Enumerating Grid Classes Using MSOL

Radek Hušek (Czech Technical University)

Joint work with Michal Opler

Graphs
Games
Optimization
Algorithms
Theoretical
Computer Science

Permutation Patterns 2025 July 8, 2025

Automaton connection

1. Geometric grid classes can be represented[1] by WS1S formulas (Braunfeld 2024).
   $\Rightarrow$ Python script to generate the formula.
2. WS1S formulas can be represented by (deterministic) finite automatons.
   $\Rightarrow$ MONA tool.
3. Obtain the generating function counting strings of length k accepted by a DFA.
   $\Rightarrow$ Solve system of linear equations (we use SageMath).

[1] We need representations to be 1:1 and size preserving.

How does the formula look?

Used version of MSO

- Objects: integers $1, \ldots, n$.
- 1st order variables.
- 2nd order variables: only unary relations (aka sets).
- We have $x \in X$ and $x < y$ but not $x + y$.

We want

Two linear orders: `left_of`(x, y) and `below_of`(x, y).

How does the formula look?

We want

Two linear orders: `left_of`(x, y) and `below_of`(x, y).

- MSO cannot encode arbitrary linear order directly (in a size preserving way) – there is too many of them.
- We must choose some small subclass of permutations…

1st option: Insertion encoding (simplified)

- We fix k – the number of insertion points.
- Inserting at a point inserts before it.
- We insert values from 1 to n.
- Encoding: partition P_i for $i \in \{1, \ldots, k\}$.
  E.g., for permutation 24513 we get $P_1 = \{2, 4, 5\}$, $P_2 = \{1, 3\}$.
- The linear orders:

$$\texttt{below\_of}(x, y) := x < y$$

$$\texttt{left\_of}(x, y) := \begin{cases} x < y & \text{if } \exists i : x \in P_i \wedge y \in P_i \\ i < j & \text{if } x \in P_i \wedge y \in P_j \text{ and } i \neq j \end{cases}$$

- Size preserving but not 1:1.
- Taking minimal representation helps.

2nd option: Grid encoding

- Fix a matrix $M \in \{-1, 0, 1\}^{r \times c}$:
  - -1: decreasing,
  - 0: empty,
  - 1: increasing.
- We number nonzero entries $1, \ldots, k$.
- Encoding: partition A_i for $i \in \{1, \ldots, k\}$.
- Signs for rows and columns need to be selected so $r_i c_j = M_{i,j}$ for all $M_{i,j} \neq 0$.
- `below_of` and `left_of` similar to `left_of` of insertion encoding but account for signs.
- Again size preserving but not 1:1.
- Idea of the fix is the same but much more complicated.
- Worked out by Braunfeld in 2024.

What about cyclic classes?

- Braunfeld's paper works for geometric grid classes.
- We implement only acyclic grid classes.
- Why?

- Existence of cycles requires extra checks – the formula is bigger.
- Even without them, computation explodes above 5 non-zero entries (partly due to limitations of MONA).
- So it's planned but low priority…

Results

Class	M	Generating function
L_1	$\begin{pmatrix} -1 & -1 \\ 1 & 0 \end{pmatrix}$	$-\frac{2x^4 - 6x^3 + 4x^2 - x}{2x^4 - 9x^3 + 12x^2 - 6x + 1}$
L_4	$\begin{pmatrix} -1 & -1 \\ 1 & 0 \end{pmatrix}$	$-\frac{x^4 - 4x^3 + 3x^2 - x}{(1-x)^2(1-3x+x^2)}$
T_5	$\begin{pmatrix} -1 & 1 & -1 \\ 0 & 1 & 0 \end{pmatrix}$	$\frac{(14x^5 - 56x^4 + 71x^3 - 39x^2 + 10x - 1)x}{(2x^2 - 4x + 1)(x^2 - 3x + 1)(3x - 1)(x - 1)^2}$
J_3	$\begin{pmatrix} -1 & -1 & 1 \\ 1 & 0 & 0 \end{pmatrix}$	$-\frac{(3x^4 - 13x^3 + 17x^2 - 7x + 1)x}{(2x^2 - 4x + 1)(x^2 - 3x + 1)(2x - 1)}$
J_{10}	$\begin{pmatrix} 1 & -1 & 1 \\ -1 & 0 & 0 \end{pmatrix}$	$-\frac{(3x^5 - 4x^4 - 15x^3 + 18x^2 - 7x + 1)x}{(x^2 - 3x + 1)(3x - 1)(2x - 1)(x - 1)}$
W_{10}	$\begin{pmatrix} -1 & 0 & 0 \\ 1 & -1 & 0 \\ 0 & 1 & -1 \end{pmatrix}$	$-\frac{(x^5 - 7x^4 + 19x^3 - 18x^2 + 7x - 1)x}{(x^3 - 6x^2 + 5x - 1)(x^2 - 3x + 1)(x - 1)}$

Example

Source code

```
https://github.com/PitelVonSacek/PerMSO
```

Input file (`ex.yaml`)

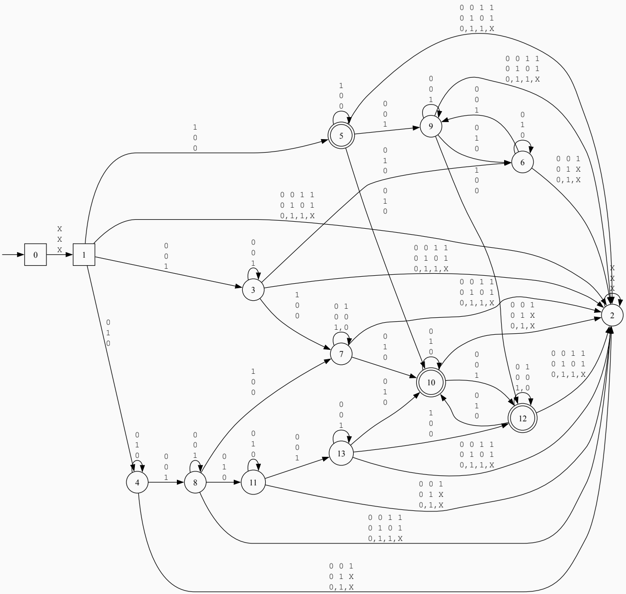
```
class: [[ 1, 1 ],
        [ 0, 1 ]]
```

Shell

```
MONA=ex.mona AUTOMATON=ex.auto EXPAND=30 \
    ./process.sh < ex.yaml
```

- Output generating function and its first 30 values.
- Saves MSOL formula to `ex.mona` and the resulting automaton to `ex.auto`.
- The resulting automaton has $13 + 1$ states.

Example – DFA



X X X
1 0 0
0 0 1
0 1 0
0 0 1 1 0 1 0 1 0,1,1,X
0 0 1 0 1 X 0,1,X
0 1 0
0 0 1
1 0 0
0 1 0
0 1 0 0 1,0
0 1 0 0 1,0
X X X
0
1
2
3
4
5
6
7
8
9
10
11
12
13

Future work

- Possibly better encoding of grid classes.
- MONA improvements:
  - Increase limit on the number of nodes (currently 16M).
  - Experiment with optimizations – notably with the order of variables.
  - Full rewrite easier than hacking MONA code.
  - In progress in very early stage.
- Generic basis calculation (so it supports extra conditions).
- Cyclic geometric grid classes.