Using Constraint Programming to Enumerate Permutations avoiding Mesh Patterns

Ruth Hoffmann, Özgür Akgün, Christopher Jefferson

Permutation Patterns 2022, Valparaiso

Why?

- At Permutation Patterns 2021 I heard that the enumeration research has stalled a wee bit.
- Constraint programming is great (and modular)!
- Sergey has had a few successes with CP and enumerating word problems.
- (Really I just want to return to the Permutation Pattern Community)
- (There are grants being submitted/reviewed that mention Permutation Patterns and CP)

Constraint Satisfaction Problem (CSP)

Constraint Satisfaction Problem is a neat way of representing a problem and the solving of it is a very fast combinatorial search. It takes

- some variables,
- some values that the variables can have,
- some rules about these variables and values,

and then mix and matches the values to the variables and checks if any of the rules are being broken.

More formally

A CSP consists of the *given*

- V – set of *decision variables*.
- For each $v \in V$ there is a *domain* D_v, which consists of potential values of v.
- C – set of *constraints* on the decision variables.

and we want to *find* an assignment of values to variables such that all constraints are satisfied The instance consists of some specific assignments which apply to a particular case of the problem.

Decision Variables

A decision variable corresponds to a choice that must be made in solving a problem.

Domains

- Values in the domain of a decision variable correspond to the options for a particular choice.
- A decision variable is assigned a single value from its domain.

Constraints

- *scope*: subset of the decision variables a constraint involves.
- Of the possible combinations of assignments to the variables in its scope, a constraint specifies:
  - Which are allowed. (Assignments that satisfy the constraint.)
    OR
  - Which are disallowed. (Assignments that violate the constraint.)
- Constraints can be specified using logical and arithmetic operators (and others).

Solving Problems with Constraints

Consists of 2 phases

1. Describe the problem to be solved as a constraint model. (In a format suitable for input to a constraint solver)
2. Search (automatically) for solutions to the model with a constraint solver.

Why should you care?

- There are a few tools out there already (Permuta, PermLab, PatternClass,...)
- This might not be faster than those (we haven't tested it yet)
- Is it giving us new insights for $Av(1324)$ or any other classes? (Not yet)

So really why?

- It's modular, and we are making it more friendly for mathematicians.
- It's *just* expressing our/your definitions in a nice way as constraints and it does all the fast computation work for us.

Demo

Let’s have a look at CP in action.

If the demo went well

This is early work, but here are our plans.

- We are looking for more properties, more problems, more anything to model.
- We are going to compare it to the existing tools out there.
- We are working on making it nicer for mathematicians to work in.
- We are working on making it faster through symmetries, 'homomorphisms', etc.

If the demo did not go well

This is very early work, the demo was run in a student created online collaborative worksheet and we are working on making it more stable. But you can create models and run things locally, which works! Here are our plans.

- We are looking for more properties, more problems, more anything to model.
- We are working on making it nicer for mathematicians to work in. (grant in review)
- We are working on making it faster through symmetries, 'homomorphisms', etc. (grant about to be submitted)

Finally, finally

Thanks to my collaborators and modelling wizards Özgür Akgün and Chris Jefferson.

These slides, more code, and the notebook are online at
`https://github.com/stacs-cp/permutation-classes-cp`

University of St Andrews

Thank you!

ruthhoffmann

@ruthhoffmann

rh347@st-andrews.ac.uk