mathrm{CSP}(\mathbb{A})$ *is* **NP***-complete.*

Tractability conjecture

Theorem

If $\mathrm{Pol}(\mathbb{A}) \to \mathcal{T}$ *then* $\mathrm{CSP}(\mathbb{A})$ *is* **NP***-complete.*

Proof. If $\mathbb{B}$ is 3-SAT then $\mathrm{Pol}(\mathbb{B}) = \mathcal{T}$, so $\mathbb{A}$ can simulate (pp-interpret) 3-SAT, and hence 3-SAT reduces to $\mathrm{CSP}(\mathbb{A})$.

Tractability conjecture

Theorem

If $\mathrm{Pol}(\mathbb{A}) \to \mathcal{T}$ *then* $\mathrm{CSP}(\mathbb{A})$ *is* **NP***-complete.*

Proof. If $\mathbb{B}$ is 3-SAT then $\mathrm{Pol}(\mathbb{B}) = \mathcal{T}$, so $\mathbb{A}$ can simulate (pp-interpret) 3-SAT, and hence 3-SAT reduces to $\mathrm{CSP}(\mathbb{A})$.

Note:

- Identities satisfied in $\mathcal{T}$ are trivial - satisfied in each $\mathrm{Pol}(\mathbb{A})$.
- If $\mathrm{Pol}(\mathbb{A}) \to \mathcal{T}$ then $\mathrm{Pol}(\mathbb{A})$ satisfies only trivial identities.

Tractability conjecture

Theorem

If $\mathrm{Pol}(\mathbb{A}) \to \mathcal{T}$ *then* $\mathrm{CSP}(\mathbb{A})$ *is* **NP***-complete.*

Proof. If $\mathbb{B}$ is 3-SAT then $\mathrm{Pol}(\mathbb{B}) = \mathcal{T}$, so $\mathbb{A}$ can simulate (pp-interpret) 3-SAT, and hence 3-SAT reduces to $\mathrm{CSP}(\mathbb{A})$.

Note:

- Identities satisfied in $\mathcal{T}$ are trivial - satisfied in each $\mathrm{Pol}(\mathbb{A})$.
- If $\mathrm{Pol}(\mathbb{A}) \to \mathcal{T}$ then $\mathrm{Pol}(\mathbb{A})$ satisfies only trivial identities.

Conjecture (CSP Tractability Conjecture; Bulatov, Jeavons, AK'05; many others)

If $\mathrm{Pol}(\mathbb{A}) \not\to \mathcal{T}$ *then* $\mathrm{CSP}(\mathbb{A})$ *is in* **P***.*

Tractability conjecture

Theorem

If $\mathrm{Pol}(\mathbb{A}) \to \mathcal{T}$ *then* $\mathrm{CSP}(\mathbb{A})$ *is* **NP***-complete.*

Proof. If $\mathbb{B}$ is 3-SAT then $\mathrm{Pol}(\mathbb{B}) = \mathcal{T}$, so $\mathbb{A}$ can simulate (pp-interpret) 3-SAT, and hence 3-SAT reduces to $\mathrm{CSP}(\mathbb{A})$.

Note:

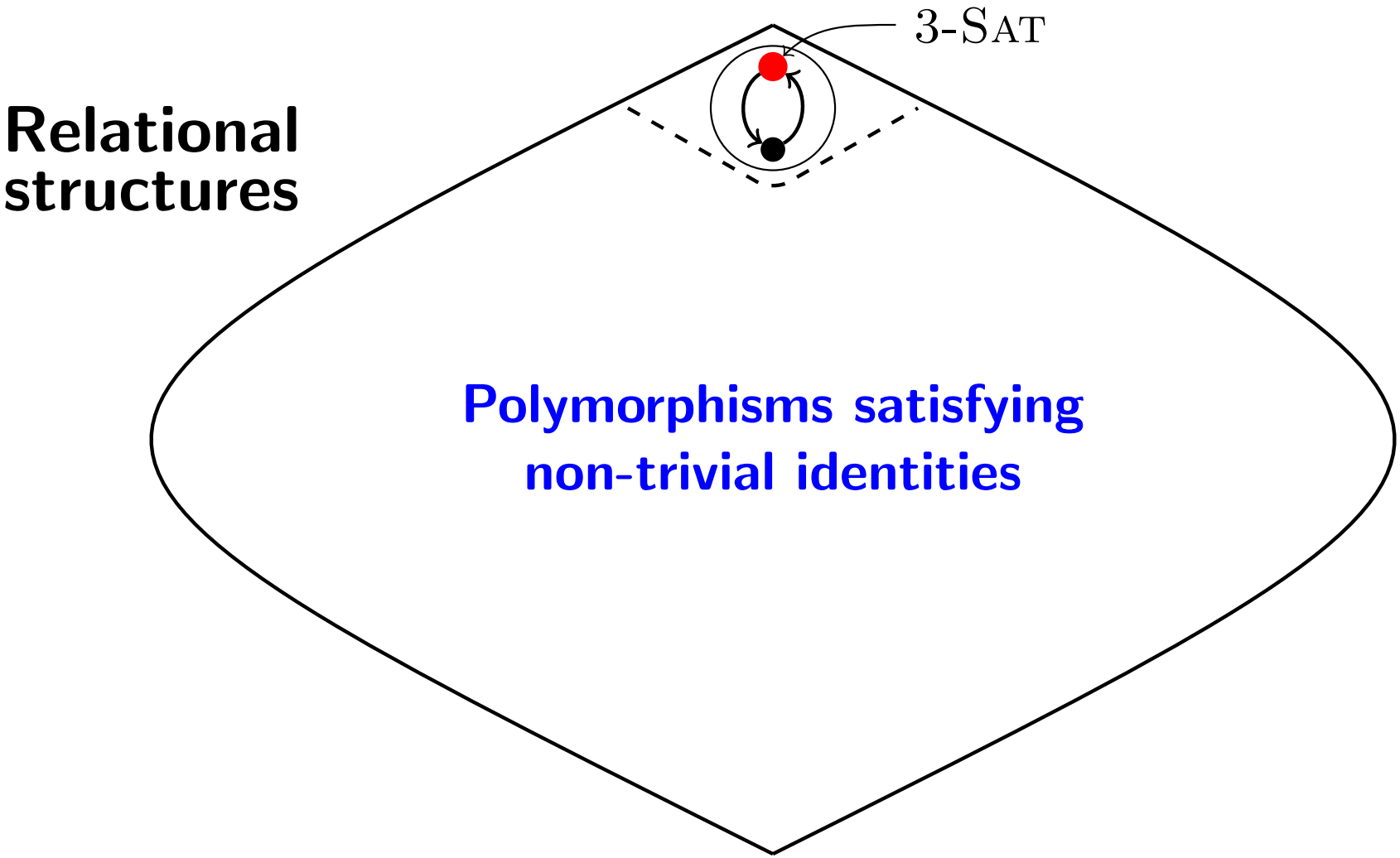
- Identities satisfied in $\mathcal{T}$ are trivial - satisfied in each $\mathrm{Pol}(\mathbb{A})$.
- If $\mathrm{Pol}(\mathbb{A}) \to \mathcal{T}$ then $\mathrm{Pol}(\mathbb{A})$ satisfies only trivial identities.

Conjecture (CSP Tractability Conjecture; Bulatov, Jeavons, AK'05; many others)

If $\mathrm{Pol}(\mathbb{A}) \not\to \mathcal{T}$ *then* $\mathrm{CSP}(\mathbb{A})$ *is in* **P***.*

- If $\mathrm{Pol}(\mathbb{A}) \not\to \mathcal{T}$, $\mathrm{Pol}(\mathbb{A})$ satisfies *some* non-trivial identities.

Algebraic dichotomy (picture)



Non-trivial symmetries $\Rightarrow$ strong symmetries

Theorem

TFAE:

1. $\mathrm{Pol}(\mathbb{A}) \not\to \mathcal{T}$, *i.e.* $\mathrm{Pol}(\mathbb{A})$ *satisfies some non-trivial identities.*

Non-trivial symmetries $\Rightarrow$ strong symmetries

Theorem

TFAE:

1. $\mathrm{Pol}(\mathbb{A}) \not\to \mathcal{T}$, *i.e.* $\mathrm{Pol}(\mathbb{A})$ *satisfies some non-trivial identities.*
2. $\mathbb{A}$ *has a weak near-unanimity polym'm [Mároti,McKenzie'08]*

$$f(y, x, \ldots, x, x) = f(x, y, \ldots, x, x) = \ldots = f(x, x, \ldots, x, y)$$

Non-trivial symmetries $\Rightarrow$ strong symmetries

Theorem

TFAE:

1. $\mathrm{Pol}(\mathbb{A}) \not\to \mathcal{T}$, *i.e.* $\mathrm{Pol}(\mathbb{A})$ *satisfies some non-trivial identities.*
2. $\mathbb{A}$ *has a weak near-unanimity polym'm [Mároti,McKenzie'08]*

$$f(y, x, \ldots, x, x) = f(x, y, \ldots, x, x) = \ldots = f(x, x, \ldots, x, y)$$

3. $\mathbb{A}$ *has a cyclic polymorphism [Barto,Kozik'12]*

$$f(x_1, x_2, x_3, \ldots, x_n) = f(x_2, x_3, \ldots, x_n, x_1)$$

Non-trivial symmetries $\Rightarrow$ strong symmetries

Theorem

TFAE:

1. $\mathrm{Pol}(\mathbb{A}) \not\to \mathcal{T}$, *i.e.* $\mathrm{Pol}(\mathbb{A})$ *satisfies some non-trivial identities.*
2. $\mathbb{A}$ *has a weak near-unanimity polym'm [Mároti,McKenzie'08]*

$$f(y, x, \ldots, x, x) = f(x, y, \ldots, x, x) = \ldots = f(x, x, \ldots, x, y)$$

3. $\mathbb{A}$ *has a cyclic polymorphism [Barto,Kozik'12]*

$$f(x_1, x_2, x_3, \ldots, x_n) = f(x_2, x_3, \ldots, x_n, x_1)$$

4. $\mathbb{A}$ *has a Siggers polymorphism [Siggers'09,KMM'14]*

$$f(r, a, r, e) = f(a, r, e, a)$$

Efficient Algorithms for CSPs

Constraint propagation: given a CSP instance, repeatedly derive new constraints

Efficient Algorithms for CSPs

Constraint propagation: given a CSP instance, repeatedly derive new constraints

If want to work in poly-time, need to

Efficient Algorithms for CSPs

Constraint propagation: given a CSP instance, repeatedly derive new constraints

If want to work in poly-time, need to

1. bound the size of subinstances used for derivation,

Efficient Algorithms for CSPs

Constraint propagation: given a CSP instance, repeatedly derive new constraints
If want to work in poly-time, need to

1. bound the size of subinstances used for derivation,
2. or use a compact representation for derived constraints

The power of each approach is well understood.

Efficient Algorithms for CSPs

Constraint propagation: given a CSP instance, repeatedly derive new constraints
If want to work in poly-time, need to

1. bound the size of subinstances used for derivation,
2. or use a compact representation for derived constraints

The power of each approach is well understood.

How polymorphisms are used:

- combine (partial) solutions to produce a better one

Efficient Algorithms for CSPs

Constraint propagation: given a CSP instance, repeatedly derive new constraints
If want to work in poly-time, need to

1. bound the size of subinstances used for derivation,
2. or use a compact representation for derived constraints

The power of each approach is well understood.

How polymorphisms are used:

- combine (partial) solutions to produce a better one
- directly - algorithm actually combines solutions (Approach 2)
- indirectly - to prove correctness (Approach 1)

Efficient Algorithms for CSPs

Constraint propagation: given a CSP instance, repeatedly derive new constraints
If want to work in poly-time, need to

1. bound the size of subinstances used for derivation,
2. or use a compact representation for derived constraints

The power of each approach is well understood.

How polymorphisms are used:

- combine (partial) solutions to produce a better one
- directly - algorithm actually combines solutions (Approach 2)
- indirectly - to prove correctness (Approach 1)
- identities say which algorithm can be used

The "few subpowers" algorithm - Approach 2

Can solve a system of linear equations over $\mathbb{Z}_p$ as follows:

The "few subpowers" algorithm - Approach 2

Can solve a system of linear equations over $\mathbb{Z}_p$ as follows:

1. Let B_i be the basis for solution set of the first i equations

The "few subpowers" algorithm - Approach 2

Can solve a system of linear equations over $\mathbb{Z}_p$ as follows:

1. Let B_i be the basis for solution set of the first i equations
2. Can find B_i from B_{i-1} and the i-th equation

The "few subpowers" algorithm - Approach 2

Can solve a system of linear equations over $\mathbb{Z}_p$ as follows:

1. Let B_i be the basis for solution set of the first i equations
2. Can find B_i from B_{i-1} and the i-th equation
3. Size of B_i is bounded by n (the number of variabless)
4. At the end, have a compact representation for all solutions

The "few subpowers" algorithm - Approach 2

Can solve a system of linear equations over $\mathbb{Z}_p$ as follows:

1. Let B_i be the basis for solution set of the first i equations
2. Can find B_i from B_{i-1} and the i-th equation
3. Size of B_i is bounded by n (the number of variabless)
4. At the end, have a compact representation for all solutions

The above can be extended to work with general constraints

The "few subpowers" algorithm - Approach 2

Can solve a system of linear equations over $\mathbb{Z}_p$ as follows:

1. Let B_i be the basis for solution set of the first i equations
2. Can find B_i from B_{i-1} and the i-th equation
3. Size of B_i is bounded by n (the number of variabless)
4. At the end, have a compact representation for all solutions

The above can be extended to work with general constraints

(a) B_i can be a "generating set", need it to be small ($O(n^c)$)

(b) need to be able perform the above item (2) efficiently

The "few subpowers" algorithm - Approach 2

Can solve a system of linear equations over $\mathbb{Z}_p$ as follows:

1. Let B_i be the basis for solution set of the first i equations
2. Can find B_i from B_{i-1} and the i-th equation
3. Size of B_i is bounded by n (the number of variabless)
4. At the end, have a compact representation for all solutions

The above can be extended to work with general constraints

(a) B_i can be a "generating set", need it to be small ($O(n^c)$)

(b) need to be able perform the above item (2) efficiently

Theorem (Idziak, Markovic, McKenzie, Valeriote, Willard'10)

Certain identities sat in $\mathrm{Pol}(\mathbb{A}) \Leftrightarrow$ *(a). If have (a), can do (b).*

Local consistency algorithm - Approach 1

Local consistency algorithm - Approach 1

Roughly: Perform local propagation until stable. If not refuted, return "sat".

Local consistency algorithm - Approach 1

Roughly: Perform local propagation until stable. If not refuted, return "sat". (Standard algorithm for 2-SAT and HORN 3-SAT).

Local consistency algorithm - Approach 1

Roughly: Perform local propagation until stable. If not refuted, return "sat". (Standard algorithm for 2-SAT and HORN 3-SAT).

More precisely:

- Fix integers $k \leq \ell$.
- (k, ℓ)-algorithm: Repeatedly derive the strongest constraints on k vars by considering ℓ vars at a time.

Local consistency algorithm - Approach 1

Roughly: Perform local propagation until stable. If not refuted, return "sat". (Standard algorithm for 2-SAT and HORN 3-SAT).

More precisely:

- Fix integers $k \leq \ell$.
- (k, ℓ)-algorithm: Repeatedly derive the strongest constraints on k vars by considering ℓ vars at a time.
- If the instance is refuted, output "no". Otherwise, "yes".
- "No" answers are always correct.

Local consistency algorithm - Approach 1

Roughly: Perform local propagation until stable. If not refuted, return "sat". (Standard algorithm for 2-SAT and HORN 3-SAT).

More precisely:

- Fix integers $k \leq \ell$.
- (k, ℓ)-algorithm: Repeatedly derive the strongest constraints on k vars by considering ℓ vars at a time.
- If the instance is refuted, output "no". Otherwise, "yes".
- "No" answers are always correct.
- If "yes" answers are also correct for each instance of $\mathrm{CSP}(\mathbb{A})$, we say that $\mathbb{A}$ has width (k, ℓ).
- if some (k, ℓ) work for $\mathbb{A}$, say $\mathbb{A}$ has bounded width.

Local consistency algorithm - Approach 1

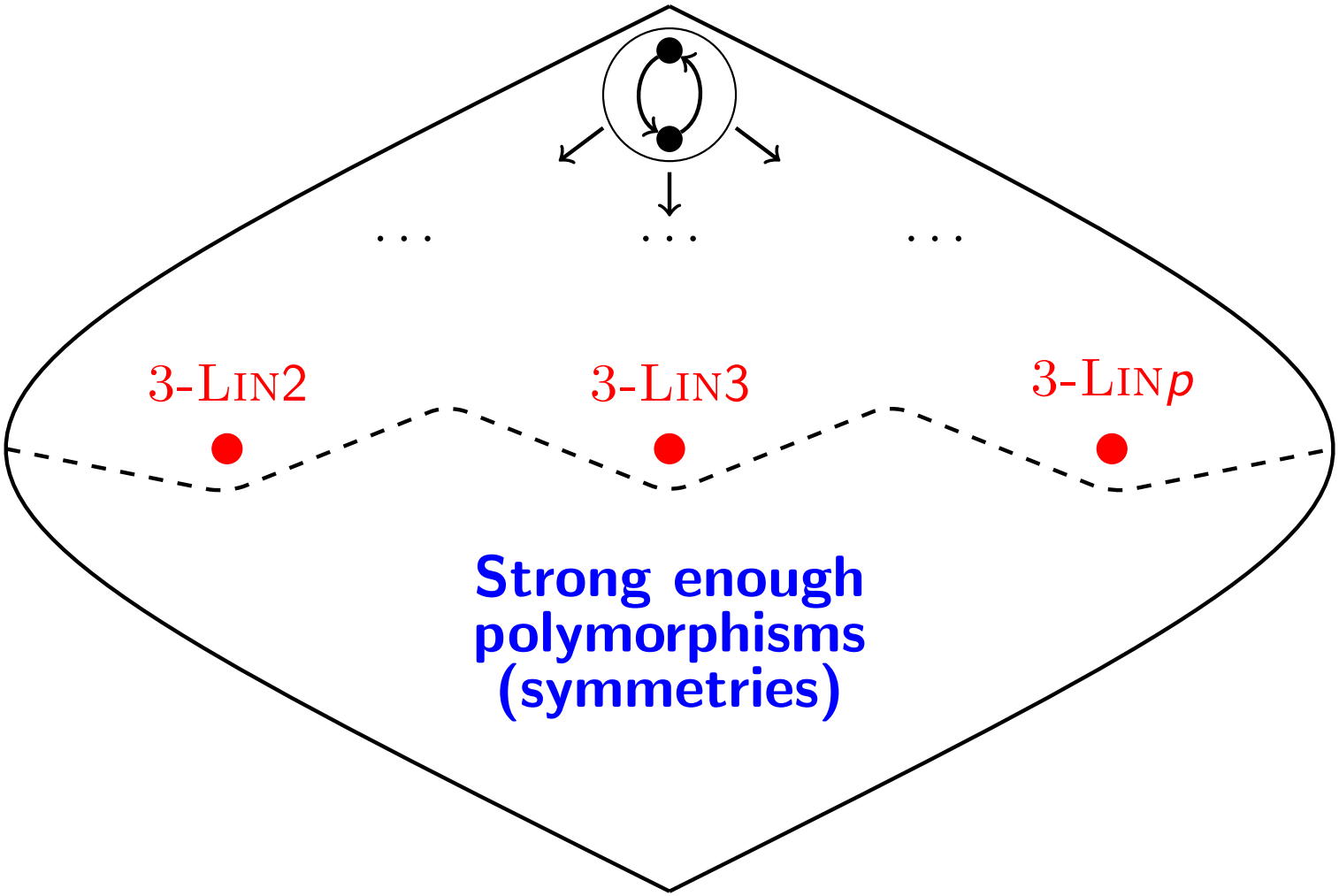
Roughly: Perform local propagation until stable. If not refuted, return "sat". (Standard algorithm for 2-SAT and HORN 3-SAT).

More precisely:

- Fix integers $k \leq \ell$.
- (k, ℓ)-algorithm: Repeatedly derive the strongest constraints on k vars by considering ℓ vars at a time.
- If the instance is refuted, output "no". Otherwise, "yes".
- "No" answers are always correct.
- If "yes" answers are also correct for each instance of $\mathrm{CSP}(\mathbb{A})$, we say that $\mathbb{A}$ has width (k, ℓ).
- if some (k, ℓ) work for $\mathbb{A}$, say $\mathbb{A}$ has bounded width.

Equivalent notions: treewidth duality, Datalog, pebble games, etc

Picture for bounded width?



Bounded width

Theorem

TFAE:

1. $\mathbb{A}$ *doesn't pp-construct* 3-$\textsc{Lin}p$ *for any* p;

Bounded width

Theorem

TFAE:

1. $\mathbb{A}$ *doesn't pp-construct* 3-LINp *for any* p*;*
2. $\mathbb{A}$ *has polymorphisms* f_3, f_4 *such that* *[Kozik et al.'14]*

$$f_3(x,x,y) = f_3(x,y,x) = f_3(y,x,x) =$$
$$f_4(x,x,x,y) = \ldots = f_4(y,x,x,x)$$

Bounded width

Theorem

TFAE:

1. $\mathbb{A}$ *doesn't pp-construct* $3\text{-}\mathrm{LIN}p$ *for any* p*;*
2. $\mathbb{A}$ *has polymorphisms* f_3, f_4 *such that* *[Kozik et al.'14]*
   $f_3(x,x,y) = f_3(x,y,x) = f_3(y,x,x) =$
   $f_4(x,x,x,y) = \ldots = f_4(y,x,x,x)$
3. $\mathbb{A}$ *has bounded width* *[Barto, Kozik'09-14]*
4. $\mathbb{A}$ *has width (2,3)* *[Barto'16, Bulatov]*

Bounded width

Theorem

TFAE:

1. $\mathbb{A}$ *doesn't pp-construct* $3\text{-}\mathrm{LIN}p$ *for any* p*;*
2. $\mathbb{A}$ *has polymorphisms* f_3, f_4 *such that* *[Kozik et al.'14]*
   $f_3(x,x,y) = f_3(x,y,x) = f_3(y,x,x) =$
   $f_4(x,x,x,y) = \ldots = f_4(y,x,x,x)$
3. $\mathbb{A}$ *has bounded width* *[Barto, Kozik'09-14]*
4. $\mathbb{A}$ *has width (2,3)* *[Barto'16, Bulatov]*
5. $\mathrm{CSP}(\mathbb{A})$ *is robustly solvable* *[Barto, Kozik'12-16]*
   — *For each almost satisfiable instance, an almost satisfying assignment can be found in poly-time.*

Bounded width

Theorem

TFAE:

1. $\mathbb{A}$ *doesn't pp-construct* $3\text{-}\mathrm{Lin}p$ *for any* p*;*
2. $\mathbb{A}$ *has polymorphisms* f_3, f_4 *such that* *[Kozik et al.'14]*
   $f_3(x,x,y) = f_3(x,y,x) = f_3(y,x,x) =$
   $f_4(x,x,x,y) = \ldots = f_4(y,x,x,x)$
3. $\mathbb{A}$ *has bounded width* *[Barto, Kozik'09-14]*
4. $\mathbb{A}$ *has width (2,3)* *[Barto'16, Bulatov]*
5. $\mathrm{CSP}(\mathbb{A})$ *is robustly solvable* *[Barto, Kozik'12-16]*
   — *For each almost satisfiable instance, an almost satisfying assignment can be found in poly-time.*

Polymorphisms are used only to prove correctness of the algorithm.

Bulatov's and Zhuk's algorithms (very roughly), 1

Theorem (Bulatov'17, Zhuk'17)

If $\mathbb{A}$ *has a weak near-unanimity polymorphism (equivalently,* $\mathrm{Pol}(\mathbb{A}) \not\to \mathcal{T}$*) then* $\mathrm{CSP}(\mathbb{A})$ *is in* **P**.

On a VERY high level, the two algorithms are similar:

Bulatov's and Zhuk's algorithms (very roughly), 1

Theorem (Bulatov'17, Zhuk'17)

If $\mathbb{A}$ *has a weak near-unanimity polymorphism (equivalently,* $\mathrm{Pol}(\mathbb{A}) \not\to \mathcal{T}$*) then* $\mathrm{CSP}(\mathbb{A})$ *is in* **P**.

On a VERY high level, the two algorithms are similar:
Both heavily use local propagation.

Bulatov's and Zhuk's algorithms (very roughly), 1

Theorem (Bulatov'17, Zhuk'17)

If $\mathbb{A}$ *has a weak near-unanimity polymorphism (equivalently,* $\mathrm{Pol}(\mathbb{A}) \not\to \mathcal{T}$*) then* $\mathrm{CSP}(\mathbb{A})$ *is in* **P**.

On a VERY high level, the two algorithms are similar:
Both heavily use local propagation.

Both are recursive:

- initially, let $dom(v) = A$ for each variable v
- if some $dom(v)$ can be reduced without losing all solutions, reduce it and recurse

Bulatov's and Zhuk's algorithms (very roughly), 1

Theorem (Bulatov'17, Zhuk'17)

If $\mathbb{A}$ *has a weak near-unanimity polymorphism (equivalently,* $\mathrm{Pol}(\mathbb{A}) \not\to \mathcal{T}$*) then* $\mathrm{CSP}(\mathbb{A})$ *is in* **P**.

On a VERY high level, the two algorithms are similar:
Both heavily use local propagation.

Both are recursive:

- initially, let $dom(v) = A$ for each variable v
- if some $dom(v)$ can be reduced without losing all solutions, reduce it and recurse
- if an instance can be split/decomposed into a constant number of disjoint instances, split/decompose and recurse
- this can be done in many situations

Bulatov's and Zhuk's algorithms (very roughly), 2

The base cases for recursion (roughly) are:

- (B and Z) all domains 1-element $\rightarrow$ solution
- (B and Z) some domain empty $\rightarrow$ no solution

Bulatov's and Zhuk's algorithms (very roughly), 2

The base cases for recursion (roughly) are:

- (B and Z) all domains 1-element $\rightarrow$ solution
- (B and Z) some domain empty $\rightarrow$ no solution
- (B) few subpowers algorithm is applicable
- (Z) systems of linear equations over $\mathbb{Z}_p$

Bulatov's and Zhuk's algorithms (very roughly), 2

The base cases for recursion (roughly) are:

- (B and Z) all domains 1-element $\rightarrow$ solution
- (B and Z) some domain empty $\rightarrow$ no solution
- (B) few subpowers algorithm is applicable
- (Z) systems of linear equations over $\mathbb{Z}_p$

However, the recursion is organised very differently:

- (B) heavily uses deep algebraic structure
- (Z) still very algebraic, but more elementary
- (B and Z) a whole range of diff algebraic tricks

Feder-Vardi conjecture proved – End of the road ???

Feder-Vardi conjecture proved – End of the road ???

Actually, we are rather closer to the beginning!

Message

1. **Decision CSP**: Can all constraints be satisfied?
2. **Counting CSP**: Count the number of solutions
3. **Max CSP**: Find a map satisfying max number of constraints
4. **Approx Max CSP**: Satisfy $c \times \mathrm{Opt}$ number of constraints
5. **Approx Min CSP**: assuming $1 - \epsilon$ fraction of constraints can be satisfied, find a map satisfying $\geq 1 - g(\epsilon)$ fraction.
6. **Promise CSP**: ... tomorrow

Each of the above has an appropriate notion of symmetry!

- lack of symmetries $\Rightarrow$ hardness
- symmetries $\Rightarrow$ efficient algorithms
- symmetries of higher dimension/arity are important

Useful surveys

Much of what is covered in this lecture can be found in survey

- Polymorphisms, and how to use them.
  L. Barto, A. Krokhin, and R. Willard.

It is written specifically for those without algebra background!

See also the full (open-access) volume of surveys:

- The Constraint Satisfaction Problem: Complexity and Approximability. Editors: A. Krokhin and S. Živný. Dagstuhl Follow-Ups series, Volume 7, 2017.
  http://drops.dagstuhl.de/portals/dfu/index.php?semnr=16027