SAT-based Techniques for Lexicographically Smallest Finite Models

Mikoláš Janota   Choiwah Chow   João Araújo
Michael Codish   Petr Vojtěchovský

MINISTRY OF EDUCATION, YOUTH AND SPORTS

AAAI'24

Motivation: Universal Algebra

- In **universal algebra** mathematicians study **classes of mathematical structures**
- **Example:** Semigroups, groups, quasigroups
- Multiplication table a popular representation
- **Example** binary operations

OR	**0**	**1**
0	0	1
1	1	1

AND	**0**	**1**
0	0	0
1	0	1

XOR	**0**	**1**
0	0	1
1	1	0

Isomorphism

Operations $*$ and $\diamond$ are isomorphic
iff there is a bijection f, s.t.

$$f(x * y) = f(x) \diamond f(y)$$

Example

$$f(x) = 1 - x$$

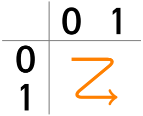
OR	**0**	**1**
0	0	1
1	1	1

AND	**0**	**1**
0	0	0
1	0	1

Comparing Tables

AND	0	1
0	0	0
1	0	1

OR	0	1
0	0	1
1	1	1



0 0 0 1 $<$ 0 1 1 1

The Task

Given:

A multiplication table $*$

Calculate:

A multiplication table $\diamond$

1. isomorphic to $*$
2. lexicographically smallest.

Example

$*$	**1**	**2**	**3**	**4**	**5**	**6**	**7**
1	7	5	6	1	4	2	3
2	5	3	1	2	6	7	4
3	6	1	5	3	7	4	2
4	1	2	3	4	5	6	7
5	4	6	7	5	2	3	1
6	2	7	4	6	3	1	5
7	3	4	2	7	1	5	6

→

$\diamond$	**1**	**2**	**3**	**4**	**5**	**6**	**7**
1	1	2	3	4	5	6	7
2	2	3	4	5	6	7	1
3	3	4	5	6	7	1	2
4	4	5	6	7	1	2	3
5	5	6	7	1	2	3	4
6	6	7	1	2	3	4	5
7	7	1	2	3	4	5	6

$\mathbb{Z}_7$

Idea for an Algorithm

$*$	**1**	**2**	**3**	**4**	**5**	**6**	**7**
1	7	5	6	1	4	2	3
2	5	3	1	2	6	7	4
3	6	1	5	3	7	4	2
4	1	2	3	4	5	6	7
5	4	6	7	5	2	3	1
6	2	7	4	6	3	1	5
7	3	4	2	7	1	5	6

$\longrightarrow$

$\diamond$	**1**	**2**	**3**	**4**	**5**	**6**	**7**
1		1	12**3**	**4**	**5**	**6**	**7**
2	**2**	**3**	**4**	**5**	**6**	**7**	**1**
3	**3**	**4**	**5**	**6**	**7**	**1**	**2**
4	**4**	**5**	**6**	**7**	**1**	**2**	**3**
5	**5**	**6**	**7**	**1**	**2**	**3**	**4**
6	**6**	**7**	**1**	**2**	**3**	**4**	**5**
7	**7**	**1**	**2**	**3**	**4**	**5**	**6**

Finding Isomorphisms with SAT

- Rewrite constraint $r \diamond c = v$ as:

$$f(f^{-1}(r) * f^{-1}(c)) = v$$

$$f(f^{-1}(r) * f^{-1}(c)) = ff^{-1}(r) \diamond ff^{-1}(c) = r \diamond c$$

- introduce variables $x_{i \to j}$ representing $f(i) = j$
- make sure $x_{i \to j}$ represent a permutation

$$\sum_{j \in D} x_{j \to i} = \sum_{j \in D} x_{i \to j} = 1, \text{ for } i \in D$$

- fix $r \diamond c = v$:

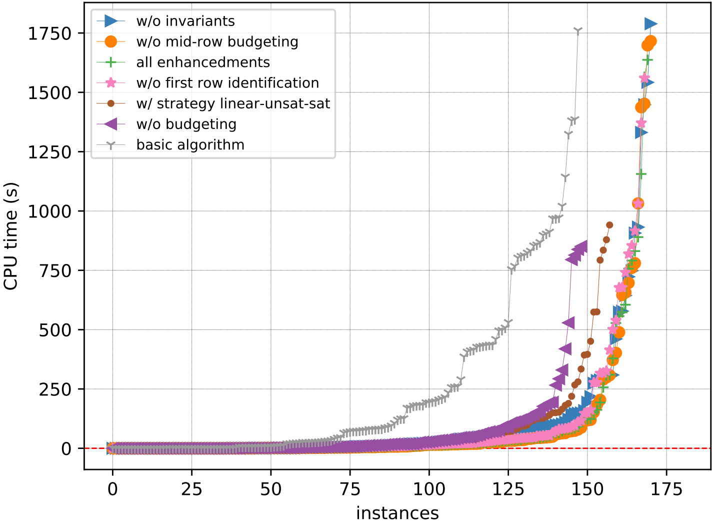
$$(x_{i \to r} \wedge x_{j \to c}) \Rightarrow x_{i * j \to v} \text{ for } i, j \in D$$

Main Improvements

- **Budgeting**
  - If every row is permutation:
    avoid unnecessary SAT calls.
  - More general:
    use max frequency of an element in a row.
- **First row identification**
  - Row r full of elements r becomes the first row
  - more generally, row r with maximal:

$$\{x \in D \mid r * x = r\}, r * r = r$$

Results



w/o invariants
w/o mid-row budgeting
all enhancedments
w/o first row identification
w/ strategy linear-unsat-sat
w/o budgeting
basic algorithm
CPU time (s)
0
250
500
750
1000
1250
1500
1750
0
25
50
75
100
125
150
175
instances

Summary and Future Work

- Canonicalize algebras by SAT solvers,
- SAT encoding via isomorphism,
- Propagation tricks to help the SAT solver.

- More propagation?
- Specific types of structures?

https://github.com/MikolasJanota/mlex