CS4402: Constraint Programming

Week 4, Lecture 1: Symmetry

Check MMS for actual c/w deadlines

Roadmap

Joan Weeks (2–4)

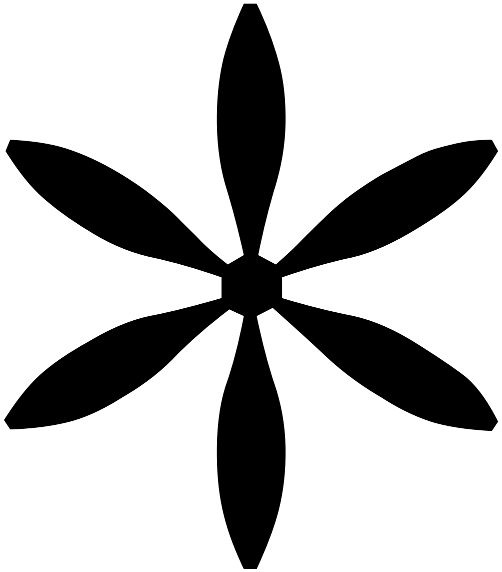
1	2	3	4	5	6	7
Cwork 1 Out			We are Here!		ILW	Cw 1 In Cw 2 Out
			Cwork 2 In	Exam Replacement		
8	9	10	11			

Basics | Modelling | Search and Propagation

Symmetry in Constraint Models

Symmetry

- A familiar everyday concept:



- A **structure-preserving transformation**.

Symmetry in Constraint Models

- A symmetry can be characterised as a **bijection** on assignments.
  - i.e. a one-to-one mapping.
- Partitions complete assignments into **equivalence classes**.
  - In every class, all members are solutions, or no member is.
- That is, symmetry **preserves solutionhood**.

Symmetry in Constraint Models

- Why do we care?
- Frequently present in the structure of a CSP.
- If we don't deal with it, can lead to a lot of wasted effort for systematic search.

Two Common Special Cases

- **Variable** symmetry.
  - Bijection can be characterised in terms of variables alone.
- **Value** symmetry.
  - Bijection can be characterised in terms of values alone.

Variable Symmetry

Variable Symmetry Example

	A Solution			
q1		Q		
q2				Q
q3	Q			
q4			Q	

Flip Horizontal →

	A Solution			
q4		Q		
q3				Q
q2	Q			
q1			Q	

- We can express this symmetry in terms of a bijection on the **variables**:
  - q1 → q4,
  - q2 → q3,
  - q3 → q2,
  - q4 → q1
- A bijection on the **values** would capture the transformation of this particular solution – but not the **general** horizontal flip.

Equivalently:

	A Solution			
q1			Q	
q2	Q			
q3				Q
q4		Q		

Variable Symmetry Example

	Non-solution			
q1		Q		
q2				Q
q3	Q			
q4		Q		

Flip Horizontal

	Non-solution			
q4		Q		
q3				Q
q2	Q			
q1		Q		

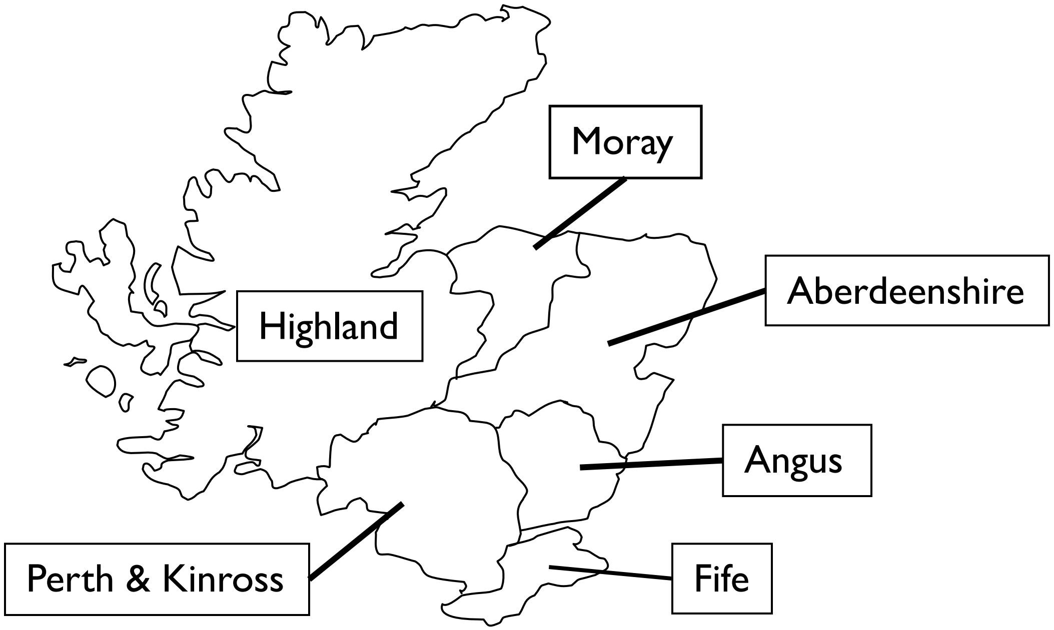
Equivalently

	Non-solution			
q1		Q		
q2	Q			
q3				Q
q4		Q		

- We can express this symmetry in terms of a bijection on the **variables**:
  - q1 → q4,
  - q2 → q3,
  - q3 → q2,
  - q4 → q1

Value Symmetry

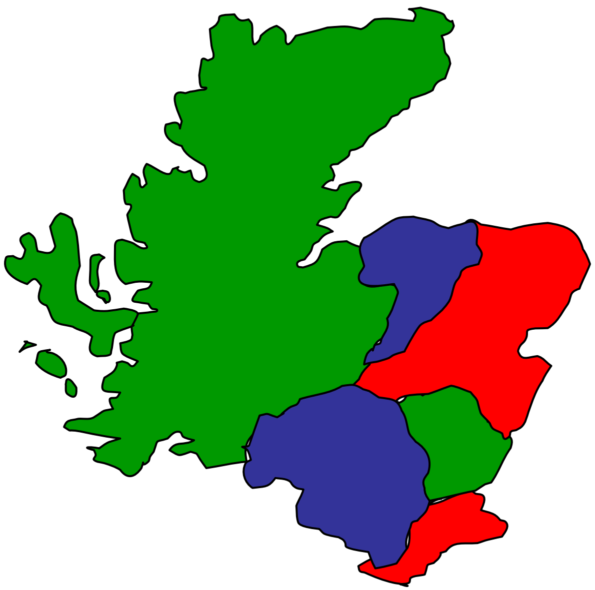
A Graph Colouring Problem



- We want to colour this map with the colours red green and blue such that adjacent regions are not coloured the same.

A Solution

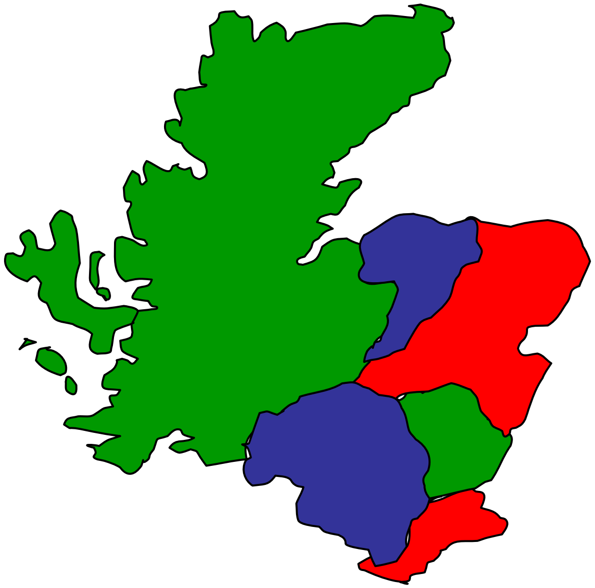
- Highland = green
- Moray = blue
- Aberdeenshire = red
- Angus = green
- Fife = red
- Perth and Kinross = blue



Can you see the symmetry?

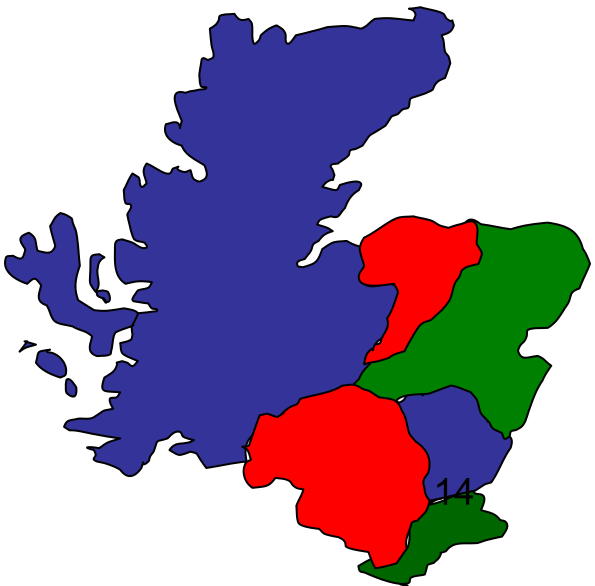
Symmetric Solutions

- Highland = green
- Moray = blue
- Aberdeenshire = red
- Angus = green
- Fife = red
- Perth and Kinross = blue



Red → green
green → blue
blue → red

- Highland = blue
- Moray = red
- Aberdeenshire = green
- Angus = blue
- Fife = green
- Perth and Kinross = red

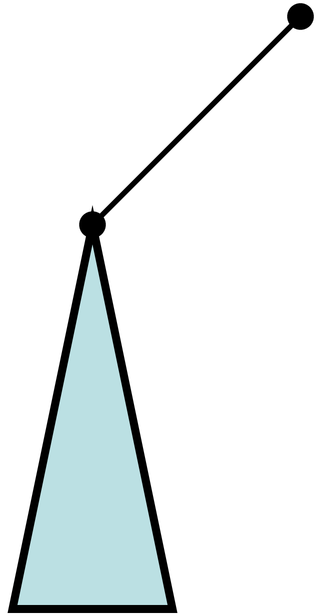


We' ve permuted the **values**.

Symmetry: Consequences

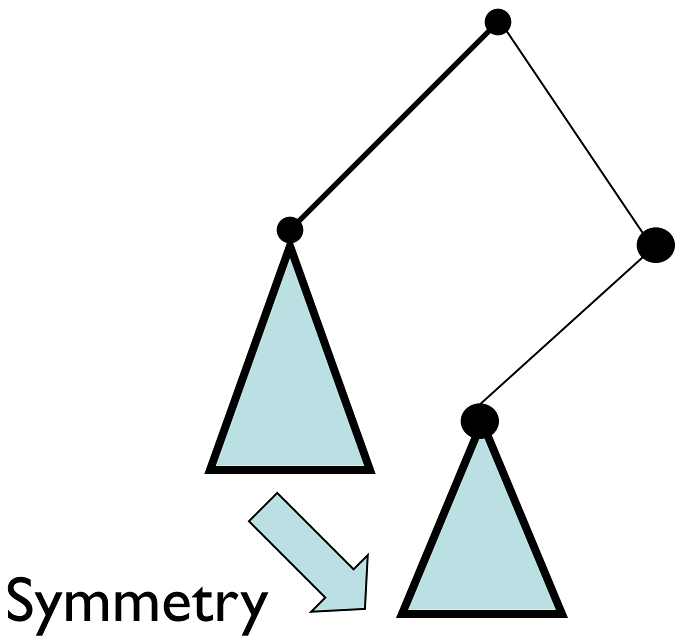
What are The Consequences of a Model Containing Symmetry?

- Assume we have explored a sub-tree in which there are no solutions:



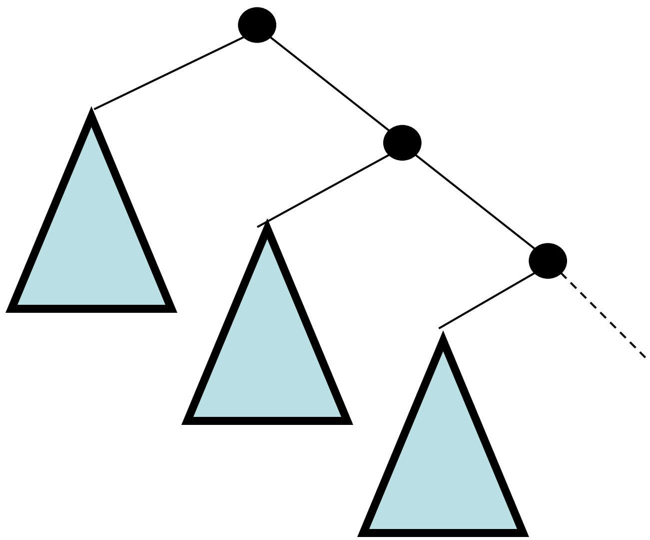
What are The Consequences of a Model Containing Symmetry?

- Symmetries in the model can map this fruitless subtree into another:



What are The Consequences of a Model Containing Symmetry?

- If we don't deal with the symmetry, a systematic search will explore all symmetric variants of this sub-tree:



Symmetry: Origins

How Does Symmetry Arise?

- Symmetry enters a model from 2 sources:
  1. It is inherent in the problem:

	A Solution			
q1		Q		
q2				Q
q3	Q			
q4			Q	

By its nature, the *n*-queens problem has the symmetries of the square

How Does Symmetry Arise?

- Symmetry enters a model from 2 sources:
  2. It is introduced during the modelling process.
- Remember the explicit model of sets:

1	2	3	4	…	n-1	n
0..m	0..m	0..m	0..m		0..m	0..m

AllDifferent

- Now there is a 1st element, a 2nd element, etc…

How Does Symmetry Arise?

- Given any (non-)solution:

1	2	3	4
a	b	c	d

- Can permute the elements to obtain a (non-)solution:

1	2	3	4
b	d	c	a

NB This is again a variable symmetry: we have given arbitrarily named indices to the elements of the set.

NB Knowing how symmetry is introduced during modelling can make some symmetry detection easy.

Symmetry-breaking Constraints

How Can Adding Constraints Affect Symmetry?

- By disallowing some of the elements of an **equivalence class** of assignments.

M	1	2	3	4	…	n-1	n
	0..m	0..m	0..m	0..m		0..m	0..m

AllDifferent(M)

- Consider ordering the elements of M:

$$M[1] \leq M[2] \leq M[3] \leq \ldots \leq M[n-1] \leq M[n]$$

How Can Adding Constraints Affect Symmetry?

- Ordering the elements distinguishes one element in each equivalence class of assignments:

1	2	3	4
a	b	c	d

1	2	3	4
b	d	c	a

- If we can propagate symmetry-breaking constraints effectively, they can prune the search tree drastically.

Valid Symmetry-breaking Constraints: Properties

- Leave at least one element in each equivalence class of assignments.
- So they **throw away solutions** to the original problem?
  - Yes: but we can recover those solutions by applying the symmetry to the solutions allowed.
- If we add enough symmetry-breaking constraints we can break all symmetry.

How do we Formulate Symmetry-breaking Constraints?

- Define a canonical solution and add constraints to choose it.
- Need:
  - An ordering on **variables**.
  - An ordering on **solutions**.

Orderings on Variables

- Simple: we just choose one.

M	1	2	3	4	…	n-1	n
	0..m	0..m	0..m	0..m		0..m	0..m

- Here, the natural thing to do is choose an ordering along the indices

Orderings on the Solutions

- Use a **lexicographic** ordering:

v1	1	2	3	4
	a	b	c	d

$v_1 \leq_{lex} v_2$

v2	1	2	3	4
	b	d	c	a

The Lex-leader Method

- Choose an ordering on the variables.
- Add one lexicographic ordering constraint per symmetry.
- Good news: this breaks all symmetry.
- Bad news: you might need a huge number of such constraints
  - But sometimes the lex constraints collapse into a simpler form (e.g. the explicit set representation).
- Compromise: use a subset of the lex-leader constraints.

Lex Leader Example

A	B	C
D	E	F

- Assume that this matrix has row and column symmetry.
- Choose ordering on variables ABCDEF

Lex Leader Example

A	B	C
D	E	F

- ABCDEF <=lex DEFABC (swap rows).
- ABCDEF <=lex ACBDFE (swap last two cols).
- ABCDEF <=lex DFEACB (swap rows & last 2 cols).
- …
- We are preventing it from being possible to obtain a “smaller” assignment than the one we currently have by applying a symmetry.

The BIBD has Row and Column Symmetry

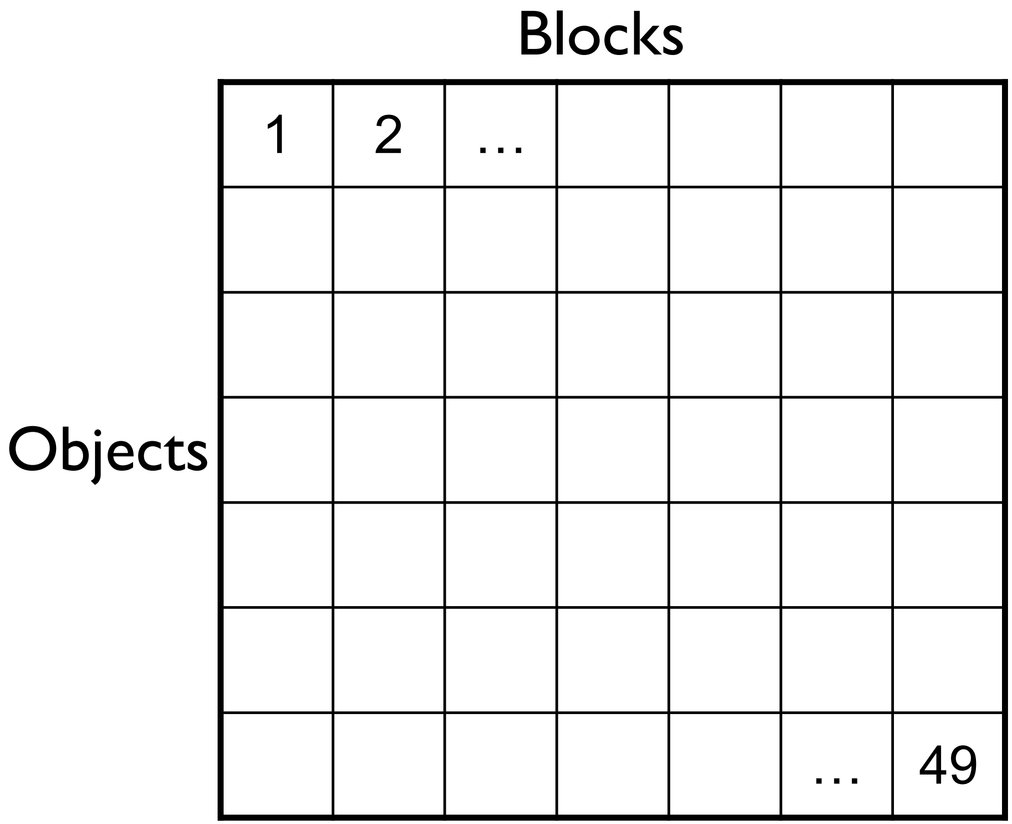
- This is **very** common in matrix models.

0	0	0	0	1	1	1
0	0	1	1	0	0	1
0	1	0	1	0	1	0
0	1	1	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	1	0	0	0	0	1

0	0	1	0	1	0	1
0	0	0	1	0	1	1
0	1	0	0	1	1	0
0	1	1	1	0	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	1	0	0	0	0	1

Lex-leader for BIBD

- Requires ***b***!***v***! lex constraints.



Lex^2 ordering for Row/Column Symmetry

- A significant fraction of lex-leader for row/column symmetry can be represented by lex ordering the rows and columns.

0	0	0	0	1	1	1
0	0	1	1	0	0	1
0	1	0	1	0	1	0
0	1	1	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	1	0	0	0	0	1

0	0	1	0	1	0	1
0	0	0	1	0	1	1
0	1	0	0	1	1	0
0	1	1	1	0	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	1	0	0	0	0	1

Simple example of Lex^2

A	B	C
D	E	F

i. **BE $\leq$ CF**

ii. **AD $\leq$ BE**

iii. **ABC $\leq$ DEF**

Lex Leader Method: On the same example

A	B	C
D	E	F

1. ABCDEF $\leq$ ACBDFE
2. ABCDEF $\leq$ BCAEFD
3. ABCDEF $\leq$ BACEDF
4. ABCDEF $\leq$ CABFDE
5. ABCDEF $\leq$ CBAFED
6. ABCDEF $\leq$ DFEACB
7. ABCDEF $\leq$ EFDBCA
8. ABCDEF $\leq$ EDFBAC
9. ABCDEF $\leq$ FDECAB
10. ABCDEF $\leq$ FEDCBA
11. ABCDEF $\leq$ DEFABC

Lex Leader vs Lex^2

A	B	C
D	E	F

i. BE $\leq$ CF
ii. AD $\leq$ BE
iii. ABC $\leq$ DEF

1. ABCDEF $\leq$ ACBDFE
2. ABCDEF $\leq$ BCAEFD
3. ABCDEF $\leq$ BACEDF
4. ABCDEF $\leq$ CABFDE
5. ABCDEF $\leq$ CBAFED
6. ABCDEF $\leq$ DFEACB
7. ABCDEF $\leq$ EFDBCA
8. ABCDEF $\leq$ EDFBAC
9. ABCDEF $\leq$ FDECAB
10. ABCDEF $\leq$ FEDCBA
11. ABCDEF $\leq$ DEFABC

Lex Leader Method: Connection to Lex^2

A	B	C
D	E	F

i. **BE $\leq$ CF**

ii. **AD $\leq$ BE**

iii. **ABC $\leq$ DEF**

1. A**B**CD**E**F $\leq$ A**C**BD**F**E
2. ABCDEF $\leq$ BCAEFD
3. **A**BC**D**EF $\leq$ **B**AC**E**DF
4. ABCDEF $\leq$ CABFDE
5. ABCDEF $\leq$ CBAFED
6. ABCDEF $\leq$ DFEACB
7. ABCDEF $\leq$ EFDBCA
8. ABCDEF $\leq$ EDFBAC
9. ABCDEF $\leq$ FDECAB
10. ABCDEF $\leq$ FEDCBA
11. **ABC**DEF $\leq$ **DEF**ABC

Lex Leader Method: Connection to Lex^2

i. BE $\leq$ CF

A	B	C
D	E	F

I. A**B**CD**E**F $\leq$ A**C**BD**F**E

- Assume A**B**CD**E**F $\leq$ A**C**BD**F**E
  - A always equals A and D always equals D
- So **B**C**E**F $\leq$ **C**B**F**E
  - If B = C then obviously C=B
- So **BE**F $\leq$ **CF**E
  - If E = F then obviously F=E
- So **BE** $\leq$ **CF**

Lex Leader Method: Connection to Lex^2

i. BE $\leq$ CF

A	B	C
D	E	F

1. A**B**CD**E**F $\leq$ A**C**BD**F**E

- These two constraints are actually *equivalent.*
- Similarly for the other constraints in Lex^2

- But Lex Leader has additional constraints
- So Lex Leader sometimes breaks *more* symmetry than Lex^2
- While the extra constraints means it can be much more expensive to reason with than Lex^2