Structural Symmetry Handling

Pierre Flener

ASTRA Research Group
Uppsala University
Sweden

www.it.uu.se/research/group/astra

SymCon'09 – The 9th International Workshop on
Symmetry in (and?) Constraint Satisfaction Problems
Lisbon (Portugal), 20 September 2009

UPPSALA UNIVERSITET

Outline

Outline

Symmetry in Nature

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

UPPSALA UNIVERSITET

Broken Symmetry in Nature

The Angora cat originated in the Turkish city of Ankara. It is admired for its long silky coat and quiet graceful charm. It is often bred to favour a pale milky colouring, as well as one blue and one amber eye. (*Turkish Daily News*, 14 Oct 2001)

The Future of SymCon?

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

Joint work with (in Swedish alphabetical order):

- Justin Pearson
- Meinolf Sellmann
- Pascal Van Hentenryck
- Magnus Ågren

Outline

Structural Symmetry Breaking (SSB)

Definition: SSB = exploiting the combinatorial structure of a CSP toward eliminating, ideally in polynomial time & space, all symmetric sub-trees at every node explored (even if there are exponentially many symmetries):

- Dynamic structural symmetry breaking (DSSB): via dedicated search procedures
- Static structural symmetry breaking (SSSB): via constraints

Reminder: The general SBD* and lex-leader schemes may take exponential time or space if there are exponentially many symmetries.

Size Does Not Matter: A number of symmetries is **no** indicator of the difficulty of breaking them! The full group S_n has $n!$ easily broken symmetries; the cyclic group C_n has n symmetries that are more difficult to break.

Structural Symmetry Breaking (SSB)

Definition: SSB = exploiting the combinatorial structure of a CSP toward eliminating, ideally in polynomial time & space, all symmetric sub-trees at every node explored (even if there are exponentially many symmetries):

- Dynamic structural symmetry breaking (DSSB): via dedicated search procedures
- Static structural symmetry breaking (SSSB): via constraints

Reminder: The general SBD* and lex-leader schemes may take exponential time or space if there are exponentially many symmetries.

Size Does Not Matter: A number of symmetries is **no** indicator of the difficulty of breaking them! The full group S_n has n! easily broken symmetries; the cyclic group C_n has n symmetries that are more difficult to break.

UPPSALA UNIVERSITET

Definitions

Constraint satisfaction problems (CSPs):

- Scalar CSP: each variable takes a scalar value
- Set-CSP: each variable takes a set of scalar values

Symmetries:

- Full: any permutation (i.e., bijection) on the variables (or values) preserves solutions
- Partial: any piecewise permutation on the variables (or values) preserves solutions
  Ex: weekdays vs the weekend, same-capacity boats
- Wreath: any wreath permutation on the variables (or values) preserves solutions
  Ex: Schedule meetings in ($day, room$) pairs, where the days are interchangeable, and the rooms are interchangeable within each day
- Rotation: any rotation on the variables (or values) preserves solutions

Outline

Objective

Exploit the key strengths of CP (global constraints and search procedures) in order to break symmetries:

- Express the closure (under all considered symmetries) of a no-good (a partial assignment that cannot be extended into a solution) by global constraints:
  ☞ Abstract no-good
- Perform symmetry breaking during search (in the SBDD tradition), so as to avoid any interference with dynamic search heuristics (especially problem-specific ones):
  ☞ Symmetry-free search procedure

Full Value Symmetry

Example (European Map 4-Colouring)

Consider the two symmetric partial assignments:

$$Portugal = green, Spain = blue, France = green$$

$$Portugal = blue, Spain = red, France = blue$$

Not the values, but the **clustering** of the variables matters!
Compact representation, using (new) global constraints:

$$allEq(Portugal, France), allDiff(Portugal, Spain)$$

Abstract no-good, based on one representative variable in the *allDiff* constraint for each equivalence class

Full Value Symmetry

Properties of abstract no-goods for full value symmetry:

- Linear-size representation of the closure of a no-good
- Linear-time algorithm for violation testing
- As many as open nodes in the DFS/... search tree
- Specialisable, say for DFS: Re-consider the partial assignment $Pt = green, Es = blue, Fr = green$, with abstract no-good $allEq(Pt, Fr), allDiff(Pt, Es)$, and find a colour for Luxembourg (Lu):
  - if $Lu = green$: $allEq(Pt, Fr, Lu), allDiff(Pt, Es)$
  - if $Lu = blue$: $allEq(Pt, Fr), allEq(Es, Lu), allDiff(Pt, Es)$
  - if $Lu = red$: $allEq(Pt, Fr), allDiff(Pt, Es, Lu)$

  But the colours of Pt, Es, and Fr are known: simplify! For each variable, only the colours already used and **one** so far unused colour, say red, need to be tried.

Full Value Symmetry

Properties of abstract no-goods for full value symmetry:

- Linear-size representation of the closure of a no-good
- Linear-time algorithm for violation testing
- As many as open nodes in the DFS/… search tree
- Specialisable, say for DFS: Re-consider the partial assignment $Pt = green$, $Es = blue$, $Fr = green$, with abstract no-good $allEq(Pt, Fr), allDiff(Pt, Es)$, and find a colour for Luxembourg (Lu):
  - if $Lu = green$: $allEq(Pt, Fr, Lu), allDiff(Pt, Es)$
  - if $Lu = blue$: $allEq(Pt, Fr), allEq(Es, Lu), allDiff(Pt, Es)$
  - if $Lu = red$: $allEq(Pt, Fr), allDiff(Pt, Es, Lu)$

  But the colours of Pt, Es, and Fr are known: simplify! For each variable, only the colours already used and **one** so far unused colour, say red, need to be tried.

Full Value Symmetry

Properties of abstract no-goods for full value symmetry:

- Linear-size representation of the closure of a no-good
- Linear-time algorithm for violation testing
- As many as open nodes in the DFS/... search tree
- Specialisable, say for DFS: Re-consider the partial assignment $Pt = green, Es = blue, Fr = green$, with abstract no-good $allEq(Pt, Fr), allDiff(Pt, Es)$, and find a colour for Luxembourg (Lu):
  - if $Lu = green$: $allEq(Pt, Fr, Lu), allDiff(Pt, Es)$
  - if $Lu = blue$: $allEq(Pt, Fr), allEq(Es, Lu), allDiff(Pt, Es)$
  - if $Lu = red$: $allEq(Pt, Fr), allDiff(Pt, Es, Lu)$

  But the colours of Pt, Es, and Fr are known: simplify! For each variable, only the colours already used and **one** so far unused colour, say red, need to be tried.

Full Value Symmetry

Properties of abstract no-goods for full value symmetry:

- Linear-size representation of the closure of a no-good
- Linear-time algorithm for violation testing
- As many as open nodes in the DFS/... search tree
- Specialisable, say for DFS: Re-consider the partial assignment $Pt = green, Es = blue, Fr = green$, with abstract no-good $allEq(Pt, Fr), allDiff(Pt, Es)$, and find a colour for Luxembourg (Lu):
  - if $Lu = green$: $allEq(Pt, Fr, Lu), allDiff(Pt, Es)$
  - if $Lu = blue$: $allEq(Pt, Fr), allEq(Es, Lu), allDiff(Pt, Es)$
  - if $Lu = red$: $allEq(Pt, Fr), allDiff(Pt, Es, Lu)$

But the colours of Pt, Es, and Fr are known: simplify! For each variable, only the colours already used and **one** so far unused colour, say red, need to be tried.

Full Value Symmetry

Properties of abstract no-goods for full value symmetry:

- Linear-size representation of the closure of a no-good
- Linear-time algorithm for violation testing
- As many as open nodes in the DFS/... search tree
- Specialisable, say for DFS: Re-consider the partial assignment $Pt = green$, $Es = blue$, $Fr = green$, with abstract no-good $allEq(Pt, Fr), allDiff(Pt, Es)$, and find a colour for Luxembourg (Lu):
  - if $Lu = green$: $allEq(Pt, Fr, Lu), allDiff(Pt, Es)$
  - if $Lu = blue$: $allEq(Pt, Fr), allEq(Es, Lu), allDiff(Pt, Es)$
  - if $Lu = red$: $allEq(Pt, Fr), allDiff(Pt, Es, Lu)$

  But the colours of Pt, Es, and Fr are known: simplify!

  For each variable, only the colours already used and **one** so far unused colour, say red, need to be tried.

Full Value Symmetry

Properties of abstract no-goods for full value symmetry:

- Linear-size representation of the closure of a no-good
- Linear-time algorithm for violation testing
- As many as open nodes in the DFS/... search tree
- Specialisable, say for DFS: Re-consider the partial assignment $Pt = green, Es = blue, Fr = green$, with abstract no-good $allEq(Pt, Fr), allDiff(Pt, Es)$, and find a colour for Luxembourg (Lu):
  - if $Lu = green$: no-good simplifies into $Lu = green$
  - if $Lu = blue$: $allEq(Pt, Fr), allEq(Es, Lu), allDiff(Pt, Es)$
  - if $Lu = red$: $allEq(Pt, Fr), allDiff(Pt, Es, Lu)$

  But the colours of Pt, Es, and Fr are known: simplify!

  For each variable, only the colours already used and **one** so far unused colour, say red, need to be tried.

Full Value Symmetry

Properties of abstract no-goods for full value symmetry:

- Linear-size representation of the closure of a no-good
- Linear-time algorithm for violation testing
- As many as open nodes in the DFS/… search tree
- Specialisable, say for DFS: Re-consider the partial assignment $Pt = green, Es = blue, Fr = green$, with abstract no-good $allEq(Pt, Fr), allDiff(Pt, Es)$, and find a colour for Luxembourg (Lu):
  - if $Lu = green$: no-good simplifies into $Lu = green$
  - if $Lu = blue$: no-good simplifies into $Lu = blue$
  - if $Lu = red$: $allEq(Pt, Fr), allDiff(Pt, Es, Lu)$

  But the colours of Pt, Es, and Fr are known: simplify!

  For each variable, only the colours already used and **one** so far unused colour, say red, need to be tried.

Full Value Symmetry

Properties of abstract no-goods for full value symmetry:

- Linear-size representation of the closure of a no-good
- Linear-time algorithm for violation testing
- As many as open nodes in the DFS/… search tree
- Specialisable, say for DFS: Re-consider the partial assignment $Pt = green$, $Es = blue$, $Fr = green$, with abstract no-good $allEq(Pt, Fr), allDiff(Pt, Es)$, and find a colour for Luxembourg (Lu):
  - if $Lu = green$: no-good simplifies into $Lu = green$
  - if $Lu = blue$: no-good simplifies into $Lu = blue$
  - if $Lu = red$: no-g. simplifies into $allDiff(green, blue, Lu)$

  But the colours of Pt, Es, and Fr are known: simplify!

  For each variable, only the colours already used and **one** so far unused colour, say red, need to be tried.

Full Value Symmetry

Properties of abstract no-goods for full value symmetry:

- Linear-size representation of the closure of a no-good
- Linear-time algorithm for violation testing
- As many as open nodes in the DFS/… search tree
- Specialisable, say for DFS: Re-consider the partial assignment $Pt = green$, $Es = blue$, $Fr = green$, with abstract no-good $allEq(Pt, Fr), allDiff(Pt, Es)$, and find a colour for Luxembourg (Lu):
  - if $Lu = green$: no-good simplifies into $Lu = green$
  - if $Lu = blue$: no-good simplifies into $Lu = blue$
  - if $Lu = red$: no-g. simplifies into $allDiff(green, blue, Lu)$

  But the colours of Pt, Es, and Fr are known: simplify!

  For each variable, only the colours already used and **one** so far unused colour, say red, need to be tried.

UPPSALA UNIVERSITET

Full Value Symmetry (Er, 1988)

Initial call: $\mathit{list}(1, -1)$ for variables $X[1..n]$ over $0..k-1$

procedure $\mathit{list}(j, u$: **integer**) $\{u$ is the largest used value$\}$
if $j > n$ **then**
  return true {all symmetry is broken at all nodes!}
else
  try all $i = 0$ **to** $u + 1$ **do**
    $X[j] \leftarrow i;$
    $\mathit{list}(j + 1, \max(i, u))$

☞ Constant time & space at every node explored!

Variations: problem-specific (dynamic) var / val orders

Applications: (by P. Van Hentenryck [and L. Michel])

- Scene allocation (*INFORMS JOC*, 2002)
- Steel mill slab design (*CPAIOR'08*)

Partial Value Symmetry (*IJCAI'03*)

UPPSALA UNIVERSITET

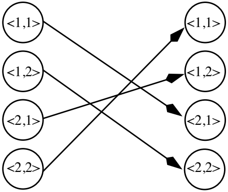
Let $D = D_1 + D_2 + \cdots + D_m$ be the domain of the variables, where the values in each set D_i are fully interchangeable. (Full value symmetry for $m = 1$.)

- Ex: weekdays vs the weekend, same-capacity boats
- Abstract no-goods: Variable clustering for each set D_i.
- Search procedure: For each variable, in each set D_i, only the values already used and **one** so far unused value need to be tried.
  ☞ Constant time & space at every node explored!
- Applications:
  - Eventually-serialisable data service deployment (L. Michel, *et al.*, and P. Van Hentenryck, *CPAIOR'08*)

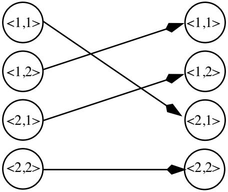
Wreath Value Symmetry (*IJCAI'03*)

Let $D = D_1 \times D_2$ be the domain of the decision variables, where the values in each set D_i are fully interchangeable. (Full value symmetry for $|D_2| = 1$.)

- Example: Schedule meetings in (*day*, *room*) pairs, where the days are interchangeable, and the rooms are interchangeable within each day



Wreath bijection



Not a wreath bijection!

Wreath Value Symmetry (*IJCAI'03*)

- Abstract no-goods: One abstract no-good on D_1, and m abstract no-goods on D_2 when m values of D_1 are used, with variable clustering as for full value symmetry.
- Search procedure: For each variable:
  1. For the first value component, in set D_1, only the values already used and **one** so far unused value need to be tried. Let $d_1 \in D_1$ be the chosen value.
  2. For the second value component, in set D_2, only the values already used with d_1 and **one** so far unused value need to be tried.

☞ Constant time & space at every node explored!

Selected Other Results

Consider a CSP with n variables over k values:

- Generalisation to any value symmetry: general equivalence (GE) trees (C. Roney-Dougal *et al.*, *ECAI'04*)
  ☞ $O(n^4)$ time overhead at every node explored.
- Partial variable symmetry and partial value symmetry (M. Sellmann and P. Van Hentenryck, *IJCAI'05*)
  ☞ $O(k^{2.5} + nk)$ time overhead at every node explored.
  ☞ Coinage of the term 'structural symmetry breaking'.

Tractability of DSSB: State of the Art

value symmetry \ variable symmetry	none	full	partial	wreath	
none		P	P	P	scalar CSP
		P	**P**	**P**	**set-CSP**
full	P	P	P	NP	scalar CSP
	P	**NP**	**NP**	**NP**	**set-CSP**
partial	P	P	P	NP	scalar CSP
	P	**NP**	**NP**	**NP**	**set-CSP**
wreath	P	P	P	NP	scalar CSP
	P	**NP**	**NP**	**NP**	**set-CSP**
any	P				scalar CSP
					set-CSP

P: All symmetric sub-trees can be eliminated, say by DSSB, with a polynomial time & space overhead at every node explored.

NP: Dominance-detection schemes like SBDD are NP-hard.

Combinatorial Generation

Listing, ranking, unranking, and randomly selecting the objects of some combinatorial structure (combination, partition, permutation, subset, tree, etc) w.r.t. some order:

- Constant amortised time (CAT): in time proportional to the number of objects listed (after some initialisations)
- Backtracking ensuring success at terminals (BEST): every leaf of the backtracking tree is a desired object
- Loopless: the next object is constructed without executing any loop
- Memoryless: the next object is constructed without using any global variables (can start from any object)

Reference: Frank Ruskey, *Combinatorial Generation*, unpublished book draft, 2003. Available as www.1stworks.com/ref/RuskeyCombGen.pdf.

Combinatorial Objects

Consider the sequences of $n = 3$ beads over $k = 2$ colours:

tuples	unlabelled tuples	necklaces	unlabelled necklaces
000	000	000	000
001	001	001	001
010	010		
011	011	011	
100			
101			
110			
111		111	
no symmetry	full value symmetry	rotation variable symmetry	rot var + full val symmetry

Combinatorial Objects

Consider the sequences of $n = 3$ beads over $k = 2$ colours:

tuples	unlabelled tuples	necklaces	unlabelled necklaces
000	000	000	000
001	001	001	001
010	010		
011	011	011	~~011~~
100			
101			
110			
111		111	
no symmetry	full value symmetry	rotation variable symmetry	rot var + full val symmetry

Unlabelled Tuples (Er, 1988)

procedure $list(j, u : \textbf{integer})$ {u is the largest used value}
var i : **integer**
if $j > n$ **then**
  return true {all sym broken at all nodes!}
else
  try all $i = 0$ **to** $\min(u+1, k-1)$ **do**
    if true then
      $X[j] \leftarrow i$;
      $list(j+1, \max(i, u))$

Initial call: $list(1, -1)$
Complexity: Constant amortised time & space:

$$\#objects = \#unlabelled\ tuples$$

Necklaces (Ruskey & Sawada, 2000)

procedure $list(j, p : \textbf{integer})$ {p = #positions to replicate}
var i : **integer**
if $j > n$ **then**
  return $n \bmod p = 0$ {not all sym broken at all nodes!}
else
  try all $i = X[j-p]$ **to** $k-1$ **do**
    if true then
      $X[j] \leftarrow i$;
      $list(j+1, \textbf{if } i = X[j-p] \textbf{ then } p \textbf{ else } j)$

Initial call: $X[0] \leftarrow 0; list(1, 1)$, where $X[0]$ is a dummy
Complexity: Constant amortised time & space:

$$\#objects \leq \#necklaces \cdot (k/(k-1))^2$$

Unlabelled Necklaces (*ECAI'08*)

procedure $list(j, p, u : \textbf{integer})$
var i : **integer**
if $j > n$ **then**
  return $n \bmod p = 0$ {not all sym broken at all nodes!}
else
  try all $i = X[j-p]$ **to** $\min(u+1, k-1)$ **do**
    if $probe(j, i, p)$ **then**
      $X[j] \leftarrow i$;
      $list(j+1, \textbf{if } i = X[j-p] \textbf{ then } p \textbf{ else } j, \max(i, u))$

Initial call: $X[0] \leftarrow 0; list(1, 1, -1)$, where $X[0]$ is a dummy
Complexity: ?

$$\#objects \; ? \; \#unlabelled\ necklaces$$

UPPSALA UNIVERSITET

Outline

Main Results (*CP'06*)

Consider a CSP with n variables over $k = \ell \cdot m$ values:

- Partial variable symmetry and partial value symmetry: $O(n + k)$ constraints break $O(n! \cdot k!)$ symmetries
- Some theorems comparing SSSB with DSSB upon static variable and value orderings
- Generalisation:
  Partial variable symmetry and wreath value symmetry: $O(n + k)$ constraints break $O(n! \cdot (m!)^\ell \cdot \ell!)$ symmetries

Example (Partial Variable & Full Value Symmetry)

- Make study groups for two sets of five indistinguishable students each. There are six indistinguishable tables.
- The decision variables $\{f_1, \dots, f_5\} \cup \{m_6, \dots, m_{10}\}$ correspond to the students and are to be assigned table values from the ordered domain $\{t_1, \dots, t_6\}$.
- Constraints breaking the variable symmetries:

$$f_1 \leq f_2 \leq f_3 \leq f_4 \leq f_5 \; \& \; m_6 \leq m_7 \leq m_8 \leq m_9 \leq m_{10}$$

- Constraints computing the signatures (counters):

$$gcc(f_1, \dots, f_5,\ t_1, \dots, t_6,\ c_1^f, \dots, c_6^f)$$
$$gcc(m_6, \dots, m_{10},\ t_1, \dots, t_6,\ c_1^m, \dots, c_6^m)$$

- Constraints breaking the value symmetries:

$$(c_1^f, c_1^m) \geq_{lex} \cdots \geq_{lex} (c_6^f, c_6^m)$$

Example (Partial Variable & Full Value Symmetry)

Consider the satisfying assignment

$$\{f_1 \mapsto t_1, f_2 \mapsto t_1, f_3 \mapsto t_2, f_4 \mapsto t_3, f_5 \mapsto t_4, \\ m_6 \mapsto t_1, m_7 \mapsto t_2, m_8 \mapsto t_2, m_9 \mapsto t_3, m_{10} \mapsto t_5\}.$$

Indeed, the variable-symmetry constraints are satisfied:

$$f_1 \leq f_2 \leq f_3 \leq f_4 \leq f_5 \ \& \ m_6 \leq m_7 \leq m_8 \leq m_9 \leq m_{10}$$

and the value-symmetry constraints are satisfied:

$$(2, 1) \geq_{lex} (1, 2) \geq_{lex} (1, 1) \geq_{lex} (1, 0) \geq_{lex} (0, 1) \geq_{lex} (0, 0)$$

Note that a pointwise ordering would not have sufficed.

Example (Partial Variable & Full Value Symmetry)

If student m_{10} moves from table t_5 to table t_6, producing a symmetrically equivalent assignment because the tables are fully interchangeable:

$$\{f_1 \mapsto t_1, f_2 \mapsto t_1, f_3 \mapsto t_2, f_4 \mapsto t_3, f_5 \mapsto t_4,\\ m_6 \mapsto t_1, m_7 \mapsto t_2, m_8 \mapsto t_2, m_9 \mapsto t_3, m_{10} \mapsto t_6\}$$

then the value-symmetry constraints are violated:

$$(2,1) \geq_{lex} (1,2) \geq_{lex} (1,1) \geq_{lex} (1,0) \geq_{lex} (0,0) \not\geq_{lex} (0,1)$$

Example (Partial Variable & Full Value Symmetry)

If students m_9 and m_{10} swap their assigned tables, producing a symmetrically equivalent assignment because both students are male:

$$\{f_1 \mapsto t_1, f_2 \mapsto t_1, f_3 \mapsto t_2, f_4 \mapsto t_3, f_5 \mapsto t_4,$$
$$m_6 \mapsto t_1, m_7 \mapsto t_2, m_8 \mapsto t_2, m_9 \mapsto t_5, m_{10} \mapsto t_3\}$$

then the signatures do not change and hence the value-symmetry constraints remain satisfied, but the variable-symmetry constraints are violated, because

$$m_6 \leq m_7 \leq m_8 \leq m_9 \not\leq m_{10}$$

UPPSALA UNIVERSITET

Outline

Structural Symmetry Detection (SSD)

Definition: SSD = exploiting the combinatorial structure of a CSP toward deriving, ideally in polynomial time & space, (some of) the symmetries of the CSP (even if there are exponentially many derived symmetries):

- Static structural symmetry detection (SSSD): before solving
- Dynamic structural symmetry detection (DSSD): while solving

Reminder: General symmetry detection schemes are graph-isomorphism complete.

Outline

Bottom-Up Derivation (*SARA'05*)

Key insight: Once the symmetries of (global) constraints and functions are identified (manually), the symmetries of a constraint model with these constraints and functions can be derived compositionally, automatically, and efficiently:

- Symmetry identification
- Symmetry composition

A subset of our results turned out to be in (P. Roy and F. Pachet, *ECAI'98 Workshop on Non-Binary Constraints*).

Symmetry Identification

Consider a CSP or COP with variables X over domain D:

- Constraint $allDiff(x_1, \ldots, x_n)$ has full value symmetry.
- Function $nbDistinct(x_1, \ldots, x_n)$ has full value symmetry.
- Constraint $atMost(m, d, [x_1, \ldots, x_n])$ has partial value symmetry over the partition $\{\{d\}, D \setminus \{d\}\}$ of D (at most m occurrences of d among the variables x_i).
- Constraint $x_1 < x_2$ has partial variable symmetry over the partition $\{\{x_1\}, \{x_2\}, X \setminus \{x_1, x_2\}\}$ of X.

Similarly for row and column symmetries.

☞ Extend the Global Constraint Catalogue accordingly!

Symmetry Composition

Consider a CSP/COP with variables X over $D = \{a, \ldots, h\}$:

- If the constraints c_1 and c_2 have full value symmetry, then their conjunction $c_1 \wedge c_2$ has full value symmetry.
- Extension to functions. Example: The expression $3 \cdot nbDistinct(x_1, x_2, x_3) + 4 \cdot nbDistinct(x_4, x_5, x_6)$ has full value symmetry.
- Generalisation to partial symmetry (PS). Example: $atMost(i, a, X)$ has PS over $\{\{a\}, \{b, c, \ldots, h\}\}$, and $atMost(j, b, X)$ has PS over $\{\{b\}, \{a, c, \ldots, h\}\}$, so their conj. has PS over $\{\{a\}, \{b\}, \{c, \ldots, h\}\}$ if $i \neq j$, but PS over $\{\{a, b\}, \{c, \ldots, h\}\}$ if $i = j$:
  ☞ Need for prior constraint aggregation into $atMost([i, j], [a, b], X)$.

☞ Each composition takes time polynomial in $|X| + |D|$.

Applications

- Scene Allocation Problem:
  full value symmetry (indistinguishable days)
- Progressive Party Problem:
  partial row symmetry (same-size guest crews),
  full column symmetry (interchangeable periods), and
  partial value symmetry (same-capacity host boats)

Outline

1. Prelude
2. Structural Symmetry Breaking
   - Dynamic Structural Symmetry Breaking
   - Static Structural Symmetry Breaking
3. **Structural Symmetry Detection**
   - Static Structural Symmetry Detection
   - **Dynamic Structural Symmetry Detection**
4. Postlude

Constraint Projection

Example: Consider a CSP or COP with decision variables $X = [x_1, x_2, \ldots, x_n]$ over domain $D = \{a, \ldots, h\}$:

- The constraint $atMost([3, 2], [a, b], X)$ has partial value symmetry over $\{\{a\}, \{b\}, \{c, \ldots, h\}\}$.
- The decision $x_1 = a$ has partial value symmetry over $\{\{a\}, \{b, \ldots, h\}\}$.
- Their conjunction thus has partial value symmetry over $\{\{a\}, \{b\}, \{c, \ldots, h\}\}$.
- Projection onto $X \setminus \{x_1\}$ of the original constraint gives $atMost([2, 2], [a, b], [x_2, \ldots, x_n])$, which has partial value symmetry over $\{\{a, b\}, \{c, \ldots, h\}\}$:
  a new symmetry was dynamically detected!

Évariste Galois (1811–1832)

Évariste Galois was 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 demonstration.
I do not have the time.)

Outline

Contributions

- Tractability map of SBDD-style symmetry breaking
- Symmetry-free search procedures for CSP classes
- Combinatorial generation
  = dynamic structural symmetry breaking
- First CSP classes where a polynomial (and even linear) number of constraints break an exponential number of compositions of variable and value symmetries
- Structural detection of ~~the~~ symmetries of a model

Future Work: Other Orders

- Lexicographic order: $12 <_{lex} 13 <_{lex} 21$

$$000, 001, 010, 011, 100, 101, 110, 111$$

- Co-lexicographic order: $21 <_{colex} 12 <_{colex} 13$
  ☞ Shorter, faster, more elegant and natural algorithms!
- Gray order (F. Gray, *US Patent* 2,632,058, 1953):

$$\underline{0}00, 00\underline{1}, 0\underline{1}1, 01\underline{0}, \underline{1}10, 11\underline{1}, 1\underline{0}1, 10\underline{0}$$

  ☞ Only one value (underlined) changes each time!
- Boustrophedonic order (Ph. Flajolet *et al.*, *TCS*, 1994): turning like oxen in ploughing; the writing of alternate lines in opposite directions (Merriam-Webster)

Used for listing objects in combinatorial generation (DSSB), but can / should be turned into constraints (for SSSB)!

Future Work

- Exploiting other orders (Gray, boustrophedonic, ...)
- Incomplete symmetry breaking
- Push symmetry breaking into global constraints
- Symmetry detection & breaking in CP systems

Acknowledgements

STINT The Swedish Foundation for International Cooperation in Research and Higher Education

- CORSA project (www.it.uu.se/research/group/astra/CORSA)
- Uppsala University and Brown University
- Institutional Grant IG2001-67
- From September 2001 to December 2007

UPPSALA UNIVERSITET

Main References

- P. Flener, J. Pearson, and M. Sellmann.
  Static and dynamic structural symmetry breaking.
  Annals of Mathematics and Artificial Intelligence, 20xx.
  (Supersedes our CP’06 paper with P. Van Hentenryck.)

- P. Flener, J. Pearson, M. Sellmann, P. Van Hentenryck, M. Ågren.
  Dynamic SSB for constraint satisfaction problems.
  Constraints, 14(4), December 2009.
  (Supersedes our IJCAI’03 and IJCAI’05 papers.)

- P. Flener and J. Pearson.
  Solving necklace constraint problems.
  Journal of Algorithms, 64(2–3):61–73, April–July 2009.
  (Supersedes our ECAI’08 paper.)

- P. Van Hentenryck, P. Flener, J. Pearson, and M. Ågren.
  Compositional derivation of symmetries for constraint satisfaction.
  Proc. of SARA’05. LNAI 3607:234–247. Springer-Verlag, 2005.