# Proving inconsistency: Towards a better Maltsev CSP algorithm

Ross Willard

Univ. Waterloo

Universal Algebra and Lattice Theory
Szeged, Hungary
June 24, 2012

**Question**: What makes an algorithm (for a yes/no problem) "good"?

**Question**: What makes an algorithm (for a yes/no problem) "good"?

- It should be efficient (e.g., polynomial-time).

**Question**: What makes an algorithm (for a yes/no problem) "good"?

- It should be efficient (e.g., polynomial-time).
- It should be correct, i.e., always give correct answers.

**Question**: What makes an algorithm (for a yes/no problem) "good"?

- It should be efficient (e.g., polynomial-time).
- It should be correct, i.e., always give correct answers.
- It should be informative:
  - Provide a transparent "proof" of the correctness of the answer.

**Question**: What makes an algorithm (for a yes/no problem) "good"?

- It should be <u>efficient</u> (e.g., polynomial-time).
- It should be <u>correct</u>, i.e., always give correct answers.
- It should be <u>informative</u>:
  - Provide a transparent "proof" of the correctness of the answer.

In this lecture I will

- discuss the two main polynomial-time CSP algorithms,
- argue that one fails to meet the above criteria,
- offer a framework for a possible alternative.

# Motivating example

Fix a finite field $F$.

*Decision Problem*: **3-LIN($F$)**

**Inputs**:

a finite list $X = \{x_1, \ldots, x_n\}$ of variables

a finite list $\Sigma = \{\varepsilon_1, \ldots, \varepsilon_m\}$ of linear equations in $X$ over $F$

– each equation involving **at most 3 variables**

**Question**: Does $\Sigma$ have a solution (in $F$)?

# Motivating example (continued)

*Algorithm*: **Gaussian elimination**

Given a set $\Sigma$ of 3-variable linear equations in $n$ variables over $F$:

# Motivating example (continued)

*Algorithm*: **Gaussian elimination**

Given a set $\Sigma$ of 3-variable linear equations in $n$ variables over $F$:

- Methodically deduce new linear equations (satisfied by any solution).

# Motivating example (continued)

*Algorithm*: **Gaussian elimination**

Given a set $\Sigma$ of 3-variable linear equations in $n$ variables over $F$:

- Methodically deduce new linear equations (satisfied by any solution).
- If the inconsistent equation $0 = 1$ is deduced, then
  - $\Sigma$ is inconsistent, and
  - the deductions producing $0 = 1$ give a "short proof" of inconsistency.

# Motivating example (continued)

*Algorithm*: **Gaussian elimination**

Given a set $\Sigma$ of 3-variable linear equations in $n$ variables over $F$:

- Methodically deduce new linear equations (satisfied by any solution).
- If the inconsistent equation $0 = 1$ is deduced, then
  - $\Sigma$ is inconsistent, and
  - the deductions producing $0 = 1$ give a "short proof" of inconsistency.
- Else,
  - $\Sigma$ is consistent, and
  - "backtracking" produces an explicit solution of $\Sigma$, which is itself a (very) "short proof" of consistency.

# Motivating example (continued)

*Algorithm*: **Gaussian elimination**

Given a set $\Sigma$ of 3-variable linear equations in $n$ variables over $F$:

- Methodically deduce new linear equations (satisfied by any solution).
- If the inconsistent equation $0 = 1$ is deduced, then
  - $\Sigma$ is inconsistent, and
  - the deductions producing $0 = 1$ give a "short proof" of inconsistency.
- Else,
  - $\Sigma$ is consistent, and
  - "backtracking" produces an explicit solution of $\Sigma$, which is itself a (very) "short proof" of consistency.
- Running time: essentially $O(|\Sigma| n^2)$ arithmetic operations in $F$.

This is a *good algorithm*.

# Transition to CSP

Recall: an input to 3-LIN($F$) is a pair $(X, \Sigma)$ where

- $X = \{x_1, \dots, x_n\}$ is a finite list of variables.
- $\Sigma = \{\varepsilon_1, \dots, \varepsilon_m\}$ is a finite list of 3-variable equations over $F$.

Define

$$\mathbf{F} = (F, \{x - y + z\} \cup \{\lambda x + (1-\lambda)y \, : \, \lambda \in F\}),$$

the idempotent reduct of the vector space ${}_F F$.

# Transition to CSP

Recall: an input to 3-LIN($F$) is a pair $(X, \Sigma)$ where

- $X = \{x_1, \ldots, x_n\}$ is a finite list of variables.
- $\Sigma = \{\varepsilon_1, \ldots, \varepsilon_m\}$ is a finite list of 3-variable equations over $F$.

Define

$$\mathbf{F} = (F, \{x - y + z\} \cup \{\lambda x + (1 - \lambda)y \,:\, \lambda \in F\}),$$

the idempotent reduct of the vector space ${}_F F$.

**Observation**: if $S$ is the set of solutions to a 3-variable linear equation $\varepsilon$ over $F$, then $S$ is a subuniverse of $\mathbf{F}^3$.

**Hence**: each equation $ax_i + bx_j + cx_k = d$ can be expressed by the statement "$(x_i, x_j, x_k) \in S$" for some $S \leq \mathbf{F}^3$.

# Transition to CSP

Recall: an input to 3-LIN($F$) is a pair $(X, \Sigma)$ where

- $X = \{x_1, \ldots, x_n\}$ is a finite list of variables.
- $\Sigma = \{\varepsilon_1, \ldots, \varepsilon_m\}$ is a finite list of 3-variable equations over $F$.

Define

$$\mathbf{F} = (F, \{x - y + z\} \cup \{\lambda x + (1-\lambda)y \,:\, \lambda \in F\}),$$

the idempotent reduct of the vector space ${}_F F$.

**Observation**: if $S$ is the set of solutions to a 3-variable linear equation $\varepsilon$ over $F$, then $S$ is a subuniverse of $\mathbf{F}^3$.

**Hence**: each equation $ax_i + bx_j + cx_k = d$ can be expressed by the statement "$(x_i, x_j, x_k) \in S$" for some $S \leq \mathbf{F}^3$.

The **(fixed template) constraint satisfaction problem** generalizes 3-LIN($F$) by permitting $\mathbf{F}$ to be replaced by any idempotent algebra, equations by membership in named subpowers, and 3 by any fixed $d \geq 2$.

# Constraint Satisfaction Problem (CSP) definition

Formally, fix:

$\mathbf{A} = (A, \mathcal{F})$ – a finite *idempotent* algebra
$d \geq 2$

$\mathrm{CSP}(\mathbf{A}, d)$ is the following decision problem:

**Inputs**:
a finite list $X = \{x_1, \ldots, x_n\}$ of variables [ranging over $A$]
a finite list $\Sigma = \{C_1, \ldots, C_m\}$ of *constraints* on the variables:

Each constraint is a pair $C = (J, R)$ where

- $J \subseteq X \quad$ with $1 \leq |J| \leq d$;
- $R \leq \mathbf{A}^J$.

**Question**: Does $\Sigma$ have a *solution*?

(I.e., a map $\alpha : X \to A$ such that $\alpha{\restriction}_{J_t} \in R_t$ for all $1 \leq t \leq m$)

# CSP Algebraic Dichotomy Conjecture

**Conjecture (Bulatov, Jeavons, Krokhin)**

Let $\mathbf{A}$ be a finite idempotent algebra and $d \geq 2$.

If $V(\mathbf{A})$ satisfies a nontrivial Maltsev condition, then $\mathrm{CSP}(\mathbf{A}, d)$ is in P.

Of course, every $\mathrm{CSP}(\mathbf{A}, d)$ is in NP:

Any solution (when $\Sigma$ is satisfiable) is a "short proof" of satisfiability.

**What is wanted** (when $V(\mathbf{A})$ satisfies a nontrivial Maltsev condition):

# CSP Algebraic Dichotomy Conjecture

**Conjecture (Bulatov, Jeavons, Krokhin)**

Let $\mathbf{A}$ be a finite idempotent algebra and $d \geq 2$.

If $V(\mathbf{A})$ satisfies a nontrivial Maltsev condition, then $\mathrm{CSP}(\mathbf{A}, d)$ is in P.

Of course, every $\mathrm{CSP}(\mathbf{A}, d)$ is in NP:

Any solution (when $\Sigma$ is satisfiable) is a "short proof" of satisfiability.

**What is wanted** (when $V(\mathbf{A})$ satisfies a nontrivial Maltsev condition):

- "Short proofs" witnessing <u>un</u>satisfiability (when $\Sigma$ is unsatisfiable); they will put $\mathrm{CSP}(\mathbf{A}, d)$ in co-NP.

# CSP Algebraic Dichotomy Conjecture

**Conjecture (Bulatov, Jeavons, Krokhin)**

Let $\mathbf{A}$ be a finite idempotent algebra and $d \geq 2$.

If $V(\mathbf{A})$ satisfies a nontrivial Maltsev condition, then $\mathrm{CSP}(\mathbf{A}, d)$ is in P.

Of course, every $\mathrm{CSP}(\mathbf{A}, d)$ is in NP:

> Any solution (when $\Sigma$ is satisfiable) is a "short proof" of satisfiability.

**What is wanted** (when $V(\mathbf{A})$ satisfies a nontrivial Maltsev condition):

- "Short proofs" witnessing <u>un</u>satisfiability (when $\Sigma$ is unsatisfiable); they will put $\mathrm{CSP}(\mathbf{A}, d)$ in co-NP.
- Polynomial-time algorithm which decides $\mathrm{CSP}(\mathbf{A}, d)$ AND provides a solution or a short proof of unsatisfiability.

# The two main CSP algorithms

1. **Local consistency (bounded width) algorithm**
   - Rather simple
   - Works whenever $V(\mathbf{A})$ is congruence SD($\wedge$) [Barto & Kozik]
2. **Few subpowers algorithm**
   - Rather more complicated
   - Works whenever $V(\mathbf{A})$ is congruence modular [Barto? + IMMVW]
   - The case when $\mathbf{A}$ has a Maltsev operation is representative.

# Algorithm #1: Local consistency

Recall that constraints in an input to $\mathrm{CSP}(\mathbf{A}, d)$ have the form $(J, R)$:

- $J$ is a "small" subset of the set $X$ of variables ($|J| \leq d$).
- $R$ ($\leq \mathbf{A}^J$) restricts the values a solution may take on $J$.

The local consistency algorithm can be viewed as built upon a **formal system** for **reasoning** about such constraints.

# Algorithm #1: Local consistency

Recall that constraints in an input to $\mathrm{CSP}(\mathbf{A}, d)$ have the form $(J, R)$:

- $J$ is a "small" subset of the set $X$ of variables ($|J| \leq d$).
- $R$ ($\leq \mathbf{A}^J$) restricts the values a solution may take on $J$.

The local consistency algorithm can be viewed as built upon a **formal system** for **reasoning** about such constraints.

**Intuition**:

For some fixed $j < k$, the system will permit deducing a $\leq j$-ary constraint from a collection of other $\leq j$-ary constraints, as long as:

- the deduction is correct (of course!), and
- the number of variables altogether is at most $k$.

**Example**: if $(\mathbf{A}, d) = (\mathbf{F}, 3)$ and $(j, k) = (3, 6)$, then the system permits deductions of the following kind:

$$\begin{array}{rrcl}
\text{From} & x + y - u & = & 0 \\
 & y + z - v & = & 0 \\
 & u + z - w & = & 0 \\
\hline
\text{deduce} & x + v - w & = & 0
\end{array}
\quad \text{i.e.,} \quad
\begin{array}{c}
(\{x, y, u\}, \mathrm{graph}(+)) \\
(\{y, z, v\}, \mathrm{graph}(+)) \\
(\{u, z, w\}, \mathrm{graph}(+)) \\
\hline
(\{x, v, w\}, \mathrm{graph}(+))
\end{array}$$

Formally, the rules are (for some fixed $j < k$):

1. **Intersect**

$$\begin{array}{l} \qquad (J, R) \qquad (J, S) \\ \hline \therefore \quad (J, R \cap S) \end{array}$$

Formally, the rules are (for some fixed $j < k$):

1. **Intersect**

$$\frac{(J,R) \qquad (J,S)}{\therefore \quad (J, R\cap S)}$$

2. **FictVar**$_k$ – add fictitious variables, up to $k$ in total

$$\frac{(J,R)}{\therefore \quad (K, (\mathrm{pr}_{K\to J})^{-1}(R))}$$

for any $J \subseteq K \subseteq X$, provided $|K| \leq k$.

Formally, the rules are (for some fixed $j < k$):

1. **Intersect**

$$\begin{array}{rc} & (J,R) \qquad (J,S) \\ \hline \therefore & (J, R \cap S) \end{array}$$

2. **FictVar$_k$** – add fictitious variables, up to $k$ in total

$$\begin{array}{rc} & (J,R) \\ \hline \therefore & (K, (\mathrm{pr}_{K \to J})^{-1}(R)) \end{array}$$

   for any $J \subseteq K \subseteq X$, provided $|K| \leq k$.

3. **Project$_j$** – projection to $\leq j$ variables

$$\begin{array}{rc} & (K,R) \\ \hline \therefore & (J, \mathrm{pr}_{K \to J}(R)) \end{array}$$

   for any $J \subseteq K$, provided $|J| \leq j$.

These rules give a formal notion of proof.

**Definition**

Given an input $(X, \Sigma)$ to $\mathrm{CSP}(\mathbf{A}, d)$, a $(j,k)$-**proof from** $(X, \Sigma)$ is a finite sequence $(C_1, \ldots, C_p)$ of constraints over $X$ such that for all $1 \leq i \leq p$,

1. $C_i \in \Sigma$, or
2. $C_i$ is the result of applying **Intersect** to two constraints from $\{C_1, \ldots, C_{i-1}\}$, or
3. $C_i$ is the result of applying $\mathbf{FictVar}_k$ or $\mathbf{Project}_j$ to a constraint from $\{C_1, \ldots, C_{i-1}\}$.

I say that $(C_1, \ldots, C_p)$ is a $(j,k)$-proof **of** $C_p$ **from** $(X, \Sigma)$.

**Note**: every solution to $\Sigma$ also satisfies all $C_i$ in a $(j,k)$-proof from $(X, \Sigma)$.

## Notation

Let's write $(X, \Sigma) \vdash_{j,k} \varnothing$ if there exists a $(j, k)$-proof from $(X, \Sigma)$ whose last constraint is *empty* (i.e., has the form $(J, \varnothing)$).

**Remark**: if $(X, \Sigma) \vdash_{j,k} \varnothing$, then:

- $\Sigma$ is unsatisfiable.
- There exists a witnessing $(j, k)$-proof of length at most $2^{|A|^k} \cdot |X|^k$. (This is a good "short proof" of unsatisfiability.)

## Definition

$(\mathbf{A}, d)$ has **width (j,k)** if, for every instance $(X, \Sigma)$ of $\mathrm{CSP}(\mathbf{A}, d)$,

$$\Sigma \text{ unsatisfiable} \;\Leftrightarrow\; (X, \Sigma) \vdash_{j,k} \varnothing.$$

In other words, $(\mathbf{A}, d)$ has width $(j, k)$ if the formal system of $(j, k)$-proofs provides short proofs for all unsatisfiable instances to $\mathrm{CSP}(\mathbf{A}, d)$.

### Definition

$(\mathbf{A}, d)$ has **width (j,k)** if, for every instance $(X, \Sigma)$ of $\mathrm{CSP}(\mathbf{A}, d)$,

$$\Sigma \text{ unsatisfiable } \Leftrightarrow (X, \Sigma) \vdash_{j,k} \varnothing.$$

In other words, $(\mathbf{A}, d)$ has width $(j, k)$ if the formal system of $(j, k)$-proofs provides short proofs for all unsatisfiable instances to $\mathrm{CSP}(\mathbf{A}, d)$.

### Definition

$(\mathbf{A}, d)$ has **bounded width** if it has width $(j, k)$ for some $j < k$.

# Local consistency algorithm

**Folklore**: For each $j < k$ there is an algorithm (the "$(j,k)$-consistency algorithm") which, given $(\mathbf{A}, d)$ having width $(j,k)$ and given an input $(X, \Sigma)$ to $\mathrm{CSP}(\mathbf{A}, d)$,

# Local consistency algorithm

**Folklore**: For each $j < k$ there is an algorithm (the "$(j, k)$-consistency algorithm") which, given $(\mathbf{A}, d)$ having width $(j, k)$ and given an input $(X, \Sigma)$ to $\mathrm{CSP}(\mathbf{A}, d)$,

- decides whether $(X, \Sigma)$ has a solution.

# Local consistency algorithm

**Folklore**: For each $j < k$ there is an algorithm (the "$(j,k)$-consistency algorithm") which, given $(\mathbf{A}, d)$ having width $(j,k)$ and given an input $(X, \Sigma)$ to $\mathrm{CSP}(\mathbf{A}, d)$,

- decides whether $(X, \Sigma)$ has a solution.
- If satisfiable, produces a solution.

# Local consistency algorithm

**Folklore**: For each $j < k$ there is an algorithm (the "$(j, k)$-consistency algorithm") which, given $(\mathbf{A}, d)$ having width $(j, k)$ and given an input $(X, \Sigma)$ to $\mathrm{CSP}(\mathbf{A}, d)$,

- decides whether $(X, \Sigma)$ has a solution.
- If satisfiable, produces a solution.
- If unsatisfiable, produces a $(j, k)$-proof witnessing $(X, \Sigma) \vdash_{j,k} \varnothing$.

# Local consistency algorithm

**Folklore**: For each $j < k$ there is an algorithm (the "$(j,k)$-consistency algorithm") which, given $(\mathbf{A}, d)$ having width $(j,k)$ and given an input $(X, \Sigma)$ to $\mathrm{CSP}(\mathbf{A}, d)$,

- decides whether $(X, \Sigma)$ has a solution.
- If satisfiable, produces a solution.
- If unsatisfiable, produces a $(j,k)$-proof witnessing $(X, \Sigma) \vdash_{j,k} \varnothing$.
- Runs in polynomial time.

This is a *good algorithm*.

The extent of the local consistency algorithm:

**Theorem (Larose & Zádori ($\Rightarrow$); Barto & Kozik ($\Leftarrow$))**

*Let $\mathbf{A}$ be a finite idempotent algebra, $d \geq 2$, and assume the clone of $\mathbf{A}$ is determined by its $d$-ary invariant relations. Then*

$$(\mathbf{A}, d) \text{ has bounded width} \quad \Leftrightarrow \quad V(\mathbf{A}) \text{ is congruence } \mathrm{SD}(\wedge).$$

Unfortunately, if $\mathbf{F}$ is the idempotent algebra corresponding to 3-LIN($F$), then $(\mathbf{F}, 3)$ does *not* have bounded width.

**Conclusion**: although Gaussian elimination is a form of "constraint" reasoning, it does not fall within the framework of local consistency proofs.

# Algorithm #2: Few subpowers

Recall that each input to $\mathrm{CSP}(\mathbf{A}, d)$ has the form $(X, \Sigma)$ where

$$\Sigma = \{C_1, C_2, \ldots, C_m\} \qquad \text{with} \quad C_t = (J_t, R_t).$$

# Algorithm #2: Few subpowers

Recall that each input to $\mathrm{CSP}(\mathbf{A}, d)$ has the form $(X, \Sigma)$ where

$$\Sigma = \{C_1, C_2, \ldots, C_m\} \qquad \text{with} \quad C_t = (J_t, R_t).$$

For $i \leq m$, define $B_i$ to be the set of solutions to the first $i$ constraints:

$$\mathbf{A}^X = \mathbf{B}_0 \geq \mathbf{B}_1 \geq \mathbf{B}_2 \geq \cdots \geq \mathbf{B}_m = \{\text{solutions to } (X, \Sigma)\}.$$

# Algorithm #2: Few subpowers

Recall that each input to $\mathrm{CSP}(\mathbf{A}, d)$ has the form $(X, \Sigma)$ where

$$\Sigma = \{C_1, C_2, \ldots, C_m\} \qquad \text{with} \quad C_t = (J_t, R_t).$$

For $i \leq m$, define $B_i$ to be the set of solutions to the first $i$ constraints:

$$\mathbf{A}^X = \mathbf{B}_0 \geq \mathbf{B}_1 \geq \mathbf{B}_2 \geq \cdots \geq \mathbf{B}_m = \{\text{solutions to } (X, \Sigma)\}.$$

The *few subpowers algorithm* (BD + IMMVW):

- is *not* based on reasoning with equations/constraints.
- instead, it successively computes *small generating sets* for each $\mathbf{B}_t$.
  - $(X, \Sigma)$ has a solution $\Leftrightarrow$ the last generating set is nonempty.

# Special case: when $\mathbf{A}$ is Maltsev

(I.e., when $V(\mathbf{A})$ is congruence permutable.)

$\boxed{\text{Bulatov \& Dalmau, A simple algorithm for Mal'tsev constraints, 2006.}}$

Based on the notion of *compact representations* of subsets of powers.

**Definition**

Suppose $A$ is a set and $B \subseteq A^n$.

# Special case: when **A** is Maltsev

(I.e., when $V(\mathbf{A})$ is congruence permutable.)

Bulatov & Dalmau, A simple algorithm for Mal'tsev constraints, 2006.

Based on the notion of *compact representations* of subsets of powers.

## Definition

Suppose $A$ is a set and $B \subseteq A^n$.

$$\mathrm{Fork}(B) = \{(i,b,c) \in [n] \times A \times A : \exists \mathbf{u},\mathbf{v} \in B \text{ with } u_j = v_j \text{ for all } 1 \le j < i, \text{ and } (u_i,v_i) = (b,c)\}.$$

# Special case: when **A** is Maltsev

(I.e., when $V(\mathbf{A})$ is congruence permutable.)

Bulatov & Dalmau, A simple algorithm for Mal'tsev constraints, 2006.

Based on the notion of *compact representations* of subsets of powers.

### Definition

Suppose $A$ is a set and $B \subseteq A^n$.

$$\mathrm{Fork}(B) = \{(i,b,c) \in [n] \times A \times A : \exists \mathbf{u},\mathbf{v} \in B \text{ with } u_j = v_j \text{ for all } 1 \le j < i, \text{ and } (u_i,v_i) = (b,c)\}.$$

A subset $T \subseteq B$ is called a **compact representation of** $B$ if $\mathrm{Fork}(T) = \mathrm{Fork}(B)$ and $T$ is minimal with respect to this property.

# Special case: when **A** is Maltsev

(I.e., when $V(\mathbf{A})$ is congruence permutable.)

Bulatov & Dalmau, A simple algorithm for Mal'tsev constraints, 2006.

Based on the notion of *compact representations* of subsets of powers.

## Definition

Suppose $A$ is a set and $B \subseteq A^n$.

$$\mathrm{Fork}(B) \;=\; \{(i,b,c) \in [n] \times A \times A : \exists \mathbf{u},\mathbf{v} \in B \text{ with } u_j = v_j \text{ for all } 1 \le j < i, \text{ and } (u_i,v_i) = (b,c)\}.$$

A subset $T \subseteq B$ is called a **compact representation of** $B$ if $\mathrm{Fork}(T) = \mathrm{Fork}(B)$ and $T$ is minimal with respect to this property.

**Exercise**: if $T$ is a compact rep. for $B \subseteq A^n$, then $|T| \le n|A|^2$.

## Key Fact (Bulatov, Dalmau)

*If* $\mathbf{A}$ *has a Maltsev term,* $\mathbf{B} \leq \mathbf{A}^n$*, and* $T$ *is a compact representation of* $B$*, then* $T$ *generates* $\mathbf{B}$*.*

## Proof idea.

## Key Fact (Bulatov, Dalmau)

*If* $\mathbf{A}$ *has a Maltsev term,* $\mathbf{B} \leq \mathbf{A}^n$*, and* $T$ *is a compact representation of* $B$*, then* $T$ *generates* $\mathbf{B}$*.*

## Proof idea.

Suppose $\mathrm{pr}_{1,\ldots,i-1}(\langle T \rangle_{\mathbf{B}}) = \mathrm{pr}_{1,\ldots,i-1}(B)$.
We will show $\mathrm{pr}_{1,\ldots,i}(\langle T \rangle_{\mathbf{B}}) = \mathrm{pr}_{1,\ldots,i}(B)$.

## Key Fact (Bulatov, Dalmau)

*If* $\mathbf{A}$ *has a Maltsev term,* $\mathbf{B} \leq \mathbf{A}^n$*, and* $T$ *is a compact representation of* $B$*, then* $T$ *generates* $\mathbf{B}$*.*

## Proof idea.

Suppose $\mathrm{pr}_{1,\dots,i-1}(\langle T\rangle_{\mathbf{B}}) = \mathrm{pr}_{1,\dots,i-1}(B)$.
We will show $\mathrm{pr}_{1,\dots,i}(\langle T\rangle_{\mathbf{B}}) = \mathrm{pr}_{1,\dots,i}(B)$.
Pick $\mathbf{a} = (a_1, \dots, a_{i-1}, a_i, \dots) \in B$.

## Key Fact (Bulatov, Dalmau)

*If* $\mathbf{A}$ *has a Maltsev term,* $\mathbf{B} \leq \mathbf{A}^n$*, and* $T$ *is a compact representation of* $B$*, then* $T$ *generates* $\mathbf{B}$*.*

## Proof idea.

Suppose $\mathrm{pr}_{1,\dots,i-1}(\langle T \rangle_{\mathbf{B}}) = \mathrm{pr}_{1,\dots,i-1}(B)$.
We will show $\mathrm{pr}_{1,\dots,i}(\langle T \rangle_{\mathbf{B}}) = \mathrm{pr}_{1,\dots,i}(B)$.
Pick $\mathbf{a} = (a_1, \dots, a_{i-1}, a_i, \dots) \in B$.
So $\exists \mathbf{a}' = (a_1, \dots, a_{i-1}, b, \dots) \in \langle T \rangle_{\mathbf{B}}$. (Thus also $\mathbf{a}' \in B$.)

## Key Fact (Bulatov, Dalmau)

*If* $\mathbf{A}$ *has a Maltsev term,* $\mathbf{B} \leq \mathbf{A}^n$*, and* $T$ *is a compact representation of* $B$*, then* $T$ *generates* $\mathbf{B}$*.*

## Proof idea.

Suppose $\mathrm{pr}_{1,\dots,i-1}(\langle T \rangle_{\mathbf{B}}) = \mathrm{pr}_{1,\dots,i-1}(B)$.
We will show $\mathrm{pr}_{1,\dots,i}(\langle T \rangle_{\mathbf{B}}) = \mathrm{pr}_{1,\dots,i}(B)$.
Pick $\mathbf{a} = (a_1, \dots, a_{i-1}, a_i, \dots) \in B$.
So $\exists \mathbf{a}' = (a_1, \dots, a_{i-1}, b, \dots) \in \langle T \rangle_{\mathbf{B}}$. (Thus also $\mathbf{a}' \in B$.)
Thus $(i, a_i, b) \in \mathrm{Fork}(B)$

## Key Fact (Bulatov, Dalmau)

*If* $\mathbf{A}$ *has a Maltsev term,* $\mathbf{B} \leq \mathbf{A}^n$*, and* $T$ *is a compact representation of* $B$*, then* $T$ *generates* $\mathbf{B}$*.*

## Proof idea.

Suppose $\mathrm{pr}_{1,\ldots,i-1}(\langle T \rangle_{\mathbf{B}}) = \mathrm{pr}_{1,\ldots,i-1}(B)$.
We will show $\mathrm{pr}_{1,\ldots,i}(\langle T \rangle_{\mathbf{B}}) = \mathrm{pr}_{1,\ldots,i}(B)$.
Pick $\mathbf{a} = (a_1, \ldots, a_{i-1}, a_i, \ldots) \in B$.
So $\exists \mathbf{a}' = (a_1, \ldots, a_{i-1}, b, \ldots) \in \langle T \rangle_{\mathbf{B}}$. (Thus also $\mathbf{a}' \in B$.)
Thus $(i, a_i, b) \in \mathrm{Fork}(B) = \mathrm{Fork}(T)$.

## Key Fact (Bulatov, Dalmau)

*If* $\mathbf{A}$ *has a Maltsev term,* $\mathbf{B} \leq \mathbf{A}^n$*, and* $T$ *is a compact representation of* $B$*, then* $T$ *generates* $\mathbf{B}$*.*

## Proof idea.

Suppose $\mathrm{pr}_{1,\ldots,i-1}(\langle T \rangle_{\mathbf{B}}) = \mathrm{pr}_{1,\ldots,i-1}(B)$.
We will show $\mathrm{pr}_{1,\ldots,i}(\langle T \rangle_{\mathbf{B}}) = \mathrm{pr}_{1,\ldots,i}(B)$.
Pick $\mathbf{a} = (a_1, \ldots, a_{i-1}, a_i, \ldots) \in B$.
So $\exists \mathbf{a}' = (a_1, \ldots, a_{i-1}, b, \ldots) \in \langle T \rangle_{\mathbf{B}}$. (Thus also $\mathbf{a}' \in B$.)
Thus $(i, a_i, b) \in \mathrm{Fork}(B) = \mathrm{Fork}(T)$.
Pick $\mathbf{u}, \mathbf{v} \in T$ witnessing this.

## Key Fact (Bulatov, Dalmau)

*If* $\mathbf{A}$ *has a Maltsev term,* $\mathbf{B} \leq \mathbf{A}^n$*, and* $T$ *is a compact representation of* $B$*, then* $T$ *generates* $\mathbf{B}$*.*

## Proof idea.

Suppose $\mathrm{pr}_{1,\ldots,i-1}(\langle T \rangle_{\mathbf{B}}) = \mathrm{pr}_{1,\ldots,i-1}(B)$.
We will show $\mathrm{pr}_{1,\ldots,i}(\langle T \rangle_{\mathbf{B}}) = \mathrm{pr}_{1,\ldots,i}(B)$.
Pick $\mathbf{a} = (a_1, \ldots, a_{i-1}, a_i, \ldots) \in B$.
So $\exists \mathbf{a}' = (a_1, \ldots, a_{i-1}, b, \ldots) \in \langle T \rangle_{\mathbf{B}}$. (Thus also $\mathbf{a}' \in B$.)
Thus $(i, a_i, b) \in \mathrm{Fork}(B) = \mathrm{Fork}(T)$.
Pick $\mathbf{u}, \mathbf{v} \in T$ witnessing this.
We have

$$\begin{aligned} \mathbf{u} &= (u_1, \ldots, u_{i-1}, a_i, \ldots) \in T \\ \mathbf{v} &= (u_1, \ldots, u_{i-1}, b, \ldots) \in T \end{aligned}$$

## Key Fact (Bulatov, Dalmau)

*If* $\mathbf{A}$ *has a Maltsev term,* $\mathbf{B} \leq \mathbf{A}^n$, *and* $T$ *is a compact representation of* $B$, *then* $T$ *generates* $\mathbf{B}$.

## Proof idea.

Suppose $\mathrm{pr}_{1,\ldots,i-1}(\langle T\rangle_{\mathbf{B}}) = \mathrm{pr}_{1,\ldots,i-1}(B)$.
We will show $\mathrm{pr}_{1,\ldots,i}(\langle T\rangle_{\mathbf{B}}) = \mathrm{pr}_{1,\ldots,i}(B)$.
Pick $\mathbf{a} = (a_1, \ldots, a_{i-1}, a_i, \ldots) \in B$.
So $\exists \mathbf{a}' = (a_1, \ldots, a_{i-1}, b, \ldots) \in \langle T\rangle_{\mathbf{B}}$. (Thus also $\mathbf{a}' \in B$.)
Thus $(i, a_i, b) \in \mathrm{Fork}(B) = \mathrm{Fork}(T)$.
Pick $\mathbf{u}, \mathbf{v} \in T$ witnessing this.
We have

$$\begin{aligned} \mathbf{u} &= (u_1, \ldots, u_{i-1}, a_i, \ldots) && \in T \\ \mathbf{v} &= (u_1, \ldots, u_{i-1}, b, \ldots) && \in T \\ \mathbf{a}' &= (a_1, \ldots, a_{i-1}, b, \ldots) && \in \langle T\rangle_{\mathbf{B}}. \end{aligned}$$

## Key Fact (Bulatov, Dalmau)

*If* $\mathbf{A}$ *has a Maltsev term,* $\mathbf{B} \leq \mathbf{A}^n$*, and* $T$ *is a compact representation of* $B$*, then* $T$ *generates* $\mathbf{B}$*.*

## Proof idea.

Suppose $\mathrm{pr}_{1,\ldots,i-1}(\langle T \rangle_{\mathbf{B}}) = \mathrm{pr}_{1,\ldots,i-1}(B)$.
We will show $\mathrm{pr}_{1,\ldots,i}(\langle T \rangle_{\mathbf{B}}) = \mathrm{pr}_{1,\ldots,i}(B)$.
Pick $\mathbf{a} = (a_1, \ldots, a_{i-1}, a_i, \ldots) \in B$.
So $\exists \mathbf{a}' = (a_1, \ldots, a_{i-1}, b, \ldots) \in \langle T \rangle_{\mathbf{B}}$. (Thus also $\mathbf{a}' \in B$.)
Thus $(i, a_i, b) \in \mathrm{Fork}(B) = \mathrm{Fork}(T)$.
Pick $\mathbf{u}, \mathbf{v} \in T$ witnessing this.
We have

$$\begin{aligned} \mathbf{u} &= (u_1, \ldots, u_{i-1}, a_i, \ldots) \ \in T \\ \mathbf{v} &= (u_1, \ldots, u_{i-1}, b, \ldots) \ \in T \\ \mathbf{a}' &= (a_1, \ldots, a_{i-1}, b, \ldots) \ \in \langle T \rangle_{\mathbf{B}}. \end{aligned}$$

Applying the Maltsev term, we get $(a_1, \ldots, a_{i-1}, a_i, \ldots) \in \langle T \rangle_{\mathbf{B}}$. $\square$

**The BD Algorithm**: Let $(X, \Sigma)$ with $\Sigma = (C_1, \ldots, C_m)$ be an input to $\mathrm{CSP}(\mathbf{A}, d)$ with $\mathbf{A}$ Maltsev. Linearly order $X = \{x_1, \ldots, x_n\}$, identify $A^X$ with $A^n$, and recall the descending chain of subpowers given by $C_1, \ldots, C_m$:

$$\mathbf{A}^n = \mathbf{B}_0 \geq \mathbf{B}_1 \geq \mathbf{B}_2 \geq \cdots \geq \mathbf{B}_m = \{\text{solutions to } (X, \Sigma)\}. \qquad (\dagger)$$

**The BD Algorithm**: Let $(X, \Sigma)$ with $\Sigma = (C_1, \ldots, C_m)$ be an input to $\mathrm{CSP}(\mathbf{A}, d)$ with $\mathbf{A}$ Maltsev. Linearly order $X = \{x_1, \ldots, x_n\}$, identify $A^X$ with $A^n$, and recall the descending chain of subpowers given by $C_1, \ldots, C_m$:

$$\mathbf{A}^n = \mathbf{B}_0 \geq \mathbf{B}_1 \geq \mathbf{B}_2 \geq \cdots \geq \mathbf{B}_m = \{\text{solutions to } (X, \Sigma)\}. \qquad (\dagger)$$

Recall: we want to compute compact representations for $B_1, B_2, \ldots, B_m$.

**The BD Algorithm**: Let $(X,\Sigma)$ with $\Sigma = (C_1,\ldots,C_m)$ be an input to $\mathrm{CSP}(\mathbf{A},d)$ with $\mathbf{A}$ Maltsev. Linearly order $X=\{x_1,\ldots,x_n\}$, identify $A^X$ with $A^n$, and recall the descending chain of subpowers given by $C_1,\ldots,C_m$:

$$\mathbf{A}^n = \mathbf{B}_0 \geq \mathbf{B}_1 \geq \mathbf{B}_2 \geq \cdots \geq \mathbf{B}_m = \{\text{solutions to } (X,\Sigma)\}. \qquad (\dagger)$$

Recall: we want to compute compact representations for $B_1, B_2, \ldots, B_m$.

[Relaxation: $\mathbf{B}_0 \leq \mathbf{A}^n$; require a compact representation for $B_0$ as input.]

**The BD Algorithm**: Let $(X, \Sigma)$ with $\Sigma = (C_1, \ldots, C_m)$ be an input to $\mathrm{CSP}(\mathbf{A}, d)$ with $\mathbf{A}$ Maltsev. Linearly order $X = \{x_1, \ldots, x_n\}$, identify $A^X$ with $A^n$, and recall the descending chain of subpowers given by $C_1, \ldots, C_m$:

$$\mathbf{A}^n \geq \mathbf{B}_0 \geq \mathbf{B}_1 \geq \mathbf{B}_2 \geq \cdots \geq \mathbf{B}_m = \{\text{solutions to } (X, \Sigma)\} \cap B_0. \qquad (\dagger)$$

Recall: we want to compute compact representations for $B_1, B_2, \ldots, B_m$.

[Relaxation: $\mathbf{B}_0 \leq \mathbf{A}^n$; require a compact representation for $B_0$ as input.]

**The BD Algorithm**: Let $(X, \Sigma)$ with $\Sigma = (C_1, \dots, C_m)$ be an input to $\mathrm{CSP}(\mathbf{A}, d)$ with $\mathbf{A}$ Maltsev. Linearly order $X = \{x_1, \dots, x_n\}$, identify $A^X$ with $A^n$, and recall the descending chain of subpowers given by $C_1, \dots, C_m$:

$$\mathbf{A}^n \geq \mathbf{B}_0 \geq \mathbf{B}_1 \geq \mathbf{B}_2 \geq \cdots \geq \mathbf{B}_m = \{\text{solutions to } (X, \Sigma)\} \cap B_0. \qquad (\dagger)$$

Recall: we want to compute compact representations for $B_1, B_2, \dots, B_m$.

[Relaxation: $\mathbf{B}_0 \leq \mathbf{A}^n$; require a compact representation for $B_0$ as input.]

**Special Case**: Show that comp. rep's can be found in the case $m < n$ and $\exists\ a_1, \dots, a_m \in A$ such that $C_t =$ "$x_t = a_t$", i.e., $(x_t, \{a_t\})$, $\forall\ 1 \leq t \leq m$.

**The BD Algorithm**: Let $(X, \Sigma)$ with $\Sigma = (C_1, \ldots, C_m)$ be an input to $\mathrm{CSP}(\mathbf{A}, d)$ with $\mathbf{A}$ Maltsev. Linearly order $X = \{x_1, \ldots, x_n\}$, identify $A^X$ with $A^n$, and recall the descending chain of subpowers given by $C_1, \ldots, C_m$:

$$\mathbf{A}^n \geq \mathbf{B}_0 \geq \mathbf{B}_1 \geq \mathbf{B}_2 \geq \cdots \geq \mathbf{B}_m = \{\text{solutions to } (X, \Sigma)\} \cap B_0. \qquad (\dagger)$$

Recall: we want to compute compact representations for $B_1, B_2, \ldots, B_m$.

[Relaxation: $\mathbf{B}_0 \leq \mathbf{A}^n$; require a compact representation for $B_0$ as input.]

**Special Case**: Show that comp. rep's can be found in the case $m < n$ and $\exists\ a_1, \ldots, a_m \in A$ such that $C_t =$ "$x_t = a_t$", i.e., $(x_t, \{a_t\})$, $\forall\ 1 \leq t \leq m$.

Now in general, we want to compute a compact representation for $B_t$, given a compact representation for $B_{t-1}$ and the constraint $C_t$.

**Key task**: For each $(i, a, b) \in [n] \times A \times A$, we need to decide whether $(i, a, b) \in \mathrm{Fork}(B_t)$ and, if "yes," we must find a witnessing pair $\mathbf{u}, \mathbf{v} \in B_t$.

**The BD Algorithm**: Let $(X, \Sigma)$ with $\Sigma = (C_1, \ldots, C_m)$ be an input to $\mathrm{CSP}(\mathbf{A}, d)$ with $\mathbf{A}$ Maltsev. Linearly order $X = \{x_1, \ldots, x_n\}$, identify $A^X$ with $A^n$, and recall the descending chain of subpowers given by $C_1, \ldots, C_m$:

$$\mathbf{A}^n \geq \mathbf{B}_0 \geq \mathbf{B}_1 \geq \mathbf{B}_2 \geq \cdots \geq \mathbf{B}_m = \{\text{solutions to } (X, \Sigma)\} \cap B_0. \qquad (\dagger)$$

Recall: we want to compute compact representations for $B_1, B_2, \ldots, B_m$.

[Relaxation: $\mathbf{B}_0 \leq \mathbf{A}^n$; require a compact representation for $B_0$ as input.]

**Special Case**: Show that comp. rep's can be found in the case $m < n$ and $\exists\ a_1, \ldots, a_m \in A$ such that $C_t =$ "$x_t = a_t$", i.e., $(x_t, \{a_t\})$, $\forall\ 1 \leq t \leq m$.

Now in general, we want to compute a compact representation for $B_t$, given a compact representation for $B_{t-1}$ and the constraint $C_t$.

**Key task**: For each $(i, a, b) \in [n] \times A \times A$, we need to decide whether $(i, a, b) \in \mathrm{Fork}(B_t)$ and, if "yes," we must find a witnessing pair $\mathbf{u}, \mathbf{v} \in B_t$.

Finding a candidate $\mathbf{u}$ is not too hard. To find $\mathbf{v}$, construct a new chain $(\dagger)$ of subpowers in the special case, starting from $\mathbf{B}_{t-1}$, using $u_1, \ldots, u_{n-1}$.

# The few subpowers algorithm and its extent

- A necessary feature of the BD algorithm is that $\exists$ polynomial $p(x)$ such that every $\mathbf{B} \leq \mathbf{A}^n$ has a generating set of size at most $p(n)$.

# The few subpowers algorithm and its extent

- A necessary feature of the BD algorithm is that $\exists$ polynomial $p(x)$ such that every $\mathbf{B} \leq \mathbf{A}^n$ has a generating set of size at most $p(n)$.
- BIMMVW characterize such $\mathbf{A}$; they are said to *have few subpowers* and are characterized by having a *cube* term (or *edge* term).

# The few subpowers algorithm and its extent

- A necessary feature of the BD algorithm is that $\exists$ polynomial $p(x)$ such that every $\mathbf{B} \leq \mathbf{A}^n$ has a generating set of size at most $p(n)$.
- BIMMVW characterize such $\mathbf{A}$; they are said to *have few subpowers* and are characterized by having a *cube* term (or *edge* term).
- An analogous notion of *compact representation* is given for such $\mathbf{A}$.

# The few subpowers algorithm and its extent

- A necessary feature of the BD algorithm is that $\exists$ polynomial $p(x)$ such that every $\mathbf{B} \leq \mathbf{A}^n$ has a generating set of size at most $p(n)$.
- BIMMVW characterize such $\mathbf{A}$; they are said to *have few subpowers* and are characterized by having a *cube* term (or *edge* term).
- An analogous notion of *compact representation* is given for such $\mathbf{A}$.
- The Bulatov-Dalmau algorithm generalizes to algebras having a cube term (IMMVW); called the *few subpowers algorithm*.

# The few subpowers algorithm and its extent

- A necessary feature of the BD algorithm is that $\exists$ polynomial $p(x)$ such that every $\mathbf{B} \leq \mathbf{A}^n$ has a generating set of size at most $p(n)$.
- BIMMVW characterize such $\mathbf{A}$; they are said to *have few subpowers* and are characterized by having a *cube* term (or *edge* term).
- An analogous notion of *compact representation* is given for such $\mathbf{A}$.
- The Bulatov-Dalmau algorithm generalizes to algebras having a cube term (IMMVW); called the *few subpowers algorithm*.
- With Barto's recently announced result, we know that (assuming $\mathbf{A}$ is determined by its $d$-ary relations),

$$\mathbf{A} \text{ has a cube term} \quad \Leftrightarrow \quad V(\mathbf{A}) \text{ is congruence modular.}$$

## Confession

I have a love/hate relationship with the few subpowers algorithm.

## Confession

I have a love/hate relationship with the few subpowers algorithm.

Why I love it:

- It works (when $\mathbf{A}$ has a cube term).
- It runs in polynomial time.
- It gave me two publications (W = Willard).

## Confession

I have a love/hate relationship with the few subpowers algorithm.

Why I love it:

- It works (when $\mathbf{A}$ has a cube term).
- It runs in polynomial time.
- It gave me two publications (W = Willard).

Why I hate it:

- It cannot be executed in the absence of a cube term.
- It does not exploit structure theory of congruence modular varieties.
- It does not give "nice" short proofs of unsatisfiability.
  - (Local consistency is *so* much better!)

## Confession

I have a love/hate relationship with the few subpowers algorithm.

Why I love it:

- It works (when **A** has a cube term).
- It runs in polynomial time.
- It gave me two publications (W = Willard).

Why I hate it:

- It cannot be executed in the absence of a cube term.
- It does not exploit structure theory of congruence modular varieties.
- It does not give "nice" short proofs of unsatisfiability.
  - (Local consistency is *so* much better!)

**Problem**: Are we stuck with it? Can we find a better algorithm?

# An idea for a new type of "short proof" of unsatisfiability

**Motivating example:** again 3-LIN($F$)

**The sad fact**: Unsatisfiable instances of 3-LIN($F$) cannot be proved to be unsatisfiable by local consistency.

**The happy fact**: Unsatisfiable instances of 3-LIN($F$) *can* be proved to be unsatisfiable by local consistency. . .

# An idea for a new type of “short proof” of unsatisfiability

**Motivating example:** again 3-LIN($F$)

**The sad fact**: Unsatisfiable instances of 3-LIN($F$) cannot be proved to be unsatisfiable by local consistency.

**The happy fact**: Unsatisfiable instances of 3-LIN($F$) *can* be proved to be unsatisfiable by local consistency. . . provided one is permitted the **introduction of new variables**.

Suppose an instance $(X, \Sigma)$ of 3-LIN$(F)$ is given.

Suppose some new variables $u_1, \ldots, u_L$ are "introduced" (i.e., defined) by $\leq$ 3-variable equations, say

$$\begin{aligned} u_1 &:= ax_5 + 1 \\ u_2 &:= bx_3 + cx_6 \\ u_3 &:= ru_1 + su_2 + 3 \\ &\vdots \end{aligned}$$

Suppose an instance $(X, \Sigma)$ of 3-LIN$(F)$ is given.

Suppose some new variables $u_1, \ldots, u_L$ are "introduced" (i.e., defined) by $\leq$ 3-variable equations, say

$$\begin{array}{rcl} u_1 & := & ax_5 + 1 \\ u_2 & := & bx_3 + cx_6 \\ u_3 & := & ru_1 + su_2 + 3 \\ & \vdots & \end{array}$$

Let $U$ be the set of new variables and let $\Gamma$ be the set of defining equations.

Clearly $(X, \Sigma)$ is satisfiable if and only if $(X \cup U,\, \Sigma \cup \Gamma)$ is satisfiable.

## Theorem

*Suppose* $(X, \Sigma)$ *is an instance of 3-LIN*$(F)$*, with* $|X| = n$ *and* $|\Sigma| = m$*. If* $\Sigma$ *is unsatisfiable, then there exists*

- $L \leq mn(m+n)$,
- *a set* $U = \{u_t \ : \ 1 \leq t \leq L\}$ *of* $L$ *new variables,*
- *a set* $\Gamma = \{\gamma_t \ : \ 1 \leq t \leq L\}$ *of* $L$ *linear equations where each* $\gamma_t$ *defines* $u_t$ *as a function of* $\leq 2$ *variables from* $X \cup \{u_1, \ldots, u_{t-1}\}$*,*

*such that* $(X \cup U, \, \Sigma \cup \Gamma) \vdash_{3,6} \varnothing$*.*

## Theorem

*Suppose $(X, \Sigma)$ is an instance of 3-LIN$(F)$, with $|X| = n$ and $|\Sigma| = m$. If $\Sigma$ is unsatisfiable, then there exists*

- $L \leq mn(m+n)$,
- *a set* $U = \{u_t : 1 \leq t \leq L\}$ *of* $L$ *new variables,*
- *a set* $\Gamma = \{\gamma_t : 1 \leq t \leq L\}$ *of* $L$ *linear equations where each* $\gamma_t$ *defines* $u_t$ *as a function of* $\leq 2$ *variables from* $X \cup \{u_1, \ldots, u_{t-1}\}$,

*such that* $(X \cup U, \Sigma \cup \Gamma) \vdash_{3,6} \varnothing$.

## Proof hint: Simulate Gaussian elimination.

Linearly order $X = \{x_1, x_2 \ldots, x_n\}$; run GE. For each "complete" equation $a_1x_1 + \cdots + a_nx_n = b$ occurring in the GE computation, introduce $n$ new variables representing the partial sums of the left-hand side:

$$u_1 := a_1x_1, \quad u_2 := u_1 + a_2x_2, \quad \ldots, \quad u_n = u_{n-1} + a_nx_n. \quad \text{(Gives } U, \Gamma.)$$

Show that for each such equation, $(X \cup U, \Sigma \cup \Gamma) \vdash_{3,6}$ "$u_n = b$." □

# Formalize and Generalize:

Fix $j < k$ and $\mathbf{A}$. Also fix $\ell_0, \ell_1$ satisfying $\ell_0 \leq j$ and $\ell_0 + \ell_1 \leq k$.

# Formalize and Generalize:

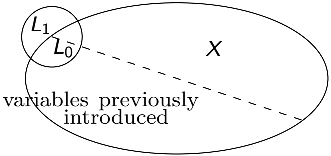
Fix $j < k$ and $\mathbf{A}$. Also fix $\ell_0, \ell_1$ satisfying $\ell_0 \leq j$ and $\ell_0 + \ell_1 \leq k$.

To the rules **Intersect**, **Project**$_j$ and **FictVar**$_k$ for $(j,k)$-proofs, add:

4. **VarIntro**$_{\ell_0,\ell_1}$

$$\frac{(L_0, R)}{\therefore \quad (L_0 \cup L_1, S)}$$

provided



# Formalize and Generalize:

Fix $j < k$ and $\mathbf{A}$. Also fix $\ell_0, \ell_1$ satisfying $\ell_0 \leq j$ and $\ell_0 + \ell_1 \leq k$.

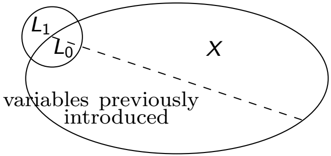
To the rules **Intersect**, $\mathbf{Project}_j$ and $\mathbf{FictVar}_k$ for $(j, k)$-proofs, add:

4. $\mathbf{VarIntro}_{\ell_0,\ell_1}$

$$\frac{(L_0, R)}{\therefore \quad (L_0 \cup L_1, S)}$$

provided

- The variables in $L_1$ are **new**. (This rule **introduces** them.)



# Formalize and Generalize:

Fix $j < k$ and $\mathbf{A}$. Also fix $\ell_0, \ell_1$ satisfying $\ell_0 \leq j$ and $\ell_0 + \ell_1 \leq k$.

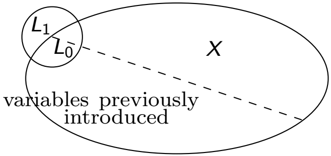
To the rules **Intersect**, $\mathbf{Project}_j$ and $\mathbf{FictVar}_k$ for $(j,k)$-proofs, add:

4. $\mathbf{VarIntro}_{\ell_0,\ell_1}$

$$\frac{(L_0, R)}{\therefore \quad (L_0 \cup L_1, S)}$$

provided

- The variables in $L_1$ are **new**. (This rule **introduces** them.)
- $R \subseteq \mathrm{pr}_{L_0 \cup L_1 \to L_0}(S)$.



# Formalize and Generalize:

Fix $j < k$ and **A**. Also fix $\ell_0, \ell_1$ satisfying $\ell_0 \leq j$ and $\ell_0 + \ell_1 \leq k$.

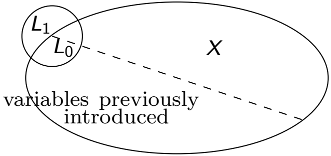
To the rules **Intersect**, **Project**$_j$ and **FictVar**$_k$ for $(j,k)$-proofs, add:

4. **VarIntro**$_{\ell_0,\ell_1}$

$$\frac{(L_0, R)}{\therefore \quad (L_0 \cup L_1, S)}$$

provided

- The variables in $L_1$ are **new**. (This rule **introduces** them.)
- $R \subseteq \mathrm{pr}_{L_0 \cup L_1 \to L_0}(S)$.
- $|L_i| \leq \ell_i$ for $i = 0, 1$.



# Formalize and Generalize:

Fix $j < k$ and $\mathbf{A}$. Also fix $\ell_0, \ell_1$ satisfying $\ell_0 \leq j$ and $\ell_0 + \ell_1 \leq k$.

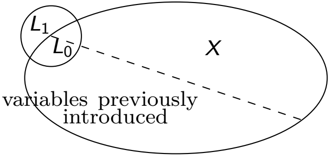
To the rules **Intersect**, $\mathbf{Project}_j$ and $\mathbf{FictVar}_k$ for $(j,k)$-proofs, add:

4. $\mathbf{VarIntro}_{\ell_0,\ell_1}$

$$\frac{(L_0, R)}{\therefore \quad (L_0 \cup L_1, S)}$$

provided

- The variables in $L_1$ are **new**. (This rule **introduces** them.)
- $R \subseteq \mathrm{pr}_{L_0 \cup L_1 \to L_0}(S)$.
- $|L_i| \leq \ell_i$ for $i = 0, 1$.
- $S \leq \mathbf{A}^{L_0 \cup L_1}$.



Using these four rules, we get a notion of "$(j, k; \ell_0, \ell_1; \mathbf{A})$-proof."

## Notation

If $(X, \Sigma)$ is an instance of $\mathrm{CSP}(\mathbf{A}, d)$, let's write

$$(\mathbf{A}, X, \Sigma) \Vdash^{N}_{j,k;\ell_0,\ell_1} \varnothing$$

if there exists a $(j, k; \ell_0, \ell_1; \mathbf{A})$-proof from $(X, \Sigma)$ whose last constraint is empty, and which introduces at most $N$ new variables.

Using these four rules, we get a notion of "$(j,k;\ell_0,\ell_1;\mathbf{A})$-proof."

## Notation

If $(X,\Sigma)$ is an instance of $\mathrm{CSP}(\mathbf{A},d)$, let's write

$$(\mathbf{A},X,\Sigma) \Vdash^{N}_{j,k;\ell_0,\ell_1} \varnothing$$

if there exists a $(j,k;\ell_0,\ell_1;\mathbf{A})$-proof from $(X,\Sigma)$ whose last constraint is empty, and which introduces at most $N$ new variables.

## Definition

$(\mathbf{A},d)$ has **VI-width** $(j,k;\ell_0,\ell_1)$ if $\exists$ polynomial $p(x)$ such that for every instance $(X,\Sigma)$ of $\mathrm{CSP}(\mathbf{A},d)$ with $|X|=n$,

$$(X,\Sigma) \text{ is unsatisfiable } \Leftrightarrow (\mathbf{A},X,\Sigma) \Vdash^{p(n)}_{j,k;\ell_0,\ell_1} \varnothing.$$

## Definition

$(\mathbf{A}, d)$ has **bounded VI-width** if it has VI-width $(j, k, \ell_0, \ell_1)$ for some $j, k, \ell_0, \ell_1$.

**Fact**: if $(\mathbf{A}, d)$ has bounded VI-width, then

- $\mathrm{CSP}(\mathbf{A}, d)$ is in NP $\cap$ co-NP.
- Unsatisfiable instances of $\mathrm{CSP}(\mathbf{A}, d)$ have nice "short proofs" of unsatisfiability.

## Definition

$(\mathbf{A}, d)$ has **bounded VI-width** if it has VI-width $(j, k, \ell_0, \ell_1)$ for some $j, k, \ell_0, \ell_1$.

**Fact**: if $(\mathbf{A}, d)$ has bounded VI-width, then

- $\mathrm{CSP}(\mathbf{A}, d)$ is in NP $\cap$ co-NP.
- Unsatisfiable instances of $\mathrm{CSP}(\mathbf{A}, d)$ have nice "short proofs" of unsatisfiability.

## Definition

$(\mathbf{A}, d)$ has **strongly bounded VI-width** if for some $j, k, \ell_0, \ell_1$:

- $(\mathbf{A}, d)$ has VI-width $(j, k; \ell_0, \ell_1)$, and
- there exists a polynomial-time algorithm solving $\mathrm{CSP}(\mathbf{A}, d)$ and which, for unsatisfiable instances, returns a $(j, k; \ell_0, \ell_1, \mathbf{A})$-proof of an empty constraint. (Such an algorithm is *good*.)

**Thus**: if $(\mathbf{A}, d)$ has strongly bounded VI-width then $\mathrm{CSP}(\mathbf{A}, d)$ is in P.

**Main Question**:

1. Which $(\mathbf{A}, d)$ have bounded VI-width? Strongly bounded VI-width?

**Main Question**:

1. Which $(\mathbf{A}, d)$ have bounded VI-width? Strongly bounded VI-width?

**What I know**:

- If $V(\mathbf{A})$ is congruence SD($\wedge$), then $(\mathbf{A}, d)$ has strongly bounded VI-width for all $d \geq 2$ (by Barto, Kozik).

**Main Question**:

1. Which $(\mathbf{A}, d)$ have bounded VI-width? Strongly bounded VI-width?

**What I know**:

- If $V(\mathbf{A})$ is congruence SD($\wedge$), then $(\mathbf{A}, d)$ has strongly bounded VI-width for all $d \geq 2$ (by Barto, Kozik).
- (Generalizing GE): If $\mathbf{A}$ is a finite affine space, then $(\mathbf{A}, d)$ has strongly bounded VI-width for all $d \geq 2$.

**Main Question**:

1. Which $(\mathbf{A}, d)$ have bounded VI-width? Strongly bounded VI-width?

**What I know**:

- If $V(\mathbf{A})$ is congruence SD($\wedge$), then $(\mathbf{A}, d)$ has strongly bounded VI-width for all $d \geq 2$ (by Barto, Kozik).
- (Generalizing GE): If $\mathbf{A}$ is a finite affine space, then $(\mathbf{A}, d)$ has strongly bounded VI-width for all $d \geq 2$.
- If $\mathbf{A} = (S_3, xy^{-1}z)$ then $(\mathbf{A}, 3)$ has strongly bounded VI-width.

**Main Question**:

1. Which $(\mathbf{A}, d)$ have bounded VI-width? Strongly bounded VI-width?

**What I know**:

- If $V(\mathbf{A})$ is congruence SD($\wedge$), then $(\mathbf{A}, d)$ has strongly bounded VI-width for all $d \geq 2$ (by Barto, Kozik).
- (Generalizing GE): If $\mathbf{A}$ is a finite affine space, then $(\mathbf{A}, d)$ has strongly bounded VI-width for all $d \geq 2$.
- If $\mathbf{A} = (S_3, xy^{-1}z)$ then $(\mathbf{A}, 3)$ has strongly bounded VI-width.

**More Questions**:

2. Is it true that if $\mathbf{G}$ is a finite group and $\mathbf{A} = (G, xy^{-1}z)$, then $(\mathbf{A}, d)$ has strongly bounded VI-width for all $d \geq 2$?

**Main Question**:

1. Which $(\mathbf{A}, d)$ have bounded VI-width? Strongly bounded VI-width?

**What I know**:

- If $V(\mathbf{A})$ is congruence SD($\wedge$), then $(\mathbf{A}, d)$ has strongly bounded VI-width for all $d \geq 2$ (by Barto, Kozik).
- (Generalizing GE): If $\mathbf{A}$ is a finite affine space, then $(\mathbf{A}, d)$ has strongly bounded VI-width for all $d \geq 2$.
- If $\mathbf{A} = (S_3, xy^{-1}z)$ then $(\mathbf{A}, 3)$ has strongly bounded VI-width.

**More Questions**:

2. Is it true that if $\mathbf{G}$ is a finite group and $\mathbf{A} = (G, xy^{-1}z)$, then $(\mathbf{A}, d)$ has strongly bounded VI-width for all $d \geq 2$?
3. Same question for any finite idempotent Maltsev algebra $\mathbf{A}$.

## Speculations

4. Is there a polynomial-time "strong bounded VI-width" algorithm for $\mathrm{CSP}(\mathbf{A}, d)$, when $\mathbf{A}$ is Maltsev, which is "essentially" local consistency + Gaussian elimination?

## Speculations

4. Is there a polynomial-time "strong bounded VI-width" algorithm for $\mathrm{CSP}(\mathbf{A}, d)$, when $\mathbf{A}$ is Maltsev, which is "essentially" local consistency $+$ Gaussian elimination?

5. If $\mathbf{A}$ is the naked 2-element set, then $\mathrm{CSP}(\mathbf{A}, 3) \equiv$ 3-SAT. It can be shown that every unsatisfiable instance of $\mathrm{CSP}(\mathbf{A}, 3)$ can $(3, 6, 2, 1, \mathbf{A})$-prove an empty constraint.

   (Hint: simulate resolution.)

## Speculations

4. Is there a polynomial-time "strong bounded VI-width" algorithm for $\mathrm{CSP}(\mathbf{A}, d)$, when $\mathbf{A}$ is Maltsev, which is "essentially" local consistency $+$ Gaussian elimination?

5. If $\mathbf{A}$ is the naked 2-element set, then $\mathrm{CSP}(\mathbf{A}, 3) \equiv$ 3-SAT. It can be shown that every unsatisfiable instance of $\mathrm{CSP}(\mathbf{A}, 3)$ can $(3, 6, 2, 1, \mathbf{A})$-prove an empty constraint.

   (Hint: simulate resolution.)

   Is it true that for every finite idempotent $\mathbf{A}$ there exist $j, k, \ell_0, \ell_1$ such that every unsatisfiable instance of $\mathrm{CSP}(\mathbf{A}, 3)$ has a $(j, k; \ell_0, \ell_1; \mathbf{A})$-proof of unsatisfiability? (Conjecture: NO)

## Speculations

4. Is there a polynomial-time "strong bounded VI-width" algorithm for $\mathrm{CSP}(\mathbf{A}, d)$, when $\mathbf{A}$ is Maltsev, which is "essentially" local consistency + Gaussian elimination?

5. If $\mathbf{A}$ is the naked 2-element set, then $\mathrm{CSP}(\mathbf{A}, 3) \equiv$ 3-SAT. It can be shown that every unsatisfiable instance of $\mathrm{CSP}(\mathbf{A}, 3)$ can $(3, 6, 2, 1, \mathbf{A})$-prove an empty constraint.

   (Hint: simulate resolution.)

   Is it true that for every finite idempotent $\mathbf{A}$ there exist $j, k, \ell_0, \ell_1$ such that every unsatisfiable instance of $\mathrm{CSP}(\mathbf{A}, 3)$ has a $(j, k; \ell_0, \ell_1; \mathbf{A})$-proof of unsatisfiability? (Conjecture: NO)

**Thank you!**