Universal algebra and the constraint satisfaction problem

Libor Barto

Charles University in Prague

ASL North American Annual Meeting
University of Colorado, 20 May 2014
Constraint satisfaction problem (CSP)

- Common framework for many practical problems
Constraint satisfaction problem (CSP)

- Common framework for many practical problems
- This is the last time when the word “practical” appears
Constraint satisfaction problem (CSP)

- Common framework for some computational problems
  - Broad enough to include interesting examples
  - Narrow enough to make significant progress (on all problems within a class, rather than just a single computational problem)
Constraint satisfaction problem (CSP)

- Common framework for some computational problems
  - Broad enough to include interesting examples
  - Narrow enough to make significant progress (on all problems within a class, rather than just a single computational problem)
- Main achievement: better understanding why problems are easy or hard:
  - Hardness comes from lack of symmetry
  - Symmetries of higher arity are important (not just automorphisms or endomorphisms)
  - Universal algebra (not just group or semigroup theory)
- Long term goal: go beyond CSP
Constraint satisfaction problem (CSP)

- Common framework for some computational problems
  - Broad enough to include interesting examples
  - Narrow enough to make significant progress (on all problems within a class, rather than just a single computational problem)
- Main achievement: better understanding why problems are easy or hard:
  - Hardness comes from lack of symmetry
  - Symmetries of higher arity are important (not just automorphisms or endomorphisms)
    → universal algebra (not just group or semigroup theory)
Constraint satisfaction problem (CSP)

- Common framework for some computational problems
  - Broad enough to include interesting examples
  - Narrow enough to make significant progress (on all problems within a class, rather than just a single computational problem)

- Main achievement: better understanding why problems are easy or hard:
  - Hardness comes from lack of symmetry
  - Symmetries of higher arity are important (not just automorphisms or endomorphisms)
    → universal algebra (not just group or semigroup theory)

- Long term goal: go beyond CSP
**Instance of the CSP**

**Definition**

Instance of the CSP is a list of constraints – expression of the form

\[ R_1(x, y, z), R_2(t, z), R_1(y, y, z), \ldots \]

where \( R_i \) are relations on a common domain \( A \)

(subsets of \( A^k \) or mappings \( A^k \rightarrow \{true, false\} \)).

**Assignment** = mapping \( variables \rightarrow domain \)
Instance of the CSP

Definition

Instance of the CSP is a list of constraints – expression of the form

\[ R_1(x, y, z), \ R_2(t, z), \ R_1(y, y, z), \ldots \]

where \( R_i \) are relations on a common domain \( A \)

(subsets of \( A^k \) or mappings \( A^k \rightarrow \{\text{true, false}\} \)).

Assignment = mapping variables \( \rightarrow \) domain

- **Decision CSP**: Is there an assignment satisfying all constraints (a solution)
Instance of the CSP

A \textit{Definition} of the CSP is a list of constraints – expression of the form

\[ R_1(x, y, z), \quad R_2(t, z), \quad R_1(y, y, z), \quad \ldots \]

where \( R_i \) are relations on a common domain \( A \)

(subsets of \( A^k \) or mappings \( A^k \rightarrow \{true, false\} \)).

\textit{Assignment} = mapping \textit{variables} \rightarrow \textit{domain}

- \textbf{Decision CSP:} Is there an assignment satisfying all constraints (a \textit{solution})
- \textbf{Search problem:} Find a solution
**Instance of the CSP**

**Definition**

**Instance of the CSP** is a list of constraints – expression of the form

\[ R_1(x, y, z), \ R_2(t, z), \ R_1(y, y, z), \ldots \]

where \( R_i \) are relations on a common domain \( A \)

(subsets of \( A^k \) or mappings \( A^k \rightarrow \{true, false\} \)).

**Assignment** = mapping **variables** \( \rightarrow \) **domain**

- **Decision CSP**: Is there an assignment satisfying all constraints (a **solution**)
- **Search problem**: Find a solution
- **Counting CSP**: How many solutions are there?
Instance of the CSP

**Definition**

Instance of the CSP is a list of constraints – expression of the form

\[ R_1(x, y, z), \ R_2(t, z), \ R_1(y, y, z), \ldots \]

where \( R_i \) are relations on a common domain \( A \)

(subsets of \( A^k \) or mappings \( A^k \to \{true, false\} \)).

**Assignment** = mapping **variables** \( \rightarrow \) **domain**

- **Decision CSP:** Is there an assignment satisfying all constraints (a **solution**)
- **Search problem:** Find a solution
- **Counting CSP:** How many solutions are there?
- **Max-CSP:** Find a map satisfying maximum number of constraints
Definition

Instance of the CSP is a list of constraints – expression of the form

\[ R_1(x, y, z), \ R_2(t, z), \ R_1(y, y, z), \ldots \]

where \( R_i \) are relations on a common domain \( A \)

(subsets of \( A^k \) or mappings \( A^k \rightarrow \{true, false\} \)).

Assignment = mapping variables \( \rightarrow \) domain

- **Decision CSP:** Is there an assignment satisfying all constraints (a solution)
- **Search problem:** Find a solution
- **Counting CSP:** How many solutions are there?
- **Max-CSP:** Find a map satisfying maximum number of constraints
- **Approx. Max-CSP:** Find a map satisfying \( 0.7 \times Optimum \) constraints
Instance of the CSP

Definition

Instance of the CSP is a list of constraints – expression of the form

\[ R_1(x, y, z), \ R_2(t, z), \ R_1(y, y, z), \ldots \]

where \( R_i \) are relations on a common domain \( A \)

(subsets of \( A^k \) or mappings \( A^k \rightarrow \{true, false\} \)).

Assignment = mapping variables \( \rightarrow \) domain

- **Decision CSP:** Is there an assignment satisfying all constraints (a solution)
- **Search problem:** Find a solution
- **Counting CSP:** How many solutions are there?
- **Max-CSP:** Find a map satisfying maximum number of constraints
- **Approx. Max-CSP:** or: Find a map satisfying 0.3-fraction of constraints given 0.6-satisfiable instance

Robust CSP: Find an almost satisfying assignment given an almost satisfiable instance
Definition

Instance of the CSP is a list of constraints – expression of the form

\[ R_1(x, y, z), \ R_2(t, z), \ R_1(y, y, z), \ldots \]

where \( R_i \) are relations on a common domain \( A \)

(subsets of \( A^k \) or mappings \( A^k \rightarrow \{\text{true}, \text{false}\} \)).

Assignment = mapping variables \( \rightarrow \) domain

- **Decision CSP:** Is there an assignment satisfying all constraints (a solution)
- **Search problem:** Find a solution
- **Counting CSP:** How many solutions are there?
- **Max-CSP:** Find a map satisfying maximum number of constraints
- **Approx. Max-CSP:** or: Find a map satisfying 0.3-fraction of constraints given 0.6-satisfiable instance
- **Robust CSP:** Find an almost satifying assignment given an almost satisfiable instance
Interesting subproblems ... restrict the set of allowed relations
Definition

\[ \mathcal{A} = (A; R_1, R_2, \ldots) \]: relational structure with \( A \) finite

Instance of \( \text{CSP}(\mathcal{A}) \): Expression of the form

\[ R_1(x, y, z), \ R_2(t, z), \ R_1(y, y, z), \ldots \]

where each \( R_i \) is in \( \mathcal{A} \).
Definition

\( \mathbb{A} = (A; R_1, R_2, \ldots) \): relational structure with \( A \) finite

Instance of \( \text{CSP}(\mathbb{A}) \): Expression of the form

\[ R_1(x, y, z), \ R_2(t, z), \ R_1(y, y, z), \ldots \]

where each \( R_i \) is in \( \mathbb{A} \).

▶ What is the computational complexity for fixed \( \mathbb{A} \)?
CSP over a structure (aka constraint language)

Definition

$\mathbb{A} = (A; R_1, R_2, \ldots)$: relational structure with $A$ finite

Instance of $CSP(\mathbb{A})$: Expression of the form

$R_1(x, y, z), R_2(t, z), R_1(y, y, z), \ldots$

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

- What is the computational complexity for fixed $\mathbb{A}$?
- **This talk:** Mainly decision $CSP(\mathbb{A})$
CSP over a structure (aka constraint language)

Definition

\[ \mathbb{A} = (A; R_1, R_2, \ldots) \]: relational structure with \( A \) finite

Instance of \( CSP(\mathbb{A}) \): Expression of the form

\[ R_1(x, y, z), \ R_2(t, z), \ R_1(y, y, z), \ldots \]

where each \( R_i \) is in \( \mathbb{A} \).

- What is the computational complexity for fixed \( \mathbb{A} \)?
- **This talk**: Mainly decision \( CSP(\mathbb{A}) \)
- **Other interesting problems**:
  - restrict something else than the set of allowed relations
  - allow infinite \( A \)
  - allow weighted relations: mappings \( A^k \to Q \cup \{\infty\} \)
  - (approximate) counting, Max-CSP, Approx Max-CSP
**Definition**

\[ \mathbb{A} = (A; R_1, R_2, \ldots) \]: relational structure with \( A \) finite

**Instance of CSP(\( \mathbb{A} \))**: Expression of the form

\[ R_1(x, y, z), \ R_2(t, z), \ R_1(y, y, z), \ldots \]

where each \( R_i \) is in \( \mathbb{A} \).

**Examples:**

- **3-SAT**: \( \mathbb{A} = (\{0, 1\}; x \lor y \lor z, x \lor y \lor \neg z, \ldots) \)
CSP over a structure (aka constraint language)

**Definition**

\[ \mathbb{A} = (A; R_1, R_2, \ldots ) \]: relational structure with \( A \) finite

**Instance of CSP(\( \mathbb{A} \))**: Expression of the form

\[ R_1(x, y, z), R_2(t, z), R_1(y, y, z), \ldots \]

where each \( R_i \) is in \( \mathbb{A} \).

**Examples:**

- **3-SAT**: \( \mathbb{A} = (\{0, 1\}; x \lor y \lor z, x \lor y \lor \neg z, \ldots ) \)
- **3-COL**: \( \mathbb{A} = (\{0, 1, 2\}; x \neq y) \)
CSP over a structure (aka constraint language)

**Definition**

\( \mathbb{A} = (A; R_1, R_2, \ldots) \): relational structure with \( A \) finite

**Instance of CSP(\( \mathbb{A} \))**: Expression of the form

\[ R_1(x, y, z), R_2(t, z), R_1(y, y, z), \ldots \]

where each \( R_i \) is in \( \mathbb{A} \).

**Examples:**

- **3-SAT**: \( \mathbb{A} = (\{0, 1\}; x \lor y \lor z, x \lor y \lor \neg z, \ldots) \)
- **3-COL**: \( \mathbb{A} = (\{0, 1, 2\}; x \neq y) \)
- **HORN-3-SAT**: \( \mathbb{A} = (\{0, 1\}; x, \neg x, x \land y \rightarrow z) \)
CSP over a structure (aka constraint language)

Definition

\[ \mathbb{A} = (A; R_1, R_2, \ldots ) \]: relational structure with \( A \) finite

Instance of CSP(\( \mathbb{A} \)): Expression of the form

\[ R_1(x, y, z), \: R_2(t, z), \: R_1(y, y, z), \: \ldots \]

where each \( R_i \) is in \( \mathbb{A} \).

Examples:

- 3-SAT: \( \mathbb{A} = (\{0, 1\}; x \lor y \lor z, x \lor y \lor \neg z, \ldots ) \)
- 3-COL: \( \mathbb{A} = (\{0, 1, 2\}; x \neq y) \)
- HORN-3-SAT: \( \mathbb{A} = (\{0, 1\}; x, \neg x, x \land y \rightarrow z) \)
- q-LIN: \( \mathbb{A} = (GF(q); \text{affine subspaces}) \).
## CSP over a structure (aka constraint language)

### Definition

\[ \mathbb{A} = (A; R_1, R_2, \ldots) : \text{relational structure with } A \text{ finite} \]

**Instance of \( \text{CSP}(\mathbb{A}) \):** Expression of the form

\[
R_1(x, y, z), \ R_2(t, z), \ R_1(y, y, z), \ldots
\]

where each \( R_i \) is in \( \mathbb{A} \).

### Examples:

- **3-SAT:** \( \mathbb{A} = (\{0, 1\}; x \lor y \lor z, x \lor y \lor \neg z, \ldots) \)
- **3-COL:** \( \mathbb{A} = (\{0, 1, 2\}; x \neq y) \)
- **HORN-3-SAT:** \( \mathbb{A} = (\{0, 1\}; x, \neg x, x \land y \rightarrow z) \)
- **q-LIN:** \( \mathbb{A} = (\text{GF}(q); \text{affine subspaces}) \).
- Digraph reachability: \( \mathbb{A} = (\{0, 1\}; x, \neg x, x \leq y) \)
- Graph reachability: \( \mathbb{A} = (\{0, 1\}, x, \neg x, x = y) \)
CSP over a structure (aka constraint language)

Definition

\[ \mathbb{A} = (A; R_1, R_2, \ldots) : \text{relational structure with } A \text{ finite} \]

Instance of \( \text{CSP}(\mathbb{A}) \): Expression of the form

\[ R_1(x, y, z), \ R_2(t, z), \ R_1(y, y, z), \ldots \]

where each \( R_i \) is in \( \mathbb{A} \).

Examples:

- (NP-c) 3-SAT: \( \mathbb{A} = (\{0, 1\}; x \lor y \lor z, x \lor y \lor \neg z, \ldots) \)
- (NP-c) 3-COL: \( \mathbb{A} = (\{0, 1, 2\}; x \neq y) \)
- (P-c) HORN-3-SAT: \( \mathbb{A} = (\{0, 1\}; x, \neg x, x \land y \rightarrow z) \)
- \( q\)-LIN: \( \mathbb{A} = (GF(q); \text{affine subspaces}) \).
- (NL-c) Digraph reachability: \( \mathbb{A} = (\{0, 1\}; x, \neg x, x \leq y) \)
- (L-c) Graph reachability: \( \mathbb{A} = (\{0, 1\}, x, \neg x, x = y) \)
Decision CSP as model checking problem

**CSP(\(\mathbb{A}\)) :**

**Instance:** Sentence \(\phi\) in the language of \(\mathbb{A}\) with \(\exists\) and \(\land\)

**Question:** Is \(\phi\) true in \(\mathbb{A}\)?

From 27 cases only 3 interesting (others reduce to these or are boring)

▶ \(\{\exists, \land, (=)\}\) (CSP) open

▶ \(\{\exists, \forall, \land, (=)\}\) (qCSP) open

▶ \(\{\exists, \forall, \land, \lor\}\) (Positive equality free) solved - tetrachotomy P, NP-c, co-NP-c, PSPACE-c

B. Martin, F. Madelaine 11
Decision CSP as model checking problem

\[ CSP(\mathbb{A}) : \]

**Instance:** Sentence \( \phi \) in the language of \( \mathbb{A} \) with \( \exists \) and \( \land \)

**Question:** Is \( \phi \) true in \( \mathbb{A} \)?

**What about:** Allow some other combination of \( \{\exists, \forall, \land, \lor, \neg, =, \neq\} \).
Decision CSP as model checking problem

$CSP(\mathbb{A})$:

**Instance:** Sentence $\phi$ in the language of $\mathbb{A}$ with $\exists$ and $\land$

**Question:** Is $\phi$ true in $\mathbb{A}$?

**What about:** Allow some other combination of $
\{\exists, \forall, \land, \lor, \neg, =, \neq\}.
$

From $2^7$ cases only 3 interesting (others reduce to these or are boring)

- $\{\exists, \land, (\neq)\}$ (CSP) open
- $\{\exists, \forall, \land, (\neq)\}$ (qCSP) open
- $\{\exists, \forall, \land, \lor\}$ (Positive equality free) solved - tetrachotomy $P$, NP-c, co-NP-c, PSPACE-c

B. Martin, F. Madelaine 11
The dichotomy conjecture

A largest natural class of problems with a dichotomy?
The dichotomy conjecture

A largest natural class of problems with a dichotomy?

Conjecture (The dichotomy conjecture Feder and Vardi’93)

For every $\mathcal{A}$, decision $\text{CSP}(\mathcal{A})$ is either in $P$ or $NP$-complete.
A largest natural class of problems with a dichotomy?

Conjecture (The dichotomy conjecture Feder and Vardi’93)
For every $\mathbb{A}$, decision $\text{CSP}(\mathbb{A})$ is either in $P$ or $NP$-complete.

- Evidence (in 93):
  - True for $|\mathbb{A}| = 2$ Schaefer’78
  - True if $\mathbb{A} = (\mathbb{A}; R)$, $R$ is binary and symmetric
    Hell and Nešetřil’90
A largest natural class of problems with a dichotomy?

**Conjecture (The dichotomy conjecture Feder and Vardi'93)**

For every $\mathbb{A}$, decision $\text{CSP}(\mathbb{A})$ is either in $P$ or $NP$-complete.

- Evidence (in 93):
  - True for $|\mathbb{A}| = 2$ Schaefer’78
  - True if $\mathbb{A} = (A; R)$, $R$ is binary and symmetric
    Hell and Nešetřil’90
  - Feder and Vardi suggested that tractability is tied to “closure properties”
The dichotomy conjecture

A largest natural class of problems with a dichotomy?

Conjecture (The dichotomy conjecture Feder and Vardi’93)

For every $A$, decision $\text{CSP}(A)$ is either in $P$ or $NP$-complete.

- Evidence (in 93):
  - True for $|A| = 2$ Schaefer’78
  - True if $A = (A; R)$, $R$ is binary and symmetric Hell and Nešetřil’90

- Feder and Vardi suggested that tractability is tied to “closure properties”

- → algebraic approach Bulatov, Jeavons, Krokhin’00
Most of the definitions will be imprecise

Almost no theorem is true as stated
PP and UA
If A “can simulate” B then \( CSP(A) \) is at least as hard as \( CSP(B) \).
If A “can simulate” B then $CSP(A)$ is at least as hard as $CSP(B)$.

What does simulate mean?
If A “can simulate” B then $CSP(A)$ is at least as hard as $CSP(B)$.

What does simulate mean?

(Slightly imprecise) answer:
“can simulate” means “positively primitively (pp) interprets”
Simulation

- If $A$ “can simulate” $B$ then $CSP(A)$ is at least as hard as $CSP(B)$.
- **What does simulate mean?**
  - (Slightly imprecise) answer: “can simulate” means “positively primitively (pp) interprets”
  - Special case of pp-interpretability is pp-definability
If $\mathcal{A}$ “can simulate” $\mathcal{B}$ then $CSP(\mathcal{A})$ is at least as hard as $CSP(\mathcal{B})$.

**What does simulate mean?**

(Slightly imprecise) answer:

“can simulate” means “positively primitively (pp) interprets”

Special case of pp-interpretability is pp-definability

Assume $\mathcal{A}, \mathcal{B}$ have the same domain.

$\mathcal{A}$ **pp-defines** $\mathcal{B} = \text{relations in } \mathcal{B} \text{ definable using relations in } \mathcal{A}, \text{ and } \exists, =, \land.$
Example of pp-definability

\[ \mathbb{A} = (A; R), \text{ where } R \text{ is ternary} \]
Example of pp-definability

- $\mathcal{A} = (A; R)$, where $R$ is ternary
- $\mathcal{B} = (B; S, T)$, where $S$ is binary and $T$ is unary
  - $S(x, y)$ iff $(\exists z) R(x, y, z) \land R(y, y, x)$
  - $T(x)$ iff $R(x, x, x)$
Example of pp-definability

- $\mathbb{A} = (A; R)$, where $R$ is ternary
- $\mathbb{B} = (B; S, T)$, where $S$ is binary and $T$ is unary
  - $S(x, y)$ iff $(\exists z) R(x, y, z) \land R(y, y, x)$
  - $T(x)$ iff $R(x, x, x)$
- Each instance of CSP($\mathbb{B}$), eg.

$$T(z), \ S(x, y)$$
Example of pp-definability

- $\mathbb{A} = (A; R)$, where $R$ is ternary
- $\mathbb{B} = (B; S, T)$, where $S$ is binary and $T$ is unary
  - $S(x, y)$ iff $(\exists z) R(x, y, z) \land R(y, y, x)
  - $T(x)$ iff $R(x, x, x)$
- Each instance of CSP($\mathbb{B}$), eg.
  \[
  T(z), \ S(x, y)
  \]
- can be rewritten to an equivalent instance of CSP($\mathbb{A}$)
  \[
  R(z, z, z), \ R(x, y, w), \ R(y, y, x)
  \]
Example of pp-definability

- $\mathbb{A} = (A; R)$, where $R$ is ternary
- $\mathbb{B} = (B; S, T)$, where $S$ is binary and $T$ is unary
  - $S(x, y)$ iff $(\exists z) \ R(x, y, z) \land R(y, y, x)$
  - $T(x)$ iff $R(x, x, x)$
- Each instance of $\text{CSP}(\mathbb{B})$, eg.
  
  $$T(z), \ S(x, y)$$

- can be rewritten to an equivalent instance of $\text{CSP}(\mathbb{A})$
  
  $$R(z, z, z), \ R(x, y, w), \ R(y, y, x)$$

- Thus $\text{CSP}(\mathbb{A})$ is at least as hard as $\text{CSP}(\mathbb{B})$
pp-interpretation, the borderline?

- A pp-interprets B if

The domain of B is a pp-definable subset of A modulo a pp-definable equivalence.

The relations of B are "pp-definable" from A (m-ary relation on B is defined as a km-ary relation on A).

If A pp-interprets the structure corresponding to 3-SAT then CSP(A) is NP-complete.

This explains NP-completeness for all known NP-complete CSPs...

Conjecture (The algebraic dichotomy conjecture Bulatov, Jeavons, Krokhin)
If A does not interpret 3-SAT then CSP(A) is in P.

Similar conjectures and hardness results about L, NL, Larose, Tesson
pp-interpretation, the borderline?

- A pp-interprets B if
  - The domain of B is a pp-definable subset of $A^k$ modulo a pp-definable equivalence
A pp-interprets B if

- The domain of B is a pp-definable subset of $A^k$ modulo a pp-definable equivalence
- The relations of B are “pp-definable” from A

($m$-ary relation on $B$ is defined as a $km$-ary relation on $A$)
pp-interpretation, the borderline?

- \( \mathbb{A} \) pp-interpret \( \mathbb{B} \) if
  - The domain of \( \mathbb{B} \) is a pp-definable subset of \( A^k \) modulo a pp-definable equivalence
  - The relations of \( \mathbb{B} \) are “pp-definable” from \( \mathbb{A} \)
    - \((m\text{-ary relation on } B \text{ is defined as a } km\text{-ary relation on } A)\)
- If \( \mathbb{A} \) pp-interprets the structure corresponding to 3-SAT then \( \text{CSP}(\mathbb{A}) \) is NP-complete \( \text{BJK} \)
A pp-interprets B if

- The domain of B is a pp-definable subset of $A^k$ modulo a pp-definable equivalence
- The relations of B are “pp-definable” from A
  ($m$-ary relation on B is defined as a $km$-ary relation on A)

If A pp-interprets the structure corresponding to 3-SAT
then CSP(A) is NP-complete BJK

This explains NP-completeness for all known NP-complete CSPs...
pp-interpretation, the borderline?

- A pp-interprets B if
  - The domain of B is a pp-definable subset of $A^k$ modulo a pp-definable equivalence
  - The relations of B are “pp-definable” from A
    ($m$-ary relation on B is defined as a $km$-ary relation on A)
- If A pp-interprets the structure corresponding to 3-SAT then CSP(A) is NP-complete BJK
- This explains NP-completeness for all known NP-complete CSPs...

Conjecture (The algebraic dichotomy conjecture Bulatov, Jeavons, Krokhin)

If A does not interpret 3-SAT then CSP(A) is in P.
pp-interpretation, the borderline?

- $A$ pp-interprets $B$ if
  - The domain of $B$ is a pp-definable subset of $A^k$ modulo a pp-definable equivalence
  - The relations of $B$ are “pp-definable” from $A$ 
    ($m$-ary relation on $B$ is defined as a $km$-ary relation on $A$)
- If $A$ pp-interprets the structure corresponding to 3-SAT then $CSP(A)$ is NP-complete BJK
- This explains NP-completeness for all known NP-complete CSPs...

Conjecture (The algebraic dichotomy conjecture Bulatov, Jeavons, Krokhin)

If $A$ does not interpret 3-SAT then $CSP(A)$ is in $P$.

Similar conjectures and hardness results about L, NL Larose, Tesson
Operation $t : A^k \rightarrow A$ is compatible with relation $R \subseteq A^n$, if $R$ is closed under coordinate-wise application of $t$. 
Operation $t : A^k \rightarrow A$ is compatible with relation $R \subseteq A^n$, if $R$ is closed under coordinate-wise application of $t$.

Operation $t : A^k \rightarrow A$ is a polymorphism of $A$ if it is compatible with every relation in $A$

- polymorphism with $k = 1 =$ endomorphism
- polymorphism with $k > 1 =$ higher arity symmetry
PP and UA

- Operation $t : A^k \to A$ is **compatible** with relation $R \subseteq A^n$, if $R$ is closed under coordinate-wise application of $t$.

- Operation $t : A^k \to A$ is a **polymorphism** of $A$ if it is compatible with every relation in $A$.

  polymorphism with $k = 1$ = endomorphism
  polymorphism with $k > 1$ = higher arity symmetry

- $\text{Pol}(A) = (A; \text{all polymorphisms of } A)$ ... the algebra of polymorphisms
Operation $t : A^k \to A$ is compatible with relation $R \subseteq A^n$, if $R$ is closed under coordinate-wise application of $t$.

Operation $t : A^k \to A$ is a polymorphism of $A$ if it is compatible with every relation in $A$.

Pol($A$) = ($A$; all polymorphisms of $A$) ... the algebra of polymorphisms

Old theorem: $A$ pp-defines $B$ iff Pol($A$) $\subseteq$ Pol($B$)

Geiger'68, Bondarchuk, Kaluznin, Kotov, Romov'69
Operation $t : A^k \to A$ is compatible with relation $R \subseteq A^n$, if $R$ is closed under coordinate-wise application of $t$.

Operation $t : A^k \to A$ is a polymorphism of $A$ if it is compatible with every relation in $A$.

Pol($A$) = ($A$; all polymorphisms of $A$) ... the algebra of polymorphisms.

Old theorem: $A$ pp-defines $B$ iff $Pol(A) \subseteq Pol(B)$

Geiger'68, Bondarchuk, Kaluznin, Kotov, Romov'69

More generally: $A$ pp-interprets $B$ iff $Pol(B)$ interprets $Pol(A)$

Birkhoff'35, Bodirsky'08
Operation $t : A^k \rightarrow A$ is compatible with relation $R \subseteq A^n$, if $R$ is closed under coordinate-wise application of $t$.

Operation $t : A^k \rightarrow A$ is a polymorphism of $A$ if it is compatible with every relation in $A$.

polymorphism with $k = 1 = \text{endomorphism}$

polymorphism with $k > 1 = \text{higher arity symmetry}$

$\text{Pol}(A) = (A; \text{all polymorphisms of } A) \ldots \text{the algebra of polymorphisms}$

Old theorem: $A$ pp-defines $B$ iff $\text{Pol}(A) \subseteq \text{Pol}(B)$

Geiger'68, Bondarchuk, Kaluznin, Kotov, Romov'69

More generally: $A$ pp-interprets $B$ iff $\text{Pol}(B)$ interprets $\text{Pol}(A)$

Birkhoff'35, Bodirsky'08

Interpretations closely connected to central objects of study in UA: varieties and Mal’tsev conditions
Interpretations closely connected to central objects of study in UA: varieties and Mal’tsev conditions
On the algebraic approach

Interpretations closely connected to central objects of study in UA: varieties and Mal’tsev conditions

- Important conditions on $\mathbb{A}$ correspond to previously studied conditions for $\text{Pol}(\mathbb{A})$

Theorem

The following are equivalent.

1. $\mathbb{A}$ does not interpret (=cannot simulate) 3-SAT
2. . . .
33. . . .
34. $\text{Pol}(\mathbb{A})$ contains an operation $t$ such that $t(a, a, \ldots, a) = a$ and $t(a_1, a_2, \ldots, a_k) = t(a_2, \ldots, a_k, a_1)$ for all $a, a_i \in \mathbb{A}$
On the algebraic approach

Interpretations closely connected to central objects of study in UA: varieties and Mal’tsev conditions

- Important conditions on \( \mathbb{A} \) correspond to previously studied conditions for \( \text{Pol}(\mathbb{A}) \)
- We can use UA to identify interesting special cases

Theorem

The following are equivalent.

1. \( A \) does not interpret (=cannot simulate) 3-SAT
2. ...
3. \( \text{Pol}(A) \) contains an operation \( t \) such that \( t(a, a, \ldots, a) = a \) and \( t(a_1, a_2, \ldots, a_k) = t(a_2, \ldots, a_k, a_1) \) for all \( a, a_i \in A \)

B, Kozik’10
On the algebraic approach

Interpretations closely connected to central objects of study in UA: varieties and Mal’tsev conditions

- Important conditions on $\mathbb{A}$ correspond to previously studied conditions for $\text{Pol}(\mathbb{A})$
- We can use UA to identify interesting special cases
- Sometimes operations are directly used in algorithms

Theorem

The following are equivalent.

1. $\mathbb{A}$ does not interpret (=cannot simulate) 3-SAT
2. . . .
33. . . .
34. $\text{Pol}(\mathbb{A})$ contains an operation $t$ such that $t(a, a, \ldots, a) = a$ and $t(a_1, a_2, \ldots, a_k) = t(a_2, \ldots, a_k, a_1)$ for all $a, a_i \in \mathbb{A}$

B, Kozik’10
On the algebraic approach

Interpretations closely connected to central objects of study in UA: varieties and Mal’tsev conditions

- Important conditions on $\mathbb{A}$ correspond to previously studied conditions for $\text{Pol}(\mathbb{A})$
- We can use UA to identify interesting special cases
- Sometimes operations are directly used in algorithms

**Theorem**

The following are equivalent.

1. $\mathbb{A}$ does not interpret (=cannot simulate) 3-SAT
2. . . .
... Taylor, Hobby, McKenzie, Bulatov, Maróti, Siggers, . . .
33. . . .
34. $\text{Pol}(\mathbb{A})$ contains an operation $t$ such that $t(a, a, \ldots, a) = a$ and $t(a_1, a_2, \ldots, a_k) = t(a_2, \ldots, a_k, a_1)$ for all $a, a_i \in A$

$B$, Kozik’10
Results
Better understanding of pre-algebraic results
Better understanding of pre-algebraic results

Far broader special cases solved. The dichotomy conjecture is true:

- if $|A| = 3$ Bulatov’06
- if $|A| = 4$ Marković et al.
- if $A$ contains all unary relations Bulatov’03, Barto’11
- if $A = (A; R)$ where $R$ is binary, without sources or sinks Barto, Kozik, Niven’09

Results
Results

- Better understanding of pre-algebraic results
- Far broader special cases solved. The dichotomy conjecture is true:
  - if $|A| = 3$ Bulatov’06
  - if $|A| = 4$ Marković et al.
  - if $A$ contains all unary relations Bulatov’03, Barto’11
  - if $A = (A; R)$ where $R$ is binary, without sources or sinks Barto, Kozik, Niven’09
- Applicability of known algorithmic principles understood
Better understanding of pre-algebraic results

Far broader special cases solved. The dichotomy conjecture is true:

- if $|A| = 3$ Bulatov’06
- if $|A| = 4$ Marković et al.
- if $A$ contains all unary relations Bulatov’03, Barto’11
- if $A = (A; R)$ where $R$ is binary, without sources or sinks Barto, Kozik, Niven’09

Applicability of known algorithmic principles understood

- Describing all solutions Idziak, Markovic, McKenzie, Valeriote, Willard’07
Better understanding of pre-algebraic results

Far broader special cases solved. The dichotomy conjecture is true:

- if $|A| = 3$ Bulatov'06
- if $|A| = 4$ Marković et al.
- if $A$ contains all unary relations Bulatov’03, Barto’11
- if $A = (A; R)$ where $R$ is binary, without sources or sinks Barto, Kozik, Niven’09

Applicability of known algorithmic principles understood

- Describing all solutions Idziak, Markovic, McKenzie, Valeriote, Willard’07
- Local consistency (constraint propagation) Barto, Kozik’09, Bulatov
Better understanding of pre-algebraic results

Far broader special cases solved. The dichotomy conjecture is true:

- if $|A| = 3$ Bulatov’06
- if $|A| = 4$ Marković et al.
- if $\mathbb{A}$ contains all unary relations Bulatov’03, Barto’11
- if $\mathbb{A} = (A; R)$ where $R$ is binary, without sources or sinks Barto, Kozik, Niven’09

Applicability of known algorithmic principles understood

- Describing all solutions Idziak, Markovic, McKenzie, Valeriote, Willard’07
- Local consistency (constraint propagation) Barto, Kozik’09, Bulatov
- All known tractable cases solvable by a combination of these two
Local consistency

**Roughly:** A has **bounded width** iff \( \text{CSP}(A) \) can be solved by checking local consistency
Local consistency

**Roughly**: \( A \) has **bounded width** iff CSP\((A)\) can be solved by checking local consistency

**More precisely**:
- Fix \( k \leq l \) (integers)
Local consistency

**Roughly:** $\mathcal{A}$ has **bounded width** iff CSP($\mathcal{A}$) can be solved by checking local consistency

**More precisely:**

- Fix $k \leq l$ (integers)
- $(k, l)$-algorithm: Derive the strongest constraints on $k$ variables which can be deduced by “considering” $l$ variables at a time.
Local consistency

**Roughly:** A has **bounded width** iff $\text{CSP}(A)$ can be solved by checking local consistency

**More precisely:**

- Fix $k \leq l$ (integers)
- $(k, l)$-algorithm: Derive the strongest constraints on $k$ variables which can be deduced by “considering” $l$ variables at a time.
- If a contradiction is found, answer “no” otherwise answer “yes”
Local consistency

**Roughly:** A has **bounded width** iff CSP(Å) can be solved by checking local consistency

**More precisely:**
- Fix \( k \leq l \) (integers)
- \((k, l)\)-algorithm: Derive the strongest constraints on \( k \) variables which can be deduced by “considering” \( l \) variables at a time.
- If a contradiction is found, answer “no” otherwise answer “yes”
- “no” answers are always correct
Local consistency

**Roughly:** A has **bounded width** iff CSP(A) can be solved by checking local consistency

**More precisely:**
- Fix $k \leq l$ (integers)
- $(k, l)$-algorithm: Derive the strongest constraints on $k$ variables which can be deduced by “considering” $l$ variables at a time.
- If a contradiction is found, answer “no” otherwise answer “yes”
- “no” answers are always correct
- if “yes” answers are correct for every instance of CSP(A) we say that A has **width** $(k, l)$.
Local consistency

Roughly:

A has bounded width \( (k, l) \) for some \( k \), \( l \) then \( A \) has bounded width \( (k, l) \).

We say that \( A \) has width \( (k, l) \).

If "yes" answers are correct for every instance of \( \text{CSP}(A) \), "no" answers are always correct.

Otherwise answer "yes".

If a contradiction is found, answer "no".

If a contradiction is found, answer "no".

a time.

variables which can be deduced by "considering" \( k \) variables at a time.

(\( k, l \)-algorithm: Derive the strongest constraints on \( k \) \( \geq l \)) (integers)

Fix \( k \geq l \) (integers)

More precisely:

Checking local consistency can be solved by

Roughly:

A has bounded width \( \iff \text{CSP}(A) \) can be solved by

Fix \( k \geq l \) (integers)
Local consistency

**Roughly:** \(\mathcal{A}\) has **bounded width** iff \(\text{CSP}(\mathcal{A})\) can be solved by checking local consistency

**More precisely:**

- Fix \(k \leq l\) (integers)
- \((k, l)\)-algorithm: Derive the strongest constraints on \(k\) variables which can be deduced by “considering” \(l\) variables at a time.
- If a contradiction is found, answer “no” otherwise answer “yes”
- “no” answers are always correct
- if “yes” answers are correct for every instance of \(\text{CSP}(\mathcal{A})\) we say that \(\mathcal{A}\) has **width** \((k, l)\).
- if \(\mathcal{A}\) has width \((k, l)\) for some \(k, l\) then \(\mathcal{A}\) has **bounded width**

Various equivalent formulations (bounded tree width duality, definability in Datalog)
Example of (2, 3)-consistency

Let \( \mathbb{A} = (\{0, 1\}; x = y, x \neq y) \)

Consider the instance

\[ x = y, y = z, z = w, x \neq w \]
Example of $(2, 3)$-consistency

Let $\mathbb{A} = (\{0, 1\}; x = y, x \neq y)$

Consider the instance

$$x = y, \ y = z, \ z = w, \ x \neq w$$

- By looking at $\{x, y, z\}$ we see (using $x = y$ and $y = z$) that $x = z$. 
Example of \((2, 3)\)-consistency

Let \(\mathbb{A} = (\{0, 1\}; x = y, x \neq y)\)

Consider the instance

\[x = y, \ y = z, \ z = w, \ x \neq w\]

- By looking at \(\{x, y, z\}\) we see (using \(x = y\) and \(y = z\)) that \(x = z\).
- By looking at \(\{x, z, w\}\) we see (using \(x = z\) and \(z = w\)) that \(x = w\).
Example of (2, 3)-consistency

Let $\mathbb{A} = (\{0, 1\}; x = y, x \neq y)$

Consider the instance

\[ x = y, \; y = z, \; z = w, \; x \neq w \]

- By looking at $\{x, y, z\}$ we see (using $x = y$ and $y = z$) that $x = z$.
- By looking at $\{x, z, w\}$ we see (using $x = z$ and $z = w$) that $x = w$.
- By looking at $\{x, w\}$ we now see a contradiction
Example of \((2, 3)\)-consistency

Let \(\mathbb{A} = (\{0, 1\}; x = y, x \neq y)\)

Consider the instance

\[x = y, \ y = z, \ z = w, \ x \neq w\]

\(\triangleright\) By looking at \(\{x, y, z\}\) we see (using \(x = y\) and \(y = z\)) that \(x = z\).

\(\triangleright\) By looking at \(\{x, z, w\}\) we see (using \(x = z\) and \(z = w\)) that \(x = w\).

\(\triangleright\) By looking at \(\{x, w\}\) we now see a contradiction

In fact, \(\mathbb{A}\) has width \((2, 3)\), that is, such reasoning is always sufficient for an instance of \(\text{CSP}(\mathbb{A})\).
Bounded width

The problems $q$-LIN do not have bounded width
Feder, Vardi‘93
Bounded width

- The problems \( q\text{-LIN} \) do not have bounded width
  Feder, Vardi‘93
- If \( A \) can simulate \( q\text{-LIN} \) then \( A \) does not have bounded width
  Larose, Zádori’07
▶ The problems $q$-$LIN$ do not have bounded width
Feder, Vardi‘93

▶ If $A$ can simulate $q$-$LIN$ then $A$ does not have bounded width
Larose, Zádori’07

▶ Thus the “obvious” necessary condition for bounded width is that $A$ cannot simulate $q$-$LIN$. 
Bounded width

- The problems $q$-$LIN$ do not have bounded width
  Feder, Vardi‘93

- If $A$ can simulate $q$-$LIN$ then $A$ does not have bounded width
  Larose, Zádori’07

- Thus the “obvious” necessary condition for bounded width is that $A$ cannot simulate $q$-$LIN$.

- It is sufficient:
Bounded width

- The problems \( q\text{-LIN} \) do not have bounded width
  Feder, Vardi'93
- If \( A \) can simulate \( q\text{-LIN} \) then \( A \) does not have bounded width
  Larose, Zádori’07
- Thus the “obvious” necessary condition for bounded width is that \( A \) cannot simulate \( q\text{-LIN} \).
- It is sufficient:

  **Theorem**

  The following are equivalent.

  1. \( A \) cannot simulate \( q\text{-LIN} \)
  2. \( A \) has bounded width \( B \), Kozik’09
  3. \( A \) has width \((2, 3)\) \( B \); Bulatov
Task: Find an almost satisfying assignment given an almost satisfiable instance
Task: Find an almost satisfying assignment given an almost satisfiable instance.

More precisely: Find an assignment satisfying at least
\((1 - g(\varepsilon))\) fraction of the constraints given an instance which
is \((1 - \varepsilon)\) satisfiable, where \(g(\varepsilon) \to 0\) as \(\varepsilon \to 0\) (\(g\) should only
depend on \(A\)).
Task: Find an almost satisfying assignment given an almost satisfiable instance

More precisely: Find an assignment satisfying at least $(1 - g(\varepsilon))$ fraction of the constraints given an instance which is $(1 - \varepsilon)$ satisfiable, where $g(\varepsilon) \to 0$ as $\varepsilon \to 0$ ($g$ should only depend on $A$).

Algorithms for 2-SAT and HORN-SAT based on linear programming and semidefinite programming Zwick’98
Task: Find an almost satisfying assignment given an almost satisfiable instance

More precisely: Find an assignment satisfying at least $(1 - g(\varepsilon))$ fraction of the constraints given an instance which is $(1 - \varepsilon)$ satisfiable, where $g(\varepsilon) \to 0$ as $\varepsilon \to 0$ ($g$ should only depend on $\mathbb{A}$).

Algorithms for 2-SAT and HORN-SAT based on linear programming and semidefinite programming Zwick’98

$q$-LIN has no robust polynomial algorithm (assuming $P \neq NP$) Hastad’01
**Task:** Find an almost satisfying assignment given an almost satisfiable instance

More precisely: Find an assignment satisfying at least \((1 - g(\varepsilon))\) fraction of the constraints given an instance which is \((1 - \varepsilon)\) satisfiable, where \(g(\varepsilon) \to 0\) as \(\varepsilon \to 0\) (\(g\) should only depend on \(A\)).

- Algorithms for 2-SAT and HORNSAT based on linear programming and semidefinite programming [Zwick’98]
- \(q\)-LIN has no robust polynomial algorithm (assuming \(P \neq NP\)) [Hastad’01]
- If \(A\) can simulate \(q\)-LIN then CSP\((A)\) has no robust algorithm [Dalmau, Krokhin’11]
If $A$ can simulate $q$-LIN then $\text{CSP}(A)$ has no robust algorithm \cite{Dalmau, Krokhin’11}
If $A$ can simulate $q$-LIN then $\text{CSP}(A)$ has no robust algorithm [Dalmau, Krokhin’11]

Conjecture of Guruswami and Zhou: this is the only obstacle
If $A$ can simulate $q$-LIN then $\text{CSP}(A)$ has no robust algorithm Dalmau, Krokhin’11

Conjecture of Guruswami and Zhou: this is the only obstacle

**Theorem (B, Kozik’12)**

The following are equivalent (assuming $P \neq NP$)
- $A$ cannot simulate $q$-LIN
- $\text{CSP}(A)$ has a robust polynomial algorithm
- canonical semidefinite programming relaxation correctly decides $\text{CSP}(A)$
Bonus 2: Counting CSP

- The complexity is also controlled by $\text{Pol}(A)$
Bonus 2: Counting CSP

- The complexity is also controlled by $\text{Pol}(A)$
- A necessary condition for tractability found
  Bulatov, Dalmau’03
  (inspiration: the other algorithm for decision CSPs)
The complexity is also controlled by $\text{Pol}(\mathbb{A})$.

- A necessary condition for tractability found by Bulatov and Dalmau (2003) (inspiration: the other algorithm for decision CSPs).
- A stronger necessary condition for tractability found by Bulatov and Grohe (2005).
The complexity is also controlled by $\text{Pol}(\mathbb{A})$.

A necessary condition for tractability found: 
Bulatov, Dalmau'03  
(inspiration: the other algorithm for decision CSPs)

A stronger necessary condition for tractability found:
Bulatov, Grohe'05

The stronger condition is sufficient:
Bulatov'08, Dyer and Richerby'10
Final remarks

For decision CSPs

- Easy criterion for hardness
- Theory gives generic reduction between any two NP-complete CSPs (instead of ad hoc reductions)
- Applicability of known algorithms understood
- The dichotomy conjecture still open in general

For other variants (Approx-CSP, Valued CSP, infinite)

- Universal algebra also relevant (Cohen, Cooper, Creed, Jeavons, ˇZivn´ y; Raghavendra; Bodirsky, Pinsker)
- More or less the same criterion for easiness/hardness
- Easiness comes from "symmetry"
- One needs symmetry of higher arity (e.g. polymorphisms) rather than just automorphisms or endomorphisms

Beyond CSPs

- ???
- There is ≥1 examples (Raghavendra)
Final remarks

For decision CSPs
- Easy criterion for hardness
- Theory gives generic reduction between any two NP-complete CSPs (instead of ad hoc reductions)
- Applicability of known algorithms understood
- The dichotomy conjecture still open in general

For other variants (Approx-CSP, Valued CSP, infinite)
- Universal algebra also relevant Cohen, Cooper, Creed, Jeavons, Živný; Raghavendra; Bodirsky, Pinsker
- More or less the same criterion for easiness/hardness
- Easiness comes from “symmetry”
- One needs symmetry of higher arity (e.g. polymorphisms) rather than just automorphisms or endomorphisms
Final remarks

For decision CSPs
  ▶ Easy criterion for hardness
  ▶ Theory gives generic reduction between any two NP-complete CSPs (instead of ad hoc reductions)
  ▶ Applicability of known algorithms understood
  ▶ The dichotomy conjecture still open in general

For other variants (Approx-CSP, Valued CSP, infinite)
  ▶ Universal algebra also relevant Cohen, Cooper, Creed, Jeavons, Živný; Raghavendra; Bodirsky, Pinsker
  ▶ More or less the same criterion for easiness/hardness
  ▶ Easiness comes from “symmetry”
  ▶ One needs symmetry of higher arity (e.g. polymorphisms) rather than just automorphisms or endomorphisms

Beyond CSPs
  ▶ ???
  ▶ There is ≥ 1 examples Raghavendra
We need coffee!
Thank you!