



Topic 5: Symmetry

(Version of 3rd August 2023)

Pierre Flener

Optimisation Group

Department of Information Technology
Uppsala University
Sweden

Course 1DL442:
Combinatorial Optimisation and Constraint Programming,
whose part 1 is Course 1DL451:
Modelling for Combinatorial Optimisation



Outline

Introduction

Symmetry
Breaking by
Reformulation

Symmetry
Breaking by
Constraints

Conclusion

- 1. Introduction**
- 2. Symmetry Breaking by Reformulation**
- 3. Symmetry Breaking by Constraints**
- 4. Conclusion**



Outline

Introduction

Symmetry
Breaking by
Reformulation

Symmetry
Breaking by
Constraints

Conclusion

- 1. Introduction**
2. Symmetry Breaking by Reformulation
3. Symmetry Breaking by Constraints
4. Conclusion



Symmetry in Nature

Introduction

Symmetry
Breaking by
Reformu-
lation

Symmetry
Breaking by
Constraints

Conclusion



Johannes Kepler, *On the Six-Cornered Snowflake*, 1611: six-fold rotation symmetry of snowflakes, role of symmetry in human perception and the arts, fundamental importance of symmetry in the laws of physics.



Broken Symmetry in Nature

Introduction

Symmetry
Breaking by
Reformu-
lation

Symmetry
Breaking by
Constraints

Conclusion



The **Angora cat** originated in the Turkish city of Ankara. It is admired for its long silky coat and quiet graceful charm. (*Turkish Daily News*, 14 October 2001)



Introduction

Symmetry
Breaking by
Reformulation

Symmetry
Breaking by
Constraints

Conclusion



The Nobel Prize in Physics 2008

"for the discovery of the mechanism of spontaneous broken symmetry in subatomic physics"

"for the discovery of the origin of the broken symmetry which predicts the existence of at least three families of quarks in nature"



Photo: University of Chicago

Yoichiro Nambu



© The Nobel Foundation
Photo: U. Montan

Makoto Kobayashi



© The Nobel Foundation
Photo: U. Montan

Toshihide Maskawa



Value Symmetry

Example (Map colouring)

Use k colours to paint the countries of a map such that no neighbour countries have the same colour.

The model where the countries (as decision variables) take colours (as values) has $k!$ **value symmetries** because any permutation of the colours transforms a (non-)solution into another (non-)solution: the values are not distinguished. (Continued on slide 36)

Example (Partitioned map colouring)

The colours of map colouring are partitioned into subsets, such that only the colours of the same subset are not distinguished. (Continued on slide 36)



Variable Symmetry

Example (n -Queens)

The model with a 2d array of decision variables in $0..1$ has 4 **reflection symmetries** and 4 **rotation symmetries**, which are **variable symmetries**, as any reflection or rotation (or both) of an $n \times n$ board with n queens transforms that (non-)solution into another (non-)solution. (Continued on slide 35)

Example (Subset)

Find an n -element subset of a given set S , subject to some constraints.

The model encoding the subset as an array of n decision variables of domain S , constrained to take distinct values, has $n!$ **variable symmetries** as the order of the elements does not matter in a set, but does matter in an array. (Continued on slide 34)



Symmetries can be introduced!

- The symmetries in the (partitioned) map colouring and n -queens models are **problem symmetries**: they are detectable in *every* model.
- The symmetries in the subset model are *not* problem symmetries but **model symmetries**: they are *not* detectable in every model.
- There can also be **instance symmetries**, which are detectable in the instance data of a problem.
Example: cargo boats with the same capacity in the same harbour.

Observation:

A solver may waste a lot of effort on gazillions of (partial) **non-solutions** that are symmetric to already visited ones, whereas a found **solution** can be transformed without search into a symmetric solution in polynomial time.



Definition (also see Cohen et al. @ *Constraints*, 2006)

A **symmetry** is a permutation of values or decision variables (or both) that **preserves solutions**: it transforms (partial) solutions into (partial) solutions, and it transforms (partial) non-solutions into (partial) non-solutions.

Symmetries form a **group**:

- The inverse of a symmetry is a symmetry.
- The identity permutation is a symmetry.
- The composition of two symmetries is a symmetry.

(Computational) **group theory** is the way to study symmetry.



Example (Agricultural experiment design, AED)

	plot1	plot2	plot3	plot4	plot5	plot6	plot7
barley	1	1	1	0	0	0	0
corn	1	0	0	1	1	0	0
millet	1	0	0	0	0	1	1
oats	0	1	0	1	0	1	0
rye	0	1	0	0	1	0	1
spelt	0	0	1	1	0	0	1
wheat	0	0	1	0	1	1	0

Constraints to be satisfied:

- 1 Equal growth load: Every plot grows 3 grains.
- 2 Equal sample size: Every grain is grown in 3 plots.
- 3 Balance: Every grain pair is grown in 1 common plot.

Instance: 7 plots, 7 grains, 3 grains/plot, 3 plots/grain, balance 1.

General term: **balanced incomplete block design (BIBD)**.



Example (AED and BIBD: the symmetries)

	plot1	plot2	plot3	plot4	plot5	plot6	plot7
barley	1	1	1	0	0	0	0
corn	1	0	0	1	1	0	0
millet	1	0	0	0	0	1	1
oats	0	1	0	1	0	1	0
rye	0	1	0	0	1	0	1
spelt	0	0	1	1	0	0	1
wheat	0	0	1	0	1	1	0

Observation: The grains and plots of an agricultural experiment design are not distinguished:

- The grains / rows can be permuted: $7!$ variable symmetries
- The plots / columns can be permuted: $7!$ variable symmetries

All these permutations preserve solutions.

(Continued on slide 28)




Example (The sport scheduling problem, SSP)

Find a schedule in $\text{Periods} \times \text{Weeks} \rightarrow \text{Teams} \times \text{Teams}$ for

- $|\text{Teams}| = n$ and n is even (note that only $n=4$ is unsatisfiable)
- $|\text{Weeks}| = n-1$
- $|\text{Periods}| = n/2$ periods per week

subject to the following constraints:

- 1 Each possible game is played exactly once.
- 2 Each team plays exactly once per week.
- 3 Each team plays at most twice per period.

Idea for a first model , and a solution for $n=8$ as an array Game:

	Wk 1	Wk 2	Wk 3	Wk 4	Wk 5	Wk 6	Wk 7
P 1	1 vs 2	1 vs 3	2 vs 6	3 vs 5	4 vs 7	4 vs 8	5 vs 8
P 2	3 vs 4	2 vs 8	1 vs 7	6 vs 7	6 vs 8	2 vs 5	1 vs 4
P 3	5 vs 6	4 vs 6	3 vs 8	1 vs 8	1 vs 5	3 vs 7	2 vs 7
P 4	7 vs 8	5 vs 7	4 vs 5	2 vs 4	2 vs 3	1 vs 6	3 vs 6



Example (SSP: the symmetries)

	Wk 1	Wk 2	Wk 3	Wk 4	Wk 5	Wk 6	Wk 7
P 1	1 vs 2	1 vs 3	2 vs 6	3 vs 5	4 vs 7	4 vs 8	5 vs 8
P 2	3 vs 4	2 vs 8	1 vs 7	6 vs 7	6 vs 8	2 vs 5	1 vs 4
P 3	5 vs 6	4 vs 6	3 vs 8	1 vs 8	1 vs 5	3 vs 7	2 vs 7
P 4	7 vs 8	5 vs 7	4 vs 5	2 vs 4	2 vs 3	1 vs 6	3 vs 6

Observation: The periods, weeks, game slots, and teams of a sport schedule are not distinguished:

- The periods / rows can be permuted: $4!$ variable symmetries
- The weeks / columns can be permuted: $7!$ variable symmetries
- The game slots can be permuted: $2!^{28}$ variable symmetries
- The team names can be permuted: $8!$ value symmetries

All these permutations preserve solutions. (Continued on slides 23 and 29)





Example (The social golfer problem, SGP)

Find schedule Weeks \times Groups \times Slots \rightarrow Players for

- $|\text{Weeks}| = w$
- $|\text{Groups}| = g$ groups per week
- $|\text{Slots}| = s$ players per group
- $|\text{Players}| = g \cdot s$

subject to the following constraint:

- 1 Any two players are at most once in the same group.

Idea for a first model , and a solution for $\langle w, g, s \rangle = \langle 4, 4, 3 \rangle$ :

	Group 1	Group 2	Group 3	Group 4
Week 1	[1, 2, 3]	[4, 5, 6]	[7, 8, 9]	[10, 11, 12]
Week 2	[1, 4, 7]	[2, 5, 10]	[3, 8, 11]	[6, 9, 12]
Week 3	[1, 8, 10]	[2, 4, 12]	[3, 5, 9]	[6, 7, 11]
Week 4	[1, 9, 11]	[2, 6, 8]	[3, 4, 10]	[5, 7, 12]

By the way, there is no solution when adding a fifth week!



Example (SGP: the symmetries)

	Group 1	Group 2	Group 3	Group 4
Week 1	[1, 2, 3]	[4, 5, 6]	[7, 8, 9]	[10, 11, 12]
Week 2	[1, 4, 7]	[2, 5, 10]	[3, 8, 11]	[6, 9, 12]
Week 3	[1, 8, 10]	[2, 4, 12]	[3, 5, 9]	[6, 7, 11]
Week 4	[1, 9, 11]	[2, 6, 8]	[3, 4, 10]	[5, 7, 12]

Observation: The weeks, groups, group slots, and players of a social golfer schedule are not distinguished:

- The weeks / rows can be permuted: $4!$ variable symmetries
- The groups can be permuted **within a week**: $4!^4$ variable symmetries
- The group slots can be permuted: $3!^{16}$ variable symmetries
- The player names can be permuted: $12!$ value symmetries

All these permutations preserve solutions. (Continued on slide 24)



Terminology, for Variable and Value Symmetries

Definitions (Special cases of symmetry)

- **Full symmetry**: any permutation preserves solutions.
The full symmetry group S_n has $n!$ symmetries over a list of n elements.
- **Partial symmetry**: any piecewise permutation preserves solutions.
This often occurs in instances.
Examples: weekdays vs weekend; same-capacity boats.
- **Wreath symmetry**: any wreath permutation preserves solutions.
Example: the composition of the first two variable symmetries of SGP.
- **Rotation symmetry**: any rotation preserves solutions.
The cyclic symmetry group C_n has n symmetries over a circular list of n elements.



Definitions (Special cases of symmetry, end)

- **Index symmetry**: any permutation of slices of an array of decision variables preserves solutions: full vs partial **row symmetry**, full vs partial **column symmetry**, ...
- **Conditional** or **dynamic symmetry**: a symmetry that appears while solving a problem. Such symmetries are beyond the scope of this topic.

Careful: Index symmetries multiply up!

If there is full row and column symmetry in an $m \times n$ array (that is, if there are $m!$ row symmetries and $n!$ column symmetries), then there are ~~$m! + n!$~~ $m! \cdot n!$ compositions of symmetries, and **at most** $m! \cdot n! - 1$ symmetric solutions per solution.

For example, none of the $2^{1 \cdot 4} = 16$ Boolean 1×4 arrays can have $1! \cdot 4! - 1 = 23$ distinct symmetric arrays.



Challenges Raised by Symmetries

Introduction

Symmetry
Breaking by
Reformulation

Symmetry
Breaking by
Constraints

Conclusion

Definition

Symmetry handling has two aspects:

- **Detecting** both ~~the~~ symmetries of the problem (in a model) and ~~the~~ symmetries introduced when modelling.
- **Breaking** (better: **exploiting**) ~~the~~ detected symmetries so that less effort is spent on the solving: multiple symmetric representations of a solution are avoided.

Automated detection is beyond the scope of this course.



Classification of Symmetry Breaking

Definition

A **symmetry class** is an equivalence class of solutions under all the considered symmetries, including their compositions.

Aim: While solving, keep ideally **one** member per symmetry class, as this may make a problem “less intractable”:

- **Symmetry breaking by reformulation:**
the elimination of ~~the~~ symmetries detectable in the model.
- **Static symmetry breaking:**
the elimination of symmetric solutions by **constraints**.
- **Dynamic symmetry breaking:**
the elimination of symmetric solutions by **search**.
This is beyond the scope of this topic: 🖱️ see Topic 15: Search (in Part 2).



Definition

Structural symmetry breaking exploits the combinatorial structure of a problem by using the key strengths of constraint-based modelling — namely the use of **constraint predicates** and **search strategies** — towards eliminating, ideally in low polynomial time and space, all symmetric solutions, even if there are exponentially many symmetries.

Careful: Size does not matter!

A number of symmetries is **no** indicator of the difficulty of breaking them!
For example, consider variable symmetry:

- The full group S_n has $n!$ easily broken symmetries: see slide 34.
- The cyclic group C_n has only n symmetries, which are harder to break.



Outline

Introduction

Symmetry
Breaking by
Reformulation

Symmetry
Breaking by
Constraints

Conclusion

1. Introduction
- 2. Symmetry Breaking by Reformulation**
3. Symmetry Breaking by Constraints
4. Conclusion



Example (The sport scheduling problem, SSP, over n teams)

Let the **domain** of the decision variables of an $\frac{n}{2} \times n$ array called Game be $\{f \cdot n + s \mid 1 \leq f < s \leq n\}$: this breaks the game-slot symmetries as the game between teams f and s is now **uniquely** identified by $f \cdot n + s$.

A **round-robin schedule** breaks many of the other symmetries:

- Restrict the games of the first week to the set $\{1 \text{ vs } 2\} \cup \{t + 1 \text{ vs } n + 2 - t \mid 1 < t \leq n/2\}$
- For the other weeks, transform each game f vs s into f' vs s' , where:

$$f' = \begin{cases} 1 & \text{if } f = 1 \\ 2 & \text{if } f = n \\ f + 1 & \text{otherwise} \end{cases}, \text{ and } s' = \begin{cases} 2 & \text{if } s = n \\ s + 1 & \text{otherwise} \end{cases}$$

We must only find the period of each game, but **not** its week: second model .



Example (The social golfer problem, SGP)

Break the slot symmetries (slide 16) within each group by switching from a 3d $w \times g \times s$ array of **integer** decision variables:

	Group 1	Group 2	Group 3	Group 4
Week 1	[1, 2, 3]	[4, 5, 6]	[7, 8, 9]	[10, 11, 12]
Week 2	[1, 4, 7]	[2, 5, 10]	[3, 8, 11]	[6, 9, 12]
Week 3	[1, 8, 10]	[2, 4, 12]	[3, 5, 9]	[6, 7, 11]
Week 4	[1, 9, 11]	[2, 6, 8]	[3, 4, 10]	[5, 7, 12]

to a 2d $w \times g$ array of **set** decision variables (see Topic 2: Basic Modelling):

	Group 1	Group 2	Group 3	Group 4
Week 1	{1, 2, 3}	{4, 5, 6}	{7, 8, 9}	{10, 11, 12}
Week 2	{1, 4, 7}	{2, 5, 10}	{3, 8, 11}	{6, 9, 12}
Week 3	{1, 8, 10}	{2, 4, 12}	{3, 5, 9}	{6, 7, 11}
Week 4	{1, 9, 11}	{2, 6, 8}	{3, 4, 10}	{5, 7, 12}

and adding the constraint that all sets be of cardinality s : second model .



Outline

Introduction

Symmetry
Breaking by
Reformulation

Symmetry
Breaking by
Constraints

Conclusion

1. Introduction
2. Symmetry Breaking by Reformulation
- 3. Symmetry Breaking by Constraints**
4. Conclusion



Useful Predicates: Lexicographic Ordering

Example

We have `lex_lesseq`([1, 2, 34, 5, 678], [1, 2, 36, 45, 78]), because $34 < 36$, even though $678 \not\leq 78$: this \leq_{lex} order on 1d integer arrays is induced by the \leq order on integers and is **not** the point-wise ordering.

Definition

The `lex_lesseq`(X, Y) constraint, where X and Y are same-length 1d arrays of decision variables, say both with indices in $1..n$, holds if and only if X is lexicographically at most equal to Y :

- either $n = 0$,
- or $X[1] < Y[1]$,
- or $X[1] = Y[1] \ \& \ \text{lex_lesseq}(X[2..n], Y[2..n])$.

Variant predicates exist.



Symmetry Breaking by Constraints

Classification:

- **Lex-leader scheme** (Crawford et al. @ *KR 1996*) (slide 31):
state one lexicographic constraint per variable symmetry.
This is general, but takes exponential space if there are exponentially many symmetries, as is often the case.
- **Structural symmetry breaking by constraints** (slide 36):
exploit the combinatorial structure of a problem towards stating fewer symmetry-breaking constraints, and not necessarily lexicographic ones.
This has already been worked out for some common combinations of variable symmetries, value symmetries, or both.

Careful: Symmetry-breaking constraints should harmonise with the choices of dummy values (see Topic 4: Modelling, and slide 36) and **search-strategy** annotations (see Topic 8: Inference & Search in CP & LCG).



Lexicographic ordering constraints along **one** dimension of an array break the index symmetry of that dimension.

Example (Balanced incomplete block design, BIBD +)

The following constraints break all the row and column symmetries (see slide 12), but **not** all their compositions:

- Each row is `lex_greater` than the next, if any.
Rows cannot be equal because of the (so far unstated) incompleteness pre-condition $2 \leq \text{blockSize} < |\text{Varieties}|$ on the parameters.
- Each column is `lex_greatereq` than the next, if any.
Columns can be equal when `balance` ≥ 2 .

The use of `lex_greatereq` (as opposed to `lex_lesseq`) will be justified in Topic 8: Inference & Search in CP & LCG.



Lexicographic ordering constraints along **one** dimension of an array break the index symmetry of that dimension.

Example (The sport scheduling problem, SSP)

The following constraints simplify the row and column lexicographic constraints on the $\frac{n}{2} \times (n - 1)$ array *Game*:

- each game in the first column is less than the next, if any,
- each game in the first row is less than the next, if any,

as the values of array *Game* are necessarily distinct.

With the following constraint on a redundant $\frac{n}{2} \times (n - 1) \times 2$ array *Team*:

- the first team of each game has a smaller number than the second team of the game (this constraint can also be enforced by the definition on slide 23 of the domain of the *Game* $[p, w]$ decision variables),

and channelling, this breaks **all** the variable symmetries (of slide 14, including their compositions) **here**, as the values of array *Game* are necessarily distinct.



Lexicographic ordering constraints along **every** dimension with index symmetry of an array have two properties:

- + No symmetry class is lost.
- In general, **not** all symmetry classes are of size 1, except if the values of the array are distinct, etc.

Counterexample

Assume full row symmetry and full column symmetry:

the arrays

0	0	1
1	1	0

 and

0	1	1
1	0	0

 have lexicographically ordered rows **and** columns, but are symmetric: each can be transformed into the other by swapping their two rows as well as swapping their first and last columns.

The lex-leader scheme (see next slide) generates lexicographic ordering constraints that are not necessarily along the dimensions of an array of decision variables. It guarantees however that all the compositions of all variable symmetries are broken.



The Lex-Leader Scheme

For **any** group G of **variable** symmetries on the indices of the decision variables x_1, \dots, x_n of domain D , which are not necessarily arranged into a 1d array:

- 1 Choose an ordering of the decision variables, say x_1, \dots, x_n .
- 2 Choose a total value ordering \preceq on D .
- 3 Choose a lexicographic-ordering predicate induced by \preceq , say `lex_lesseq` (which is induced by \leq).
- 4 For every symmetry $\sigma \in G$, add the constraint

$$\text{lex_lesseq}([x_1, \dots, x_n], [x_{\sigma(1)}, \dots, x_{\sigma(n)}])$$

to the problem model.

- 5 Simplify the resulting constraints, locally and globally.

This yields **exactly** one solution per symmetry class.



Example (2×3 array with full row and column symmetry)

Consider the array

x1	x2	x3
x4	x5	x6

 with full row and column symmetry:

$2! \cdot 3! - 1$ constraints for the ordering $x_1, x_2, x_3, x_4, x_5, x_6$ of the decision variables:

```
1 constraint lex_lesseq([x1, x2, x3, x4, x5, x6], [x2, x1, x3, x5, x4, x6]);
2 constraint lex_lesseq([x1, x2, x3, x4, x5, x6], [x1, x3, x2, x4, x6, x5]);
3 constraint lex_lesseq([x1, x2, x3, x4, x5, x6], [x4, x5, x6, x1, x2, x3]);
4 constraint lex_lesseq([x1, x2, x3, x4, x5, x6], [x6, x4, x5, x3, x1, x2]);
5 constraint lex_lesseq([x1, x2, x3, x4, x5, x6], [x5, x6, x4, x2, x3, x1]);
6 constraint lex_lesseq([x1, x2, x3, x4, x5, x6], [x4, x6, x5, x1, x3, x2]);
7 constraint lex_lesseq([x1, x2, x3, x4, x5, x6], [x5, x4, x6, x2, x1, x3]);
8 constraint lex_lesseq([x1, x2, x3, x4, x5, x6], [x6, x5, x4, x3, x2, x1]);
9 constraint lex_lesseq([x1, x2, x3, x4, x5, x6], [x3, x2, x1, x6, x5, x4]);
10 constraint lex_lesseq([x1, x2, x3, x4, x5, x6], [x2, x3, x1, x5, x6, x4]);
11 constraint lex_lesseq([x1, x2, x3, x4, x5, x6], [x3, x1, x2, x6, x4, x5]);
```

Introduction

Symmetry
Breaking by
ReformulationSymmetry
Breaking by
Constraints

Conclusion



Example (2×3 array with full row and column symmetry)

Consider the array

x1	x2	x3
x4	x5	x6

 with full row and column symmetry:

Simplified constraints for the ordering $x_1, x_2, x_3, x_4, x_5, x_6$ of the decision variables:

```

1 constraint lex_lesseq([x1      , x4      ], [x2      , x5      ]);
2 constraint lex_lesseq([  x2      , x5      ], [  x3      , x6      ]);
3 constraint lex_lesseq([x1, x2, x3      ], [x4, x5, x6      ]);
4 constraint lex_lesseq([x1, x2, x3      ], [x6, x4, x5      ]);
5 constraint lex_lesseq([x1, x2, x3, x4      ], [x5, x6, x4, x2      ]);
6 constraint lex_lesseq([x1, x2, x3      ], [x4, x6, x5      ]);
7 constraint lex_lesseq([x1, x2, x3      ], [x5, x4, x6      ]);
8 constraint lex_lesseq([x1, x2, x3      ], [x6, x5, x4      ]);
9 % constraints 1 and 2: lexicographic ordering on the columns
10 % constraint 3:      lexicographic ordering on the rows
11 % constraint 4 bans the right array of the counterexample on slide 30

```



Example (Full variable symmetry)

For the $n!$ symmetries of the full symmetry group S_n , the $n! - 1$ n -ary `lex_lesseq` constraints (over arrays of size n) simplify into $n - 1$ binary \leq constraints (over integers):

$$x_1 \leq x_2 \wedge x_2 \leq x_3 \wedge \dots \leq x_n$$

Let the chosen ordering of the decision variables form a 1d array X , such as in the model for the subset problem of slide 8: use `strictly_increasing(X)`.

In practice:

Breaking **all** the symmetries may increase the solving time:

- Break only **some** symmetries, but which ones?
- **Double-lex** often works well on a 2d array X with full row and column symmetry: the `lex2(X)` constraint breaks the row symmetries and column symmetries, but **not** all their compositions; see the examples on slides 28 to 30.



Example (Rotation and reflection symmetry)

A **magic square** of order n is an $n \times n$ array containing all the integers 1 to n^2 exactly once, so that the sums of the rows, columns, and main diagonals are all equal (namely $\frac{n^2 \cdot (n^2 + 1)}{2 \cdot n}$).

For instance, a magic square M of order 3 has row sum 15:

2	9	4
7	5	3
6	1	8

Rotation and reflection symmetry-breaking constraint [↗](#):

```
constraint M[1,1] < M[1,n] /\ M[1,n] < M[n,1] /\ M[1,1] < M[n,n];
```



Structural Symmetry Breaking

Recall the (partitioned) map colouring problems of slide 7:

Example (Full or partial value symmetry; Law and Lee @ CP 2004)

Consider 1d array X of decision variables in the domain $1..k$:

- **Full value symmetry** over the full domain:

The first occurrences, if any, of the domain values are ordered:

```
constraint forall(i in 1..k-1) (value_precede(i, i+1, X));
```

or, logically equivalently but better:

```
constraint value_precede_chain(1..k, X);
```

- **Partial value symmetry** over the partitioned domain $D[1] \cup \dots \cup D[m]$:

```
constraint forall(i in 1..m) (value_precede_chain(D[i], X));
```

Think if dummy values are symmetric to non-dummy ones.



Example (Partial variable symmetry + full value symmetry @ CP 2006)

- Make study groups for two sets of five indistinguishable students each. There are six indistinguishable tables.
- The arrays P and M of decision variables, both with indices in $1..5$, correspond to the study groups and each decision variable is to be given a table identifier from the domain $1..6$.

- Variable-symmetry-breaking constraint:

```
constraint increasing(P) /\ increasing(M);
```

- Constraint on the table occupation counts by both groups:

```
constraint global_cardinality_closed(P, 1..6, CP) /\  
global_cardinality_closed(M, 1..6, CM);
```

- Value-symmetry-breaking constraint using those counts:

```
constraint forall(i in 1..(6-1))  
lex_greatereq([CP[i], CM[i]], [CP[i+1], CM[i+1]]);
```



Example (continued)

Consider the solution

$$P = [1, 1, 2, 3, 4]$$

$$M = [1, 2, 2, 3, 5]$$

The variable-symmetry-breaking constraint is satisfied:

$$\text{increasing}(P) \ / \ \text{increasing}(M)$$

and the value-symmetry-breaking constraint is satisfied:

$$[2, 1] \geq_{\text{lex}} [1, 2] \geq_{\text{lex}} [1, 1] \geq_{\text{lex}} [1, 0] \geq_{\text{lex}} [0, 1] \geq_{\text{lex}} [0, 0]$$

Note that a pointwise ordering would not have sufficed.



Example (continued)

If student $M[5]$ moves to table 6 from table 5, producing a symmetrically equivalent solution because the tables are fully indistinguishable:

$$P = [1, 1, 2, 3, 4]$$

$$M = [1, 2, 2, 3, 6]$$

then the value-symmetry-breaking constraint is violated:

$$[2, 1] \geq_{\text{lex}} [1, 2] \geq_{\text{lex}} [1, 1] \geq_{\text{lex}} [1, 0] \geq_{\text{lex}} [0, 0] \not\geq_{\text{lex}} [0, 1]$$



Example (end)

If students $M[4]$ and $M[5]$ swap their tables, producing a symmetrically equivalent solution because those students are fully indistinguishable:

$$P = [1, 1, 2, 3, 4]$$

$$M = [1, 2, 2, 5, 3]$$

then the table occupation counts do not change
and hence the value-symmetry-breaking constraint remains satisfied,
but the variable-symmetry-breaking constraint is violated, because

$$M[1] \leq M[2] \leq M[3] \leq M[4] \not\leq M[5]$$



Symmetry Breaking in MiniZinc

Good practice in MiniZinc: Flag symmetry-breaking constraints using the `symmetry_breaking_constraint` predicate. This allows backends to handle them differently, if wanted (see Topic 9: Modelling for CBLIS):

```
predicate symmetry_breaking_constraint (var bool: c) = c;
```

VS

```
predicate symmetry_breaking_constraint (var bool: c) = true;
```

Examples

```
1 constraint symmetry_breaking_constraint (increasing (X) );  
2 constraint symmetry_breaking_constraint (lex_lesseq (X, Y) );
```

Especially for MIP backends, try commenting away some symmetry-breaking constraints, as for `symmetry_breaking_constraint` the first definition above is currently used.



Outline

Introduction

Symmetry
Breaking by
Reformulation

Symmetry
Breaking by
Constraints

Conclusion

1. Introduction

2. Symmetry Breaking by Reformulation

3. Symmetry Breaking by Constraints

4. Conclusion



Évariste Galois (1811–1832)



Évariste Galois is one of the parents of group theory. Insight: The structure of the symmetries of an equation determines whether it has solutions or not.

Marginal note in his last paper:

“Il y a quelque chose à compléter dans cette démonstration. Je n’ai pas le temps.” (There is something to complete in this proof. I do not have the time.)



In Practice

- Are there few symmetries in real-life problems?
- Keep in mind the objective:
first solution, all solutions, or best solution?
Symmetry breaking might not pay off when searching for the first solution.
- Problem constraints can sometimes be simplified in the presence of symmetry-breaking constraints.
Example: $z = \text{abs}(x-y)$ can be simplified into $z = x-y$ if symmetry breaking requires $x \geq y$, thereby eliminating an undesirable disjunction, but upon `symmetry_breaking_constraint(x >= y)` this is incorrect for backends dropping such constraints.



Bibliography

Introduction

Symmetry
Breaking by
Reformulation

Symmetry
Breaking by
Constraints

Conclusion



D.A. Cohen, P. Jeavons, Ch. Jefferson, K.E. Petrie, and B.M. Smith.
Symmetry definitions for constraint satisfaction problems.
Constraints 11(2–3):115–137, 2006.



P. Flener, J. Pearson, and M. Sellmann.
Static and dynamic structural symmetry breaking.
Annals of Mathematics and Artificial Intelligence, 57(1):37–57, 2009.
(Supersedes our CP 2006 paper with P. Van Hentenryck.)