Minimal Generating Sets in Magmas

Mikoláš Janota[1] António Morgado[2] Petr Vojtěchovský[3]

[1] Czech Technical University in Prague
[2] IST/INESC-ID, University of Lisbon, Portugal,
[3] University of Denver

AITP 2021

Background

In algebra we have

- groups
- semi-groups
- quasi-groups
- loops
- ...

Background

In algebra we have

- groups
- semi-groups
- quasi-groups
- loops
- ...

These are all magmas.

Generating Set (Generators)

- A set S is generating if all the other elements can be obtained by a finite number of multiplications
- **Example:** $\mathbb{N}$ is generated by $\{0, 1\}$ under $+$.

Smallest Generating Sets

$*$	1	2	3
1	2	3	1
2	3	1	2
3	1	2	3

- Is $\{2, 3\}$ generating?

Smallest Generating Sets

$*$	1	2	3
1	2	3	1
2	3	1	2
3	1	2	3

- Is $\{2, 3\}$ generating?... YES
- Is it the smallest possible? NO

Smallest Generating Sets

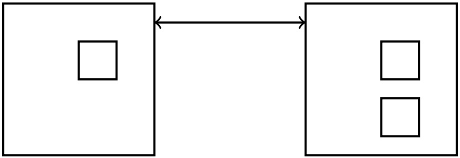
$*$	1	2	3
1	2	3	1
2	3	1	2
3	1	2	3

- Is $\{2, 3\}$ generating?... YES
- Is it the smallest possible? NO
- $\{1\}$ is already generating

$$\begin{aligned} 1 &= 1 \\ 1 * 1 &= 2 \\ 1 * 1 * 1 &= 3 \end{aligned}$$

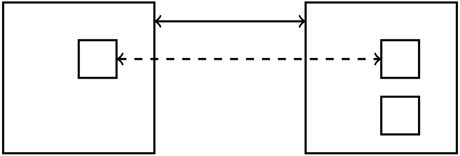
Why Interesting?

Example: Determine if to structures are isomorphic



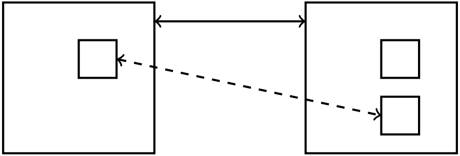
Why Interesting?

Example: Determine if to structures are isomorphic



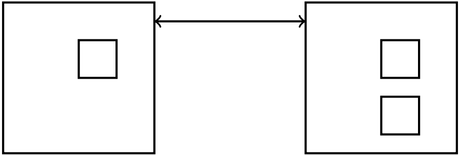
Why Interesting?

Example: Determine if to structures are isomorphic



Why Interesting?

Example: Determine if to structures are isomorphic



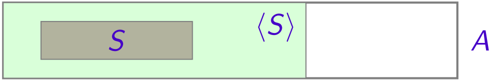
Many other algorithms in computational algebra depend on the generating set and its size.

Calculating Minimal Generating set (finite case)

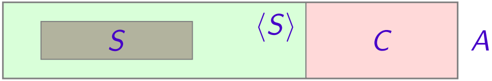
- The problem is in NP
- SAT solvers scale poorly on the problem
- Can we use SAT without overloading it?

Idea

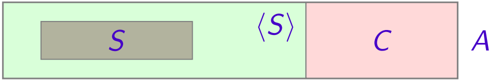
Idea



Idea

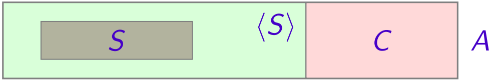


Idea



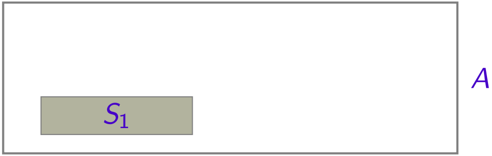
1. If $C = \emptyset$, we are DONE.

Idea

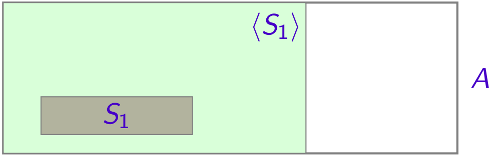


1. If $C = \emptyset$, we are DONE.
2. If $C \neq \emptyset$, any generating S' must intersect with C.

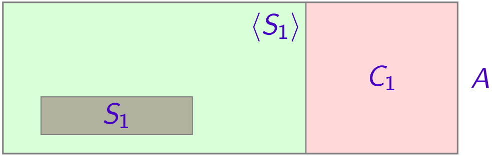
From Idea to Algorithm



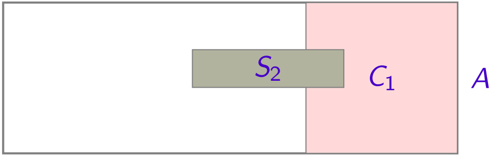
From Idea to Algorithm



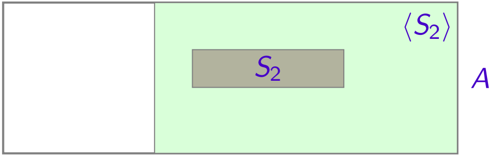
From Idea to Algorithm



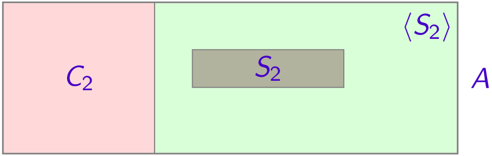
From Idea to Algorithm



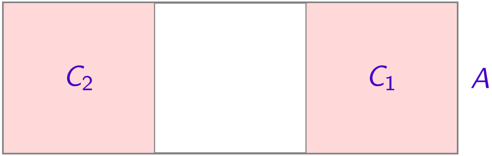
From Idea to Algorithm



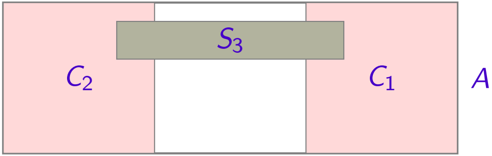
From Idea to Algorithm



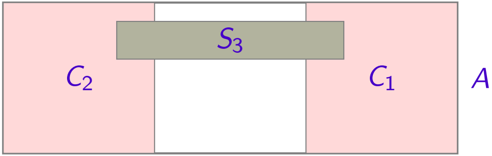
From Idea to Algorithm



From Idea to Algorithm



From Idea to Algorithm



To guarantee the smallest, always calculate the smallest candidate.

A Few Words about the Implementation and Results

1. The problem in each iteration is in fact:
   Minimum Hitting Set

A Few Words about the Implementation and Results

1. The problem in each iteration is in fact:
   Minimum Hitting Set
2. SAT performs poorly on those

A Few Words about the Implementation and Results

1. The problem in each iteration is in fact:
   Minimum Hitting Set
2. SAT performs poorly on those
3. Integer Linear Programming solvers (gurobi) extremely well.

A Few Words about the Implementation and Results

1. The problem in each iteration is in fact:
   Minimum Hitting Set
2. SAT performs poorly on those
3. Integer Linear Programming solvers (gurobi) extremely well.
4. Thousands of elements

A Few Words about the Implementation and Results

1. The problem in each iteration is in fact: Minimum Hitting Set
2. SAT performs poorly on those
3. Integer Linear Programming solvers (gurobi) extremely well.
4. Thousands of elements
5. We have seen minimal generating sets up to 7

Musing

- Why does it work so well?
  Example: In 2000 elements, complements at least thousand elements, only hundreds are needed to force size 6 generating set.

Musing

- Why does it work so well?
  Example: In 2000 elements, complements at least thousand elements, only hundreds are needed to force size 6 generating set.
- In general: when does it pay off to directly target the original problem and when to do the gradual refinement?

Musing

- Why does it work so well?
  Example: In 2000 elements, complements at least thousand elements, only hundreds are needed to force size 6 generating set.
- In general: when does it pay off to directly target the original problem and when to do the gradual refinement?
- Understanding the results

Musing

- Why does it work so well?
  Example: In 2000 elements, complements at least thousand elements, only hundreds are needed to force size 6 generating set.
- In general: when does it pay off to directly target the original problem and when to do the gradual refinement?
- Understanding the results
  - If A_1 generated by size m, if A_2 generated by size n, $A_1 \times A_2$ generated by size $m + n$.

Musing

- Why does it work so well?
  Example: In 2000 elements, complements at least thousand elements, only hundreds are needed to force size 6 generating set.
- In general: when does it pay off to directly target the original problem and when to do the gradual refinement?
- Understanding the results
  - If A_1 generated by size m, if A_2 generated by size n, $A_1 \times A_2$ generated by size $m + n$.
  - BUT sometimes less.

Musing

- Why does it work so well?
  Example: In 2000 elements, complements at least thousand elements, only hundreds are needed to force size 6 generating set.
- In general: when does it pay off to directly target the original problem and when to do the gradual refinement?
- Understanding the results
  - If A_1 generated by size m, if A_2 generated by size n, $A_1 \times A_2$ generated by size $m + n$.
  - BUT sometimes less.
  - When?