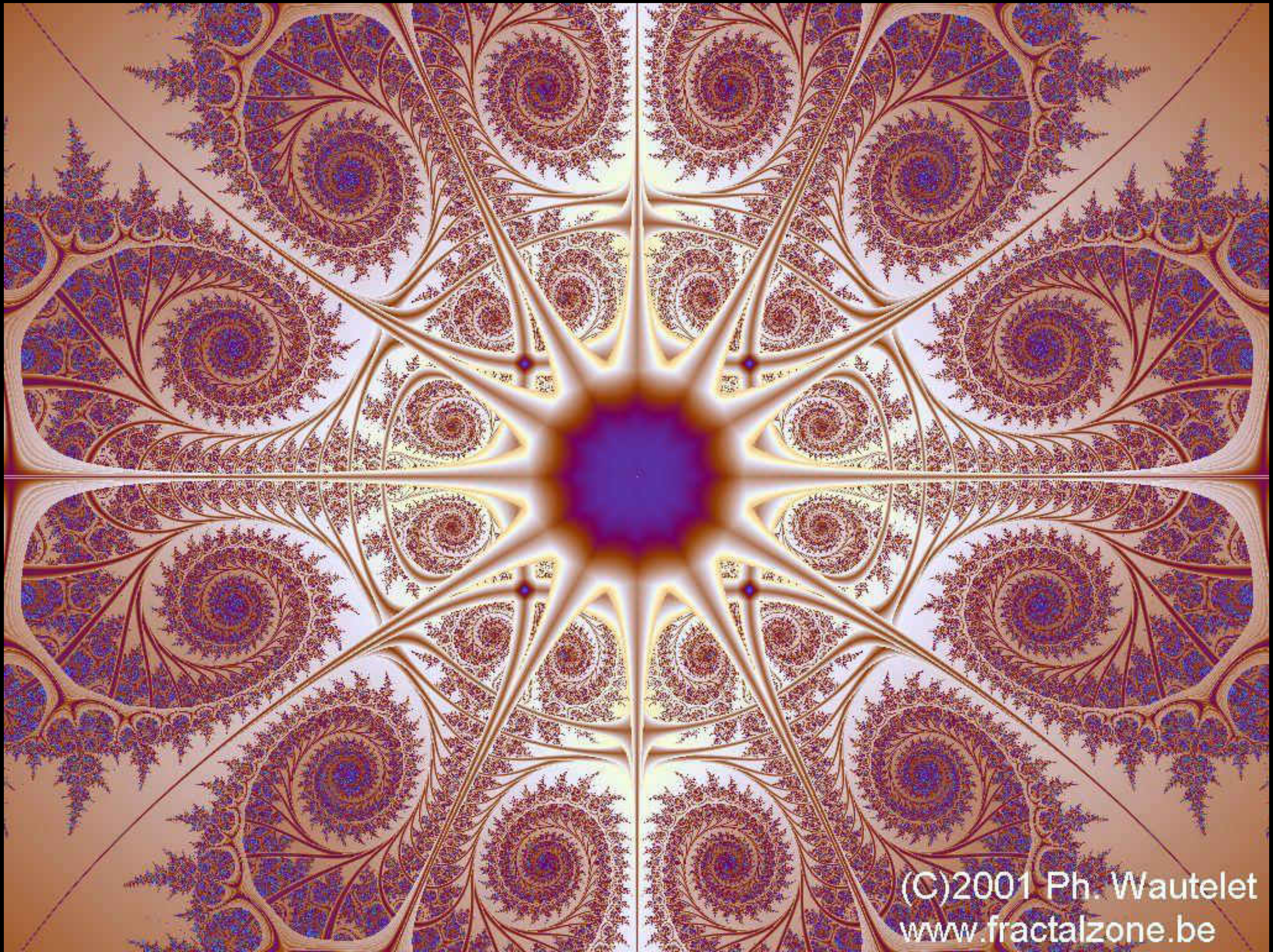
Symmetry

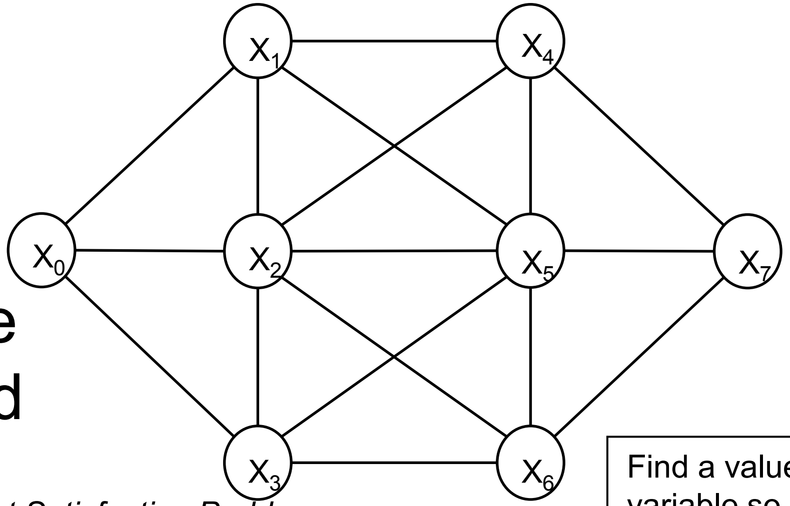


Crystal maze revisited

Crystal Maze Revisited

The graph is symmetric.
Given any solution, we can get another solution by reflecting in the vertical axis, or by reflecting in the horizontal axis, or by rotating 180 degree.

Similarly, the values are symmetrical. Given any solution, we know we can get another by swapping value 1 with 8, 2 with 7, 3 with 6 and 4 with 5.



A *Constraint Satisfaction Problem*:

Variables: $\{X_0, X_1, X_2, X_3, X_4, X_5, X_6, X_7\}$
Values: {1,2,3,4,5,6,7,8}
Constraints: $\{\text{alldifferent}(\{X_0, X_1, X_2, X_3, X_4, X_5, X_6, X_7\}), \ldots, |X_0-X_1|>1, |X_0-X_2|>1, \ldots\}$

Find a value for each variable so that all constraints are satisfied simultaneously

Impact of symmetry on the search

If we are searching for all solutions, then we would be better searching for one solution, and then generating others by swapping the values or the assignments.

Similarly, suppose we discover that $X_2 = 2$ has no solution. Then we know that there is no solution where $X_5 = 2$.

Similarly, if there is no solution in which X_i takes value 1, then there will also be no solution in which X_i takes value 8 (and similarly for the pairs (2,7), (3,6) and (4,5)).

Breaking symmetry by posting constraints

Any solution with $X_2 < X_5$ will have a corresponding solution with $X_2 > X_5$.

$\Rightarrow$ post a new constraint $X_2 < X_5$ onto the model.

Even then, for any solution with $X_1 < X_3$, there will be an equivalent solution with $X_1 > X_3$

$\Rightarrow$ post a constraint $X_1 < X_3$

Magic Square

Many symmetries in the magic square problem
-rotate 90
-rotate 180
-rotate 270
-reflect in horizontal
-reflect in vertical
-reflect in down-right diagonal
-reflect in up-right diagonal

We modelled the problem as a matrix, and posted the constraints $s[0][0] < s[0][n-1]$, $s[0][0] < s[n-1][0]$, $s[0][0] < s[n-1][n-1]$, and $s[0][n-1] < s[n-1][0]$

Symmetry in constraint solving

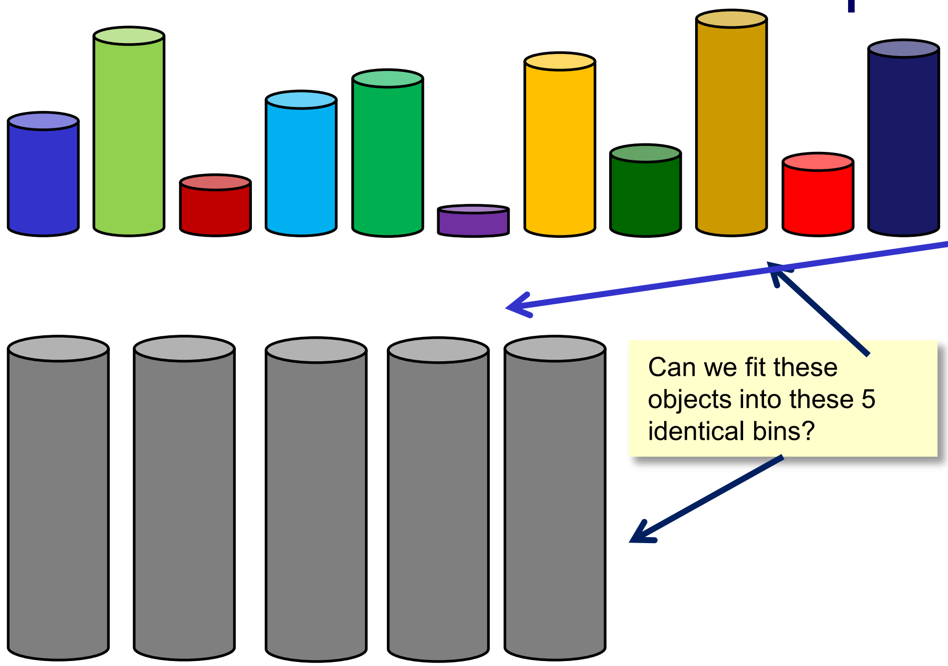
Add symmetry breaking constraints to the model

- leave at least one solution
- remove some or all of the symmetric ones

Modify the search procedure to detect symmetries during search

We will focus on adding symmetry breaking constraints

Bin packing



the bins are identical ...

Can we fit these objects into these 5 identical bins?

"Matrix model"

bin_i contains object_j if and only if table[i][j] == 1

	obj 0	obj 1	obj 2	...	obj (n-1)
bin 0	{0,1}	{0,1}	{0,1}		{0,1}
bin 1	{0,1}				{0,1}
...	...				
bin k	{0,1}	{0,1}	{0,1}		{0,1}

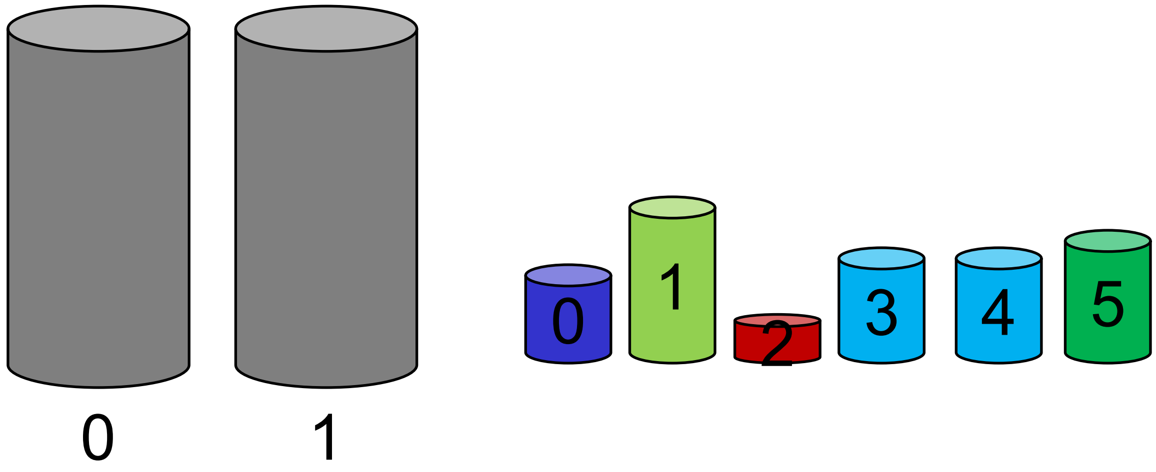
Matrix models

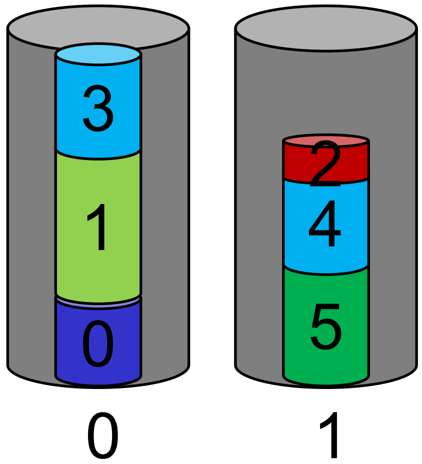
Matrix models are common way to model many challenging problems

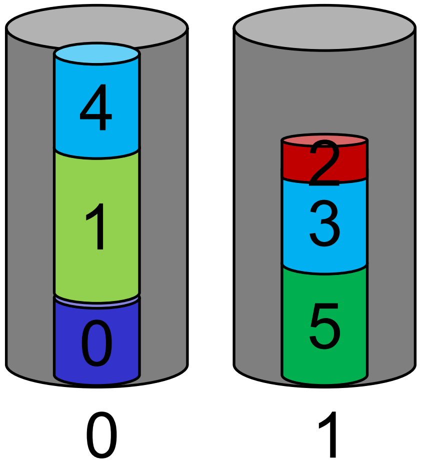
- bin packing
- combinatorial designs
- workforce scheduling
- covering arrays of software test suites
- round-robin sports tournament scheduling
- ...

Depending on the problem, many of the rows (or columns) may represent interchangeable objects

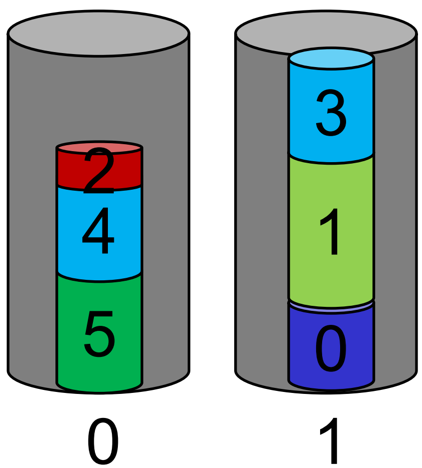
Simple bin example

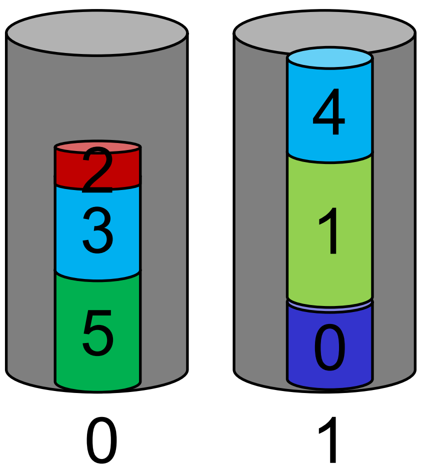


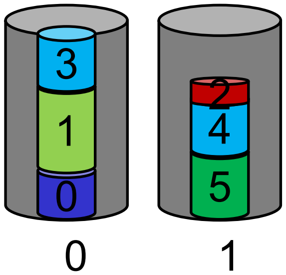




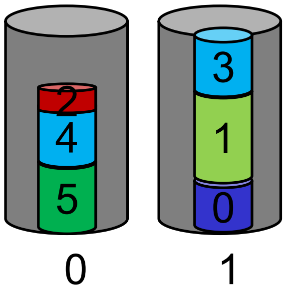
Is there any difference?



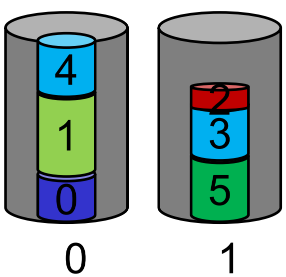




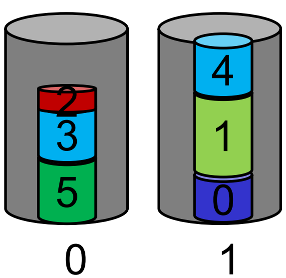
	obj 0	obj 1	obj 2	obj 3	obj 4	obj 5
bin 0	1	1	0	1	0	0
bin 1	0	0	1	0	1	1



	obj 0	obj 1	obj 2	obj 3	obj 4	obj 5
bin 0	0	0	1	0	1	1
bin 1	1	1	0	1	0	0



	obj 0	obj 1	obj 2	obj 3	obj 4	obj 5
bin 0	1	1	0	0	1	0
bin 1	0	0	1	1	0	1



	obj 0	obj 1	obj 2	obj 3	obj 4	obj 5
bin 0	0	0	1	1	0	1
bin 1	1	1	0	0	1	0

The social golfer

32 golfers want to organise weekly foursomes, but where no two golfers play together more than once.

- matrix model, with constraints ...

	wk0	wk1	wk2					...	wk 10)
g0	{0--7}	{0--7}	{0--7}						{0--7}
g1	{0--7}	{0--7}							{0--7}
g2									
...	...								
g32	{0--7}	{0--7}	{0--7}						{0--7}

any two rows are interchangeable

any two columns are interchangeable

32!*10! symmetric solutions by changing rows and columns

Matrix models: symmetry

For any matrix model, we will define two solutions, S_1 and S_2, as being symmetric

if and only if

S2 can be obtained from S1 by a sequence of row swaps and column swaps.

We still need a clear and systematic way of expressing symmetry breaking constraints on matrix models

Lexicographic ordering

We can treat each row of the matrix as a vector of values.
Any two vectors have a *lexicographic* ordering between them.

Lexicographic order ~ Dictionary order

For two vectors (or arrays ...) where the values are from some set with a total order $\lhd$ (i.e. the lexicographic order)

x = x[0], x[1], x[2], ..., x[n-1]
y = y[0], y[1], y[2], ..., y[n-1]

x and y are in lexicographic order ($x \lhd y$)
if and only if,
$\exists i \in \{0,\ldots,n-1\}\ \ \forall j<i\ (x[j] == y[j]\ \ \&\&\ x[i] \lhd y[i])$

Lexicographic ordering is a total order.

Forcing the matrix row sequence to be in lexicographic order breaks all row symmetries.

A	B	E
B	C	D
B	C	E

We can also do the same on the columns

[A,B,E] $\lhd$ [B,C,D]   because A $\lhd$B
[B,C,D] $\lhd$ [B,C,E]   because A=A, B=B, and C$\lhd$E

from now on, I will drop the '$\lhd$' symbol and just use ≤ , <, etc

Breaking both row and column symmetries by lexicographic order is subtle

- once we break the row symmetries, we are distinguishing the columns from one another
- arbitrary constraints on lex orders might not work

Consider a 4×3 matrix of 0/1 variables x_{ij}, with the constraints that $\sum_{i,j} x_{ij} = 7$ and for each pair j≠k $\sum_i x_{ij}.x_{ik} \leq 1$ (i.e. two rows can have at most one column where they both have '1')

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

is a solution

The rows are interchangeable.

The columns are interchangeable.

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

is a lex-ordered solution

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

but lex-ordering these columns breaks the row ordering ...

However:

for any matrix model where the rows are interchangeable and the columns are interchangeable, if there is a solution then there will always be at least one solution where both the rows and columns are lex ordered.

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

... but it is not necessarily unique

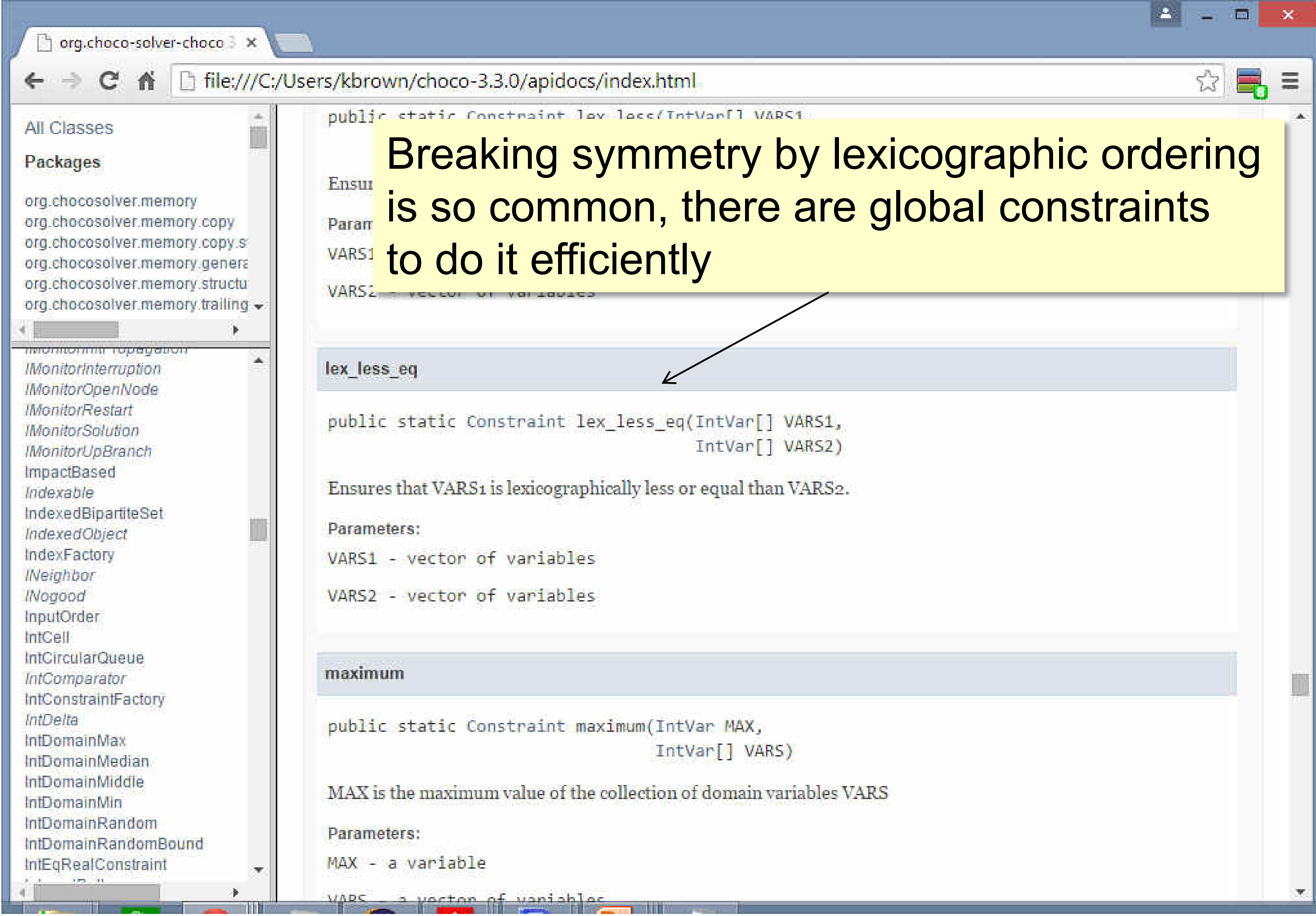
0	0	1
0	1	1
1	0	1
1	1	0

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

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

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

Breaking symmetry by lexicographic ordering is so common, there are global constraints to do it efficiently



org.choco-solver-choco 3

file:///C:/Users/kbrown/choco-3.3.0/apidocs/index.html

All Classes

Packages

org.chocosolver.memory
org.chocosolver.memory.copy
org.chocosolver.memory.copy.s
org.chocosolver.memory.genera
org.chocosolver.memory.structu
org.chocosolver.memory.trailing

IMonitorInterruption
IMonitorOpenNode
IMonitorRestart
IMonitorSolution
IMonitorUpBranch
ImpactBased
Indexable
IndexedBipartiteSet
IndexedObject
IndexFactory
INeighbor
INogood
InputOrder
IntCell
IntCircularQueue
IntComparator
IntConstraintFactory
IntDelta
IntDomainMax
IntDomainMedian
IntDomainMiddle
IntDomainMin
IntDomainRandom
IntDomainRandomBound
IntEqRealConstraint

lex_less_eq

```
public static Constraint lex_less_eq(IntVar[] VARS1,
                                     IntVar[] VARS2)
```

Ensures that VARS1 is lexicographically less or equal than VARS2.

Parameters:

VARS1 - vector of variables

VARS2 - vector of variables

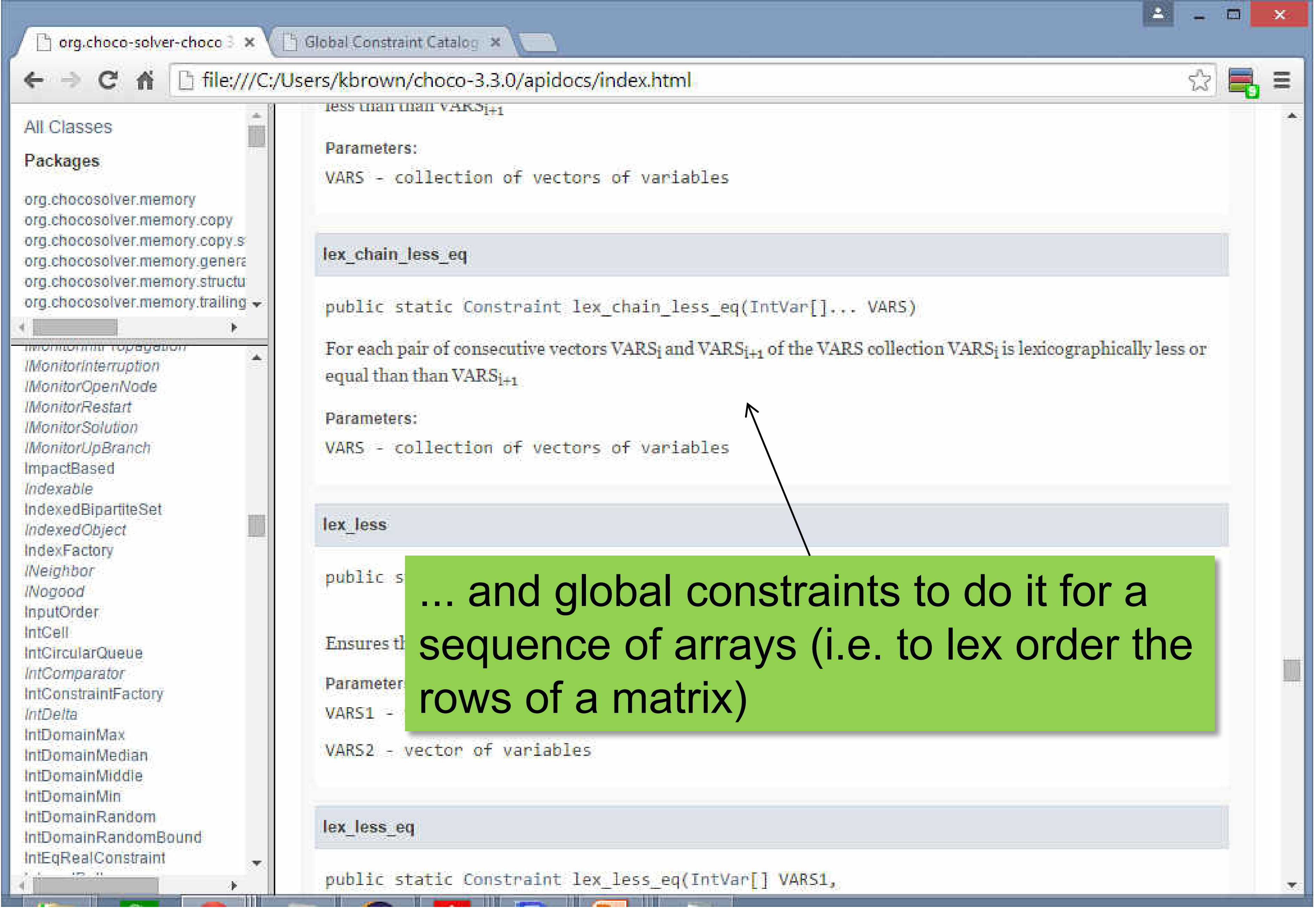
maximum

```
public static Constraint maximum(IntVar MAX,
                                 IntVar[] VARS)
```

MAX is the maximum value of the collection of domain variables VARS

Parameters:

MAX - a variable



file:///C:/Users/kbrown/choco-3.3.0/apidocs/index.html

All Classes

Packages

org.chocosolver.memory
org.chocosolver.memory.copy
org.chocosolver.memory.copy.s
org.chocosolver.memory.genera
org.chocosolver.memory.structu
org.chocosolver.memory.trailing

IMonitorInterruption
IMonitorOpenNode
IMonitorRestart
IMonitorSolution
IMonitorUpBranch
ImpactBased
Indexable
IndexedBipartiteSet
IndexedObject
IndexFactory
INeighbor
INogood
InputOrder
IntCell
IntCircularQueue
IntComparator
IntConstraintFactory
IntDelta
IntDomainMax
IntDomainMedian
IntDomainMiddle
IntDomainMin
IntDomainRandom
IntDomainRandomBound
IntEqRealConstraint

less than than $VARS_{i+1}$

Parameters:

`VARS - collection of vectors of variables`

lex_chain_less_eq

```
public static Constraint lex_chain_less_eq(IntVar[]... VARS)
```

For each pair of consecutive vectors $VARS_i$ and $VARS_{i+1}$ of the VARS collection $VARS_i$ is lexicographically less or equal than than $VARS_{i+1}$

Parameters:

`VARS - collection of vectors of variables`

lex_less

```
public s
```

Ensures th

Parameter

`VARS1 -`

`VARS2 - vector of variables`

lex_less_eq

```
public static Constraint lex_less_eq(IntVar[] VARS1,
```

... and global constraints to do it for a sequence of arrays (i.e. to lex order the rows of a matrix)

Next lecture ...

What happens under the hood ...

Acknowledgments: example taken from slides by Brahim Hnich