Parameterized Complexity in Constraint Programming

Toby Walsh
NICTA and UNSW

Newcastle, March 2010

Parameterized Complexity in Constraint Programming

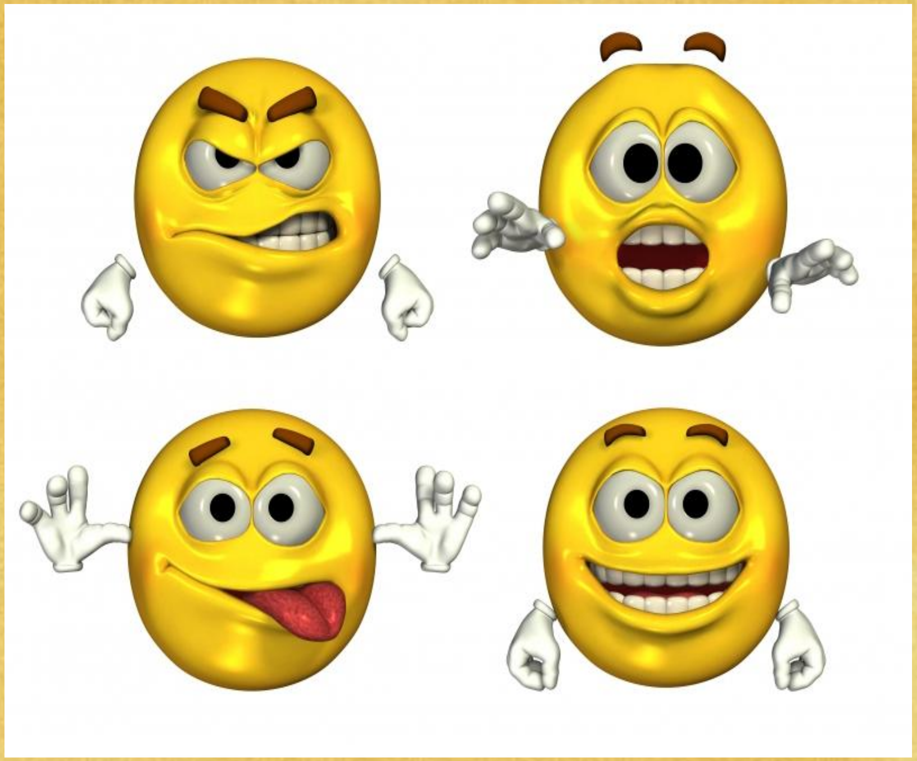
Mostly based on [Bessiere, Hebrard, Hnich, Kiziltan, Quimper, Walsh, AAAI-08]

Parameterized Complexity in Constraint Programming

End of talk includes work with George Katsirelos and Nina Narodytska

Motivation

- Beyond the NP-hardness of CP
  - Parameterized complexity
- Some common themes
  - Tree width
  - Cutsets & backdoors
  - Dynamic programming



Constraint Programming 101

9			1					5
		5		9		2		1
8				4				
				8				
			7					
				2	6			9
2			3					6
			2			9		
		1	9		4	5	7	

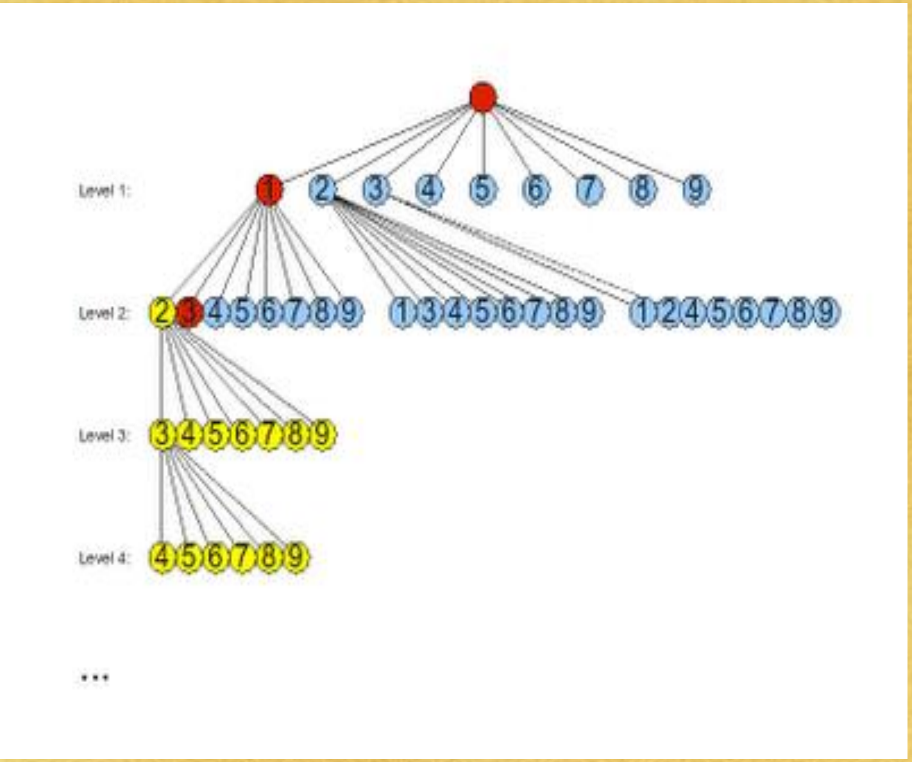
- Variables
  - Each with a finite domain of values
- Constraints
  - Allowed tuples of values

Constraint Programming 101

9			1					5
		5		9		2		1
8				4				
				8				
			7					
				2	6			9
2			3					6
			2			9		
		1	9		4	5	7	

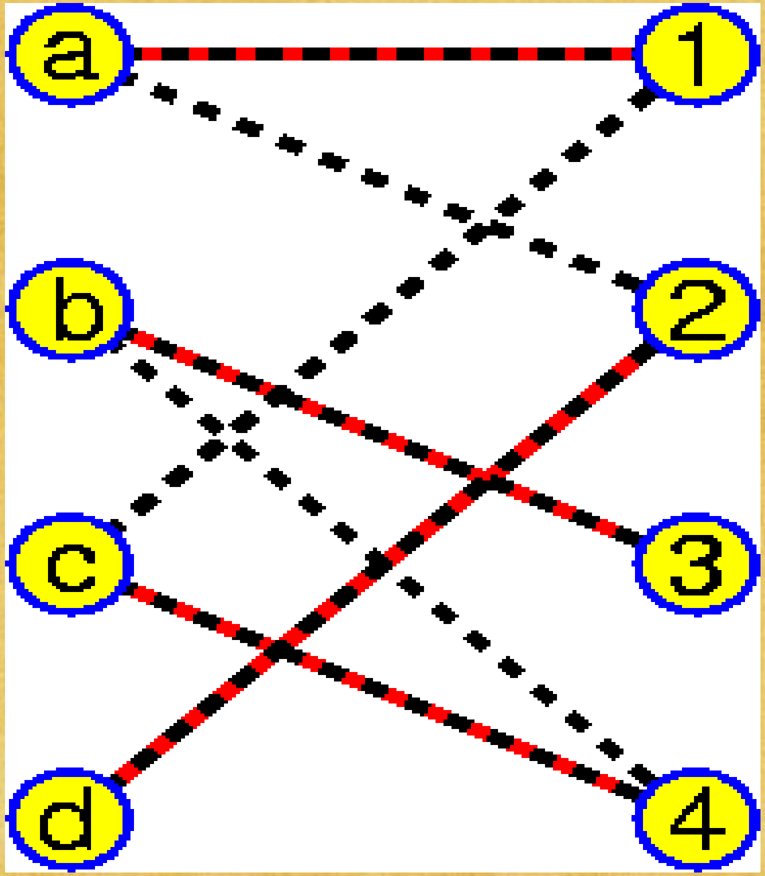
- Variables
  - Xij ∈ {1,...,9}
- Constraints
  - AllDiff(X11,..,X19), ..
  - AllDiff(X11,..,X91), ..
  - AllDiff(X11,..,X33), ..

Constraint Programming 101



- Backtracking search
  - Try X11=1
  - Else X11=2
  - ..
- Propagation
  - If X11=1 then X12≠1
  - If

Constraint Programming 101



- Global constraints
  - Capture common patterns
  - Efficient propagation algorithms
  - E.g. AllDiff

Parameterized complexity

- Useful insight into (in)tractability of CP
- Some common themes
  - Tree width
  - Backdoors/cutsets
  - Dynamic programming

Parameterized complexity

- Useful insight into (in)tractability of CP
- Some common themes
  - Tree width
  - Backdoors/cutsets
  - Dynamic programming

Practical algorithms?

Parameterized complexity

- Useful insight into (in)tractability of CP
- Some common themes
  - Tree width ☺
  - Backdoors/cutsets
  - Dynamic programming

Practical algorithms?

Parameterized complexity

- Useful insight into (in)tractability of CP
- Some common themes
  - Tree width ☺
  - Backdoors/cutsets ☹
  - Dynamic programming

Practical algorithms?

Parameterized complexity

- Useful insight into (in)tractability of CP
- Some common themes
  - Tree width ☺
  - Backdoors/cutsets ☹
  - Dynamic programming ☺

Practical algorithms?

Parameterized complexity & CP

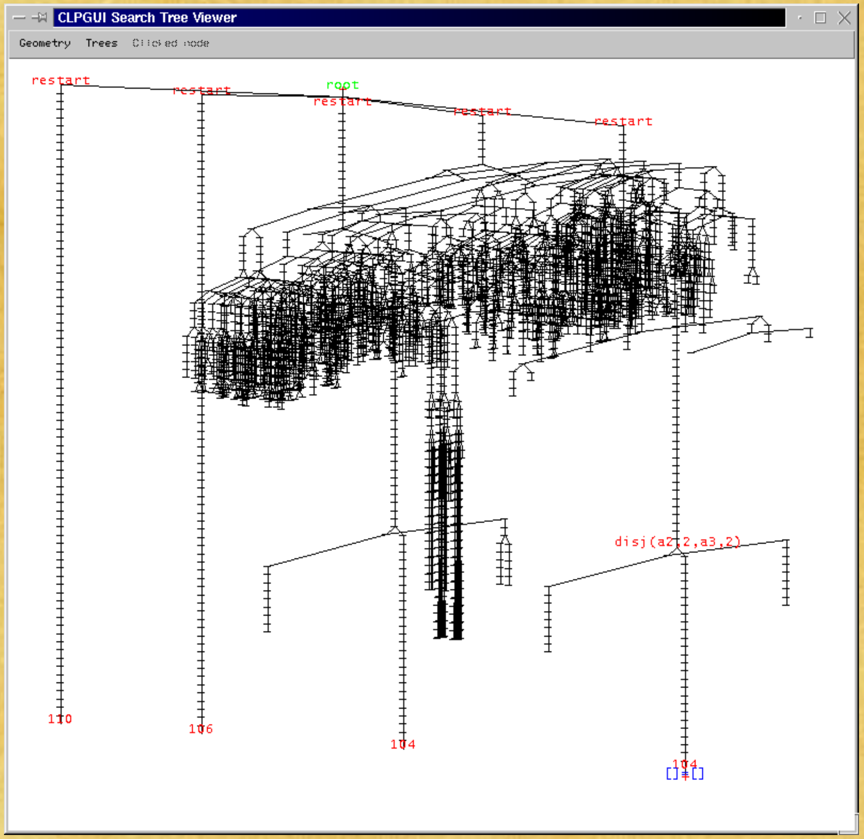
Complexity **of** searching for a solution

Exponential sized search tree

Complexity **within** the search for a solution

Propagation at each node of this tree

Complexity of search



- Finding solutions
  - Or proving unsatisfiability
- Consider the whole search tree!

k-consistency

- Arc-consistency (k=2)
  - If we assign any value to any variable, we can find a satisfying assignment for any other variable
  - E.g. $X>Y$, dom(X)=dom(Y)={1,2} then enforcing AC sets X=2, Y=1

k-consistency

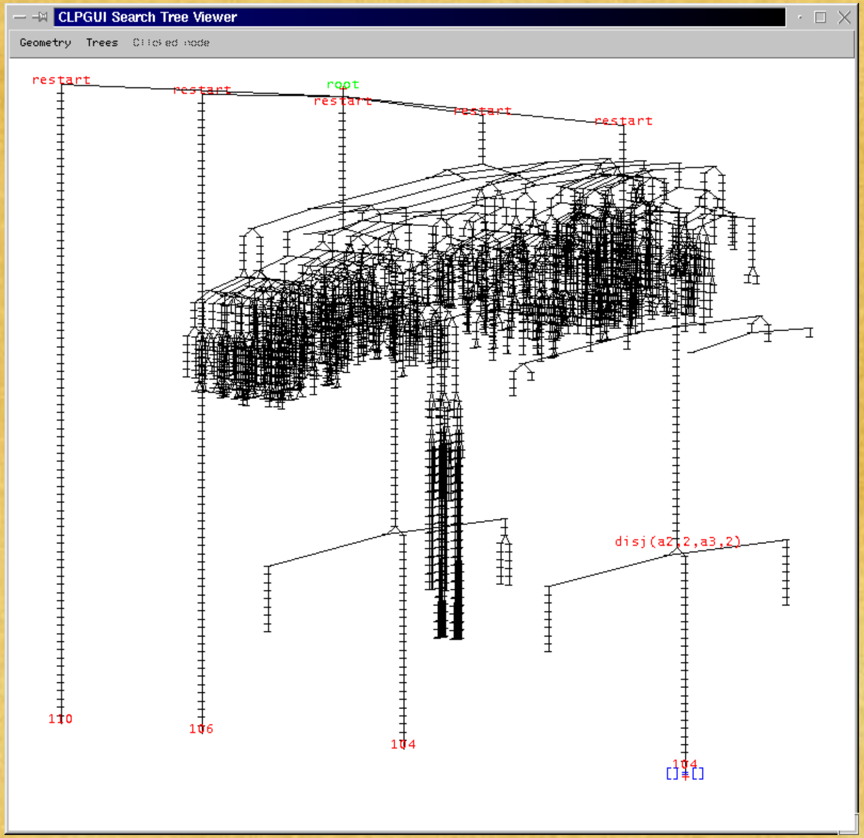
- Arc-consistency (k=2)
  - If we assign any value to any variable, we can find a satisfying assignment for any other variable
  - Takes $O(ed^2)$ time to enforce
  - If constraint graph is acyclic then enforcing AC, we can find solutions backtrack free

k-consistency

- k-consistency ($k \geq 2$)
  - If we assign any values to any k-1 variables, we can find a satisfying assignment for any other variable
  - Takes $O(d^k)$ time to enforce
  - If constraint graph has tree width $\leq k$ then enforcing k-consistency, we can find solutions backtrack free [Freuder 82]

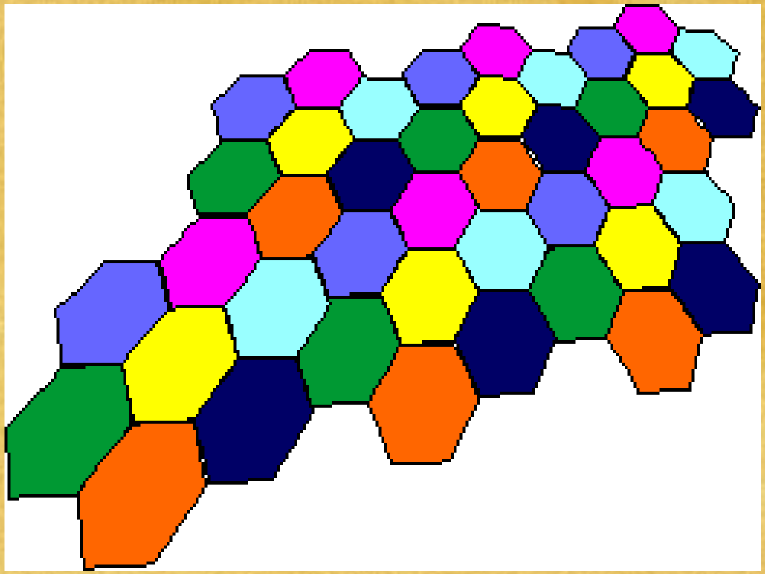
Complexity within search

- Global constraints are often on limits of tractability
- At each node of search tree
  - Propagation may be very costly



NValues(X1,..,Xn,N)

- Values often represent resources
  - E.g. frequency assignment
- NValues satisfied iff
  - |{Xi | 1≤i≤n}|=N
  - Generalization of AllDifferent



NValues(X1,..,Xn,N)

- Propagation is NP-hard
  - Finding a satisfying assignment
  - Reduction from 3SAT
  - $X_i \in \{z_i, -z_i\}$ for $1 \leq i \leq n$
  - $X_{n+j} \in \{z_a, -z_b, z_c\}$ if jth clause is

    z_a or $-z_b$ or z_c
  - $N=n$

NValues(X1,..,Xn,N)

- Consider two different parameters
  - k = |$\bigcup$dom(Xi)| then fixed parameter tractable
  - k = max(dom(N)) then W[2]-hard

NValues(X1,..,Xn,N)

- k = |⋃dom(Xi)|
  - Define automaton that accepts only those sequences X1,..,Xn,N that satisfy NValues
  - States of automaton are sets of values used so far
  - Global REGULAR constraint propagates this in $O(ndQ)=O(nd2^k)$ time using dynamic programming

NValues(X1,..,Xn,N)

- k = max(dom(N)) then W[2]-hard
  - Hitting set is smallest set that *hits* every set in a collection
  - Hitting set is W[2]-complete in size of hitting set
  - Immediate reduction

Backdoors

- Strong backdoor
  - Subset of variables which give a polynomial subproblem
  - Correlated with problem hardness [Williams, Gomes, Selman IJCAI-03] [Kilby, Slaney, Thiebaux, Walsh, AAAI-05]

Backdoors

- Many of our *fixed-parameter tractability* results for global constraints exploit
  - **Cycle cutsets** that are **backdoors** into an acyclic (and thus polynomial) subproblem
  - Once cycle cutset is instantiated, we can use 2-consistency (aka arc-consistency) to solve problem

Disjoint([X1,..,Xn],[Y1,..,Ym])

- Xi≠Yj for any i, j
- Useful in scheduling and time-tabling problems
- NP-hard to propagate
  - Simple reduction (homework exercise :-)

Disjoint([X1,..,Xn],[Y1,..,Ym])

- Fixed-parameter tractable in

  $k=|\bigcup dom(Xi) \cap \bigcup dom(Yj)|$

- Try all 2^k possible subsets for

  $\bigcup Xi \cap \bigcup dom(Yj)$

  - Each is a cycle cutset
  - Remaining Xi and Yj are disjoint

Disjoint([X1,..,Xn],[Y1,..,Ym])

- Fixed-parameter tractable in

  $k=|\bigcup dom(Xi) \cap \bigcup dom(Yj)|$

Disjoint([X1,..,Xn],[Y1,..,Ym])

- Fixed-parameter tractable in

  $k=|\bigcup dom(Xi) \cap \bigcup dom(Yj)|$

  $k=|\bigcup dom(Xi) \cup \bigcup dom(Yj)|$

Other FPT results

- Uses([X1,..,Xn],[Y1,..,Ym])
  - $\bigcup \mathrm{dom}(Xi) \subseteq \bigcup \mathrm{dom}(Yj)$
  - $k=|\bigcup \mathrm{dom}(Yj)|$
- Among([X1,..,Xn],S,N)
  - $N=|\{i \mid Xi \in S\}|$
  - $k=|ub(S) \setminus lb(S)|$

Other FPT results

- ROOTS([X1,..,Xn],S,T)
  - S={i | Xi ∈T}
  - Used to encode Among, AtMost, AtLeast, Uses, Domain, ...
  - k=|ub(T) \ lb(T)|
- CardPath([X1,..,Xn],C,N)
  - N = |{i | C(Xi,..,Xi+p)}|
  - Used to encode Regular, Sequence, Contiguity, Lex

FPT results in symmetry

- Lex Leader
  - General symmetry breaking method
- Double Lex
  - Specialized method for row and column symmetry

Symmetry

- All interval series
  - X1,..,X11 = 3,7,4,6,5,0,10,1,9,2,8

    4 3 2 1 5 10 9 8 7 6
  - Problem from musical composition

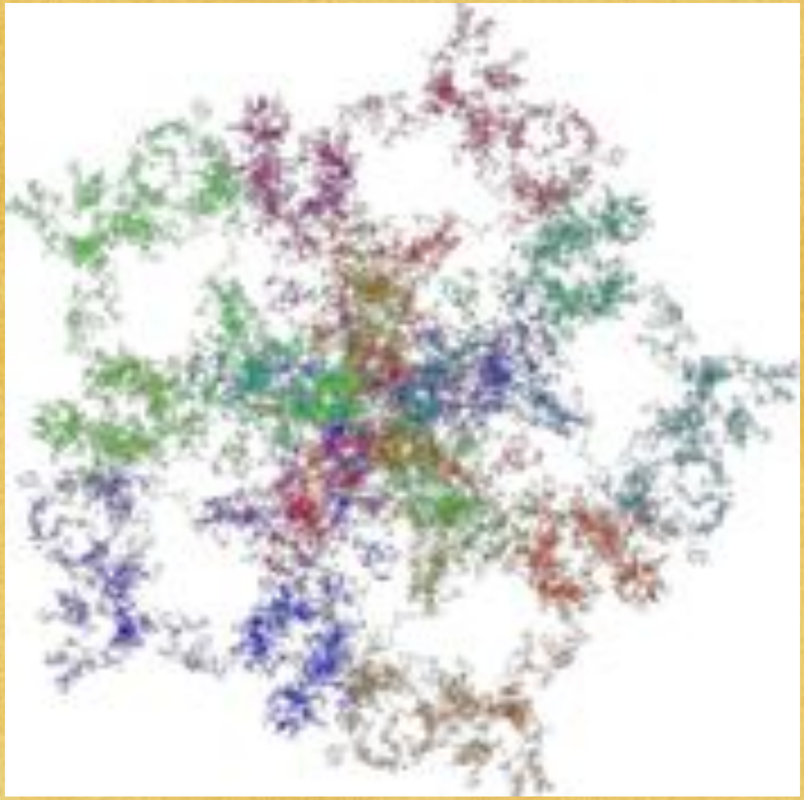
Symmetry

- All interval series
  - $X_1,..,X_{11} = 3,7,4,6,5,0,10,1,9,2,8$
  - Value symmetry: $i \mapsto 10-i$
  - $X_1,..,X_{11} = 7,3,6,4,5,10,0,9,1,8,2$

Symmetry

- All interval series
  - X1,..,X11 = 3,7,4,6,5,0,10,1,9,2,8
  - Variable symmetry: Xi ↦ X12-i
  - X1,..,X11 = 8,2,9,1,10,0,5,6,4,7,3

Symmetry constraints



- Eliminate *symmetric* solutions
  - If we want such solutions, easy to generate
  - This reduces search exponentially in best case
    - Both

Symmetry constraints

- All interval series
- Value symmetry
  - $X1,..,X11 \leq_{lex} 10\text{-}X1,..,10\text{-}X11$
- Variable symmetry
  - $X1,..,X11 \leq_{lex} X11,..,X1$

Lex Leader

- General method for any group of symmetries Σ
  - Leave just the smallest (in a lex ordering) in each symmetry class
  - $X1,..,Xn \leq_{lex} \sigma(X1),..,\sigma(Xn)$ for all $\sigma \in \Sigma$

Lex Leader

- General method for any group of symmetries Σ
  - $X1,..,Xn \leq_{lex} \sigma(X1),..,\sigma(Xn)$ for all $\sigma \in \Sigma$
  - E.g. inversion value symmetry that maps *a* onto d-*a*
  - $X1,..,Xn \leq_{lex} d\text{-}X1,..,d\text{-}Xn$

Lex Leader

- General method for any group of symmetries Σ
  - Let LexLeader(Σ,X1,..,Xn) hold iff X1,..,Xn $\leq_{lex}$ σ(X1),..,σ(Xn) for all σ∈Σ
  - Propagating LexLeader is NP-hard

Lex Leader

- General method for any group of symmetries Σ
  - Let LexLeader(Σ,X1,..,Xn) hold iff X1,..,Xn $\leq_{lex}$ σ(X1),..,σ(Xn) for all σ∈Σ
  - Propagating LexLeader is NP-hard
  - Fixed-parameter tractable in k=|Σ|

Lex Leader

- General method for any group of symmetries Σ
  - Let LexLeader(Σ,X1,..,Xn) hold iff X1,..,Xn $\leq_{lex}$ σ(X1),..,σ(Xn) for all $\sigma \in \Sigma$
  - Propagating LexLeader is NP-hard
  - Fixed-parameter tractable in $k=|\Sigma|$
    - Actually number of generators of Σ!

Lex Leader

- General method for any group of symmetries Σ
  - Let LexLeader(Σ,X1,..,Xn) hold iff X1,..,Xn $\leq_{lex}$ σ(X1),..,σ(Xn) for all σ∈Σ
  - Define automaton that accepts only sequences X1 to Xn that satisfy lex inequalities
  - States are which subset of inequalities have been satisfied so far, 2^k such states

Row & Col Symmetry

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}.$$

- Matrix model
  - Arrays of decision variables
- Row and col symmetry
  - Interchangeable rows & columns

Double Lex

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}.$$

- DoubleLex
  - Lex ordering rows (breaks all row symmetry)
  - Lex ordering cols (breaks all col symmetry)
  - But leaves some row/col symmetries

Double Lex

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}.$$

- DoubleLex
  - Lex order rows
  - Lex order cols
- Eliminates most symmetry
  - NP-hard to check if all symmetry is eliminated

Double Lex

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}.$$

- DoubleLex
  - Lex order rows
  - Lex order cols
- Propagating DoubleLex is NP-hard

Double Lex

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}.$$

- DoubleLex
  - Lex order rows
  - Lex order cols
- Propagating DoubleLex is NP-hard
  - Took 10 years to show this!

Double Lex

- DoubleLex
  - Lex order rows
  - Lex order cols
- Does not eliminate all row & col symmetry
  - Look in the shadow!

001		001
010	and	011
110		100

Row & Col Symmetry

- Breaking all row & col symmetry
  - NP-hard in general
  - Fixed parameter tractable in k=min(#rows,#cols)
  - Simple observation:

All row symmetry is eliminated by lex ordering rows

Consider all 2^k possible permutations of columns where k=#cols

Conclusions

- Parameterized complexity gives useful new insight into
  - Complexity of search in CP
  - Complexity of propagation in CP

Conclusions

- Parameterized complexity gives useful new insight into
  - Complexity of search in CP
  - Complexity of propagation in CP
- Some common themes
  - Tree width
  - Cycle cutsets & backdoors
  - Dynamic programming & automata