Symmetry
and the
Generation of Constraint Models

Alan M. Frisch

Artificial Intelligence Group

Department of Computer Science

University of York

Collaborators

Christopher Jefferson Bernadette Martinez Hernandez Ian Miguel

Constraint Programming

Constraint Programming (CP) is useful for solving a wide range of important, complex problems

- scheduling, allocation, layout, configuration, …

Def (Narrow view): Solve combinatorial (optimisation) problem by

- **Model** it by mapping it to a finite-domain constraint satisfaction problem
- **Solve** the constraint satisfaction problem
- **Map** the solution back to original problem

Symmetry

- Models often contain an enormous number of symmetries.
- Symmetry in a model yields symmetry in the space that is searched for a solution, which leads to inefficiency.
- Once symmetries in the model are identified, there are a variety of methods for removing it from the model or from the search.
- **How do we identify them?**

Conclusion

- Symmetries enter a model from two sources:
  - inherent in the problem
  - introduced by the modelling process
- The modelling process can be formalised and automated [IJCAI-05].
- A formal/automated account of modelling should provide a formal/auotmated account of the symmetries introduced by modelling.

The Plan

- Constraint satisfaction problems and their symmetries
- An introduction to modelling (by example)
- The automation of modelling
- Automatic identification of symmetry introduced by modelling

Part 1

Constraint Satisfaction Problems and their Symmetries

What is the (finite domain) CSP?

An instance comprises:

- Finite set of variables
- Each associated with a finite domain
- Finite set of constraints on the values taken by the variables
- (Objective function)

Solution is an assignment of values to variables that satisfies the constraints (and optimises objective function)

Example of a CSP

Problem: Find three distinct digits that sum to 4.

X, Y, Z in $\{0,1,2,3,4,5,6,7,8,9\}$

$X \neq Y, \quad Y \neq Z, \quad X \neq Z$

$X + Y + Z = 4$

A Solution: $\{X \rightarrow 0, Y \rightarrow 1, Z \rightarrow 3\}$

What Is Symmetry?

- A symmetry is a total bijective function from total instantiations to total instantiations which maps every solution to a solution.
- Usually consider a set of symmetries that is closed under inverse and composition and to contain identity function, i.e. a group.
- This induces a partitioning on the total instantiations. Each partition is called a symmetry class. Every member is a solution or no member is.
- In general, other objects can be mapped and other properties preserved.

Value Symmetry

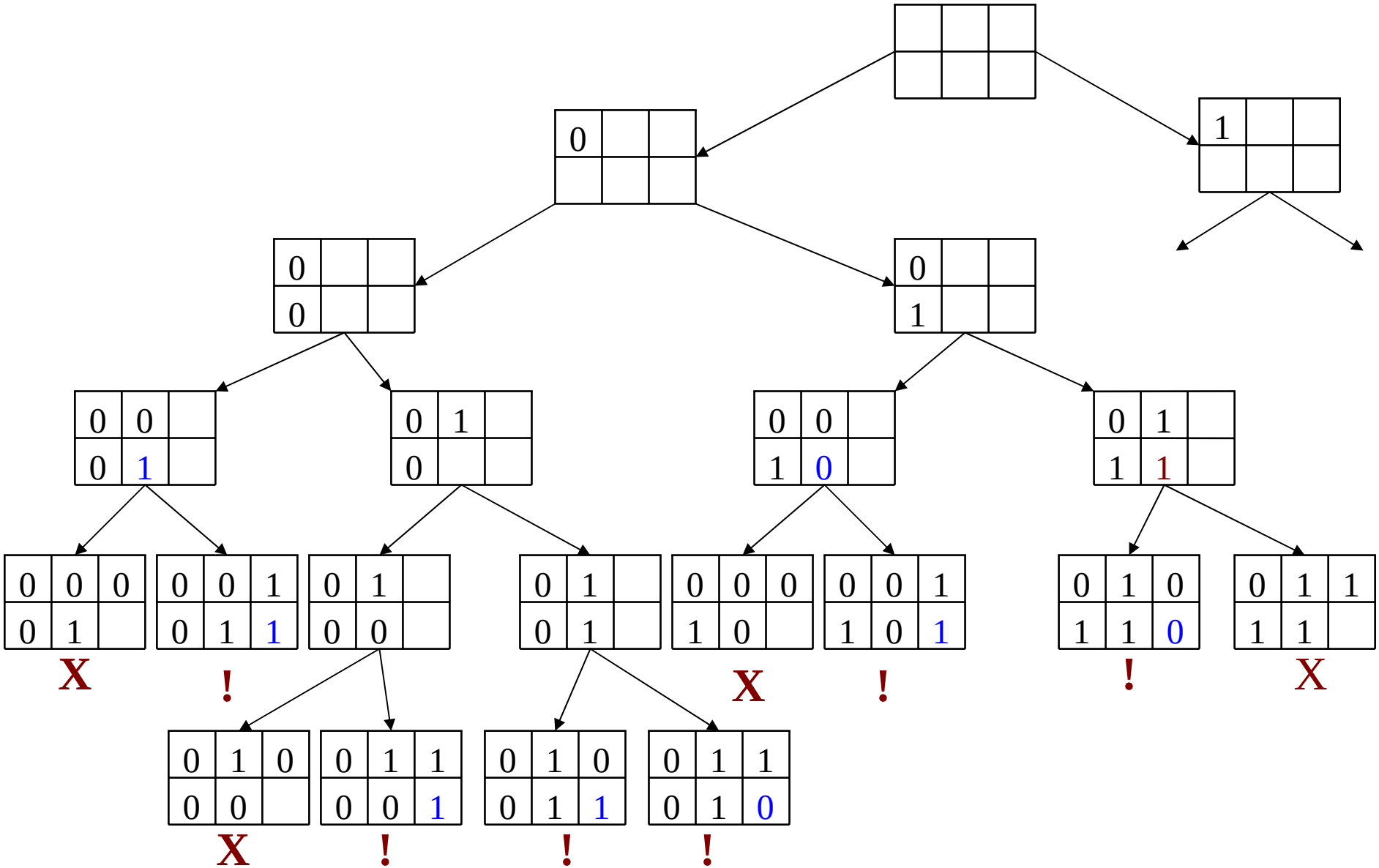
- A group of symmetries on values that induces a group of symmetries on instantiations.
- Example: “Assign r, b or g to each node of a graph so that no arc connects two nodes of the same colour” has value symmetry.
  $S = \{r \rightarrow b, b \rightarrow g, g \rightarrow r\}$.
- Induces a symmetry on instantiations that maps
  $\{N1 \rightarrow r, N2 \rightarrow b, N3 \rightarrow g\}$ to
  $\{N1 \rightarrow S(r), N2 \rightarrow S(b), N3 \rightarrow S(g)\} =$
  $\{N1 \rightarrow b, N2 \rightarrow g, N3 \rightarrow r\}$

Variable Symmetry

- A group of symmetries on variables that induces a group of symmetries on instantiations.
- Example: “Assign 0,…,9 to X, Y, Z such that X, Y, Z are all different and sum to 4” has variable symmetry.
  $S = \{X \rightarrow Y,\ Y \rightarrow Z,\ Z \rightarrow X\}$.
- Induces a symmetry on instantiations that maps
  $\{X \rightarrow 0,\ Y \rightarrow 1,\ Z \rightarrow 3\}$ to
  $\{S(X) \rightarrow 0,\ S(Y) \rightarrow 1,\ S(Z) \rightarrow 2\} =$
  $\{Y \rightarrow 0,\ Z \rightarrow 1,\ X \rightarrow 2\}$

Partial Instantiation Search

(Forward Checking)



How to Break Symmetries

- Be clever during search (SBDS, SBDD, …)
- Reformulate the problem
- Add symmetry breaking constraints to problem formulation.
  - Consistent: At **least** one instantiation in every symmetry class satisfies the constraints.
  - Complete: At **most** one instantiation in every symmetry class satisfies the constraints.
  - Enforcing these constraints during search prunes paths---solution paths **and failure paths.**

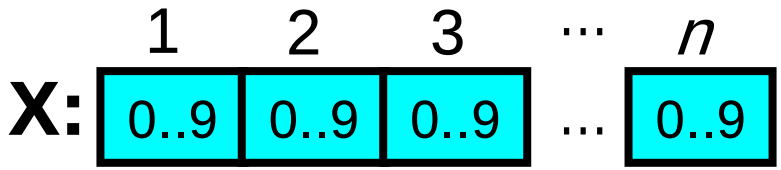
Part 2

An Introduction to Modelling (by example)

Model using Explicit Representation

Given *n* and *s*,

Find a set of *n* digits that sum to *s*.



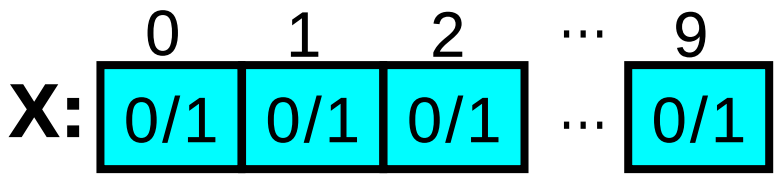
given	*n*:int, *s*:int
find	X: matrix (indexed by 1..*n*) of int (0..9)
such that	AllDiff(X)
	Sum(X) = *s*

Symmetries: permutations of the index values

Model using Occurrence Representation

Given *n* and *s*,

Find a set of *n* digits that sum to *s*.



given *n*:int, *s*:int

find X: matrix (indexed by 0..9) of int (0..1)

such that Sum(X) = *n*

$\sum_{i \in 0..9} i \bullet X[i] = s$

Symmetries: none

The SONET Problem

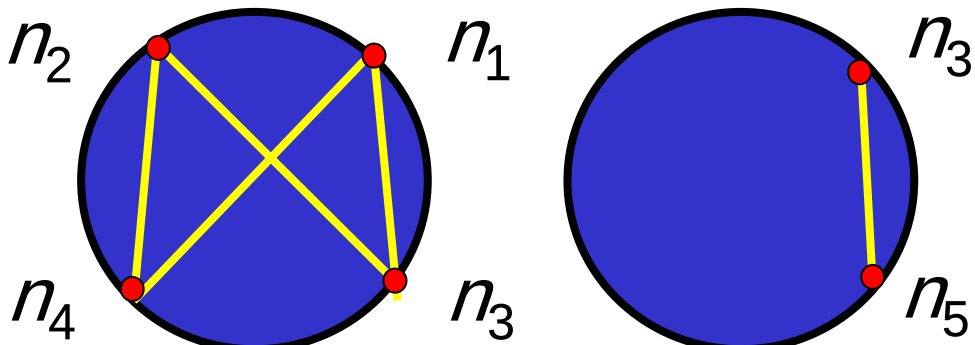
Specification

- Given ***nrings*** rings, ***nnodes*** nodes, a set of pairs of nodes (communication ***demand***) and an integer ***capacity*** (of each ring). Install nodes on rings satisfying demand and capacity constraints. Minimise installations.

Instance

- *nrings*=2, nnodes=5, capacity = 4
- demand: n_1 & n_3, n_1 & n_4, n_2 & n_3, n_2 & n_4, n_3 & n_5

Solution



Model of the SONET Problem

Rings

Nodes

0..1	0..1	0..1	0..1
0..1	0..1	0..1	0..1
0..1	0..1	0..1	0..1
0..1	0..1	0..1	0..1
0..1	0..1	0..1	0..1
0..1	0..1	0..1	0..1

Symmetries:

- Permute the 2nd index (columns)

The Social Golfers Problem

A number of golfers play a round of golf in each of *nweeks* weeks. Each week they are divided into *ngroups*, groups, each of which plays together. To maximise socialisation, no two golfers can play together twice. How should the golfers be divided each week?

	groups		
weeks	[1,2,3]	[4,5,6]	[7,8,9]
	[1,4,7]	[2,5,8]	[3,6,9]
	[1,5,9]	[2,6,7]	[3,4,8]

Model Social Golfers Problem

groups

weeks			
	[1,2,3]	[4,5,6]	[7,8,9]
	[1,4,7]	[2,5,8]	[3,6,9]
	[1,5,9]	[2,6,7]	[3,4,8]

Symmetries

- Permutations of the values
- Permutations of the first index (rows)
- Permutations of the second index (columns)
- Permutation of the third index

Model Social Golfers Problem

weeks \ groups			
	[1,2,3]	[4,5,6]	[7,8,9]
	[1,4,7]	[2,5,8]	[3,6,9]
	[1,5,9]	[2,6,7]	[3,4,8]

Symmetries

- Permutations of the values
- Permutations of the first index (rows)
- Within each row, permutations of the 2nd index (cols)
- Within each cell, permutations of the 3rd index

Part 3

Automated Modelling

The Modelling Bottleneck

- Modelling a problem as a constraint program requires moderate/great expertise.
  - The model chosen has a substantial effect on efficiency of solving
- Major barrier to widespread use of Constraint Programming.

Reducing the Modelling Bottleneck

- Systematise the knowledge of the expert.
  - ~5 years: We have been building models more and more systematically
- Embed this knowledge in a “compiler” that can refine a high-level problem specification into a set of constraints that can be executed efficiently using existing toolkits.

CONJURE
Automated Model Generation

- Given an abstract formal problem specification, in ESSENCE
- Generates a set of correct models
  - Should include good models generated by experts
  - These models are in ESSENCE', a subset of ESSENCE similar to what is provided by existing constraint toolkits

– ESSENCE must enable specification at a level of abstraction above that at which modelling decisions are made

ESSENCE

- The language enables problems to be specified at a level of abstraction above that at which modelling decisions are made.
  - Allow problems to be specified without making modelling decisions
- This requires features not found in current constraint programming languages.

SONET Specification

given n:int, m:int, c:int
$demand$: set of set $_{(\text{size } 2)}$ of int (1..m)

find $rings$: mset $_{(\text{size } n)}$ of set $_{(\text{maxsize } c)}$ of int (1..m)

minimising $\sum_{r \in rings} . |r|$

such that $\forall pair \in demand \,.\, \exists r \in rings \,.\, pair \subseteq r$

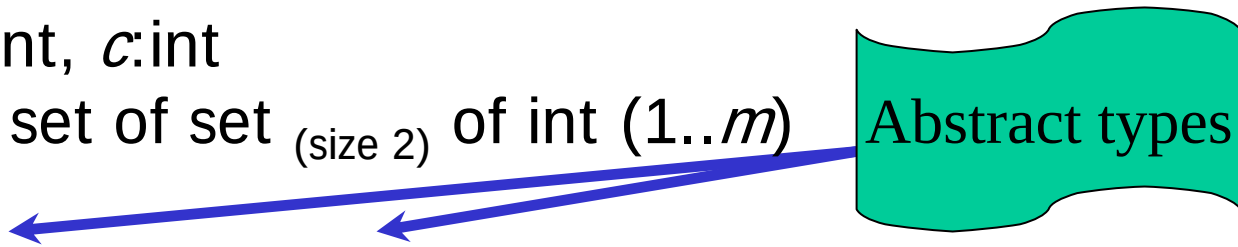
ESSENCE Provides Abstract Types

Sets, Multisets, Partitions, Unnamed Types

Functions, Relations, Tuples

given *n*:int, *m*:int, *c*:int

demand: set of set (size 2) of int (1..*m*)

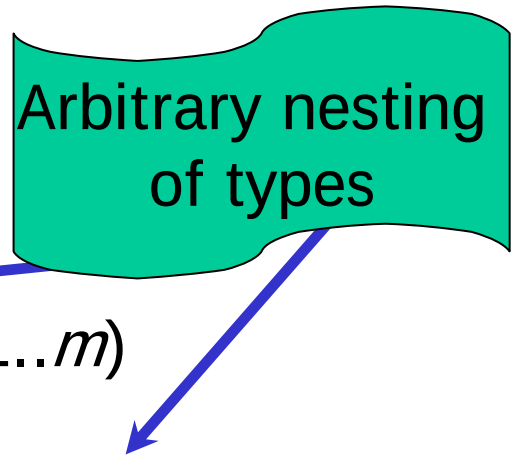


find *rings*: mset (size *n*) of set (maxsize *c*) of int (1..*m*)

minimising $\sum_{r \in rings} .|r|$

such that $\forall pair \in demand . \exists r \in rings . pair \subseteq r$

ESSENCE Supports Arbitrarily-Nested Types



given *n*:int, *m*:int, *c*:int
demand: set of set (size 2) of int (1..*m*)

find *rings*: mset (size *n*) of set (maxsize *c*) of int (1..*m*)

minimising $\sum_{r \in rings} .|r|$

such that $\forall pair \in demand . \exists r \in rings . pair \subseteq r$

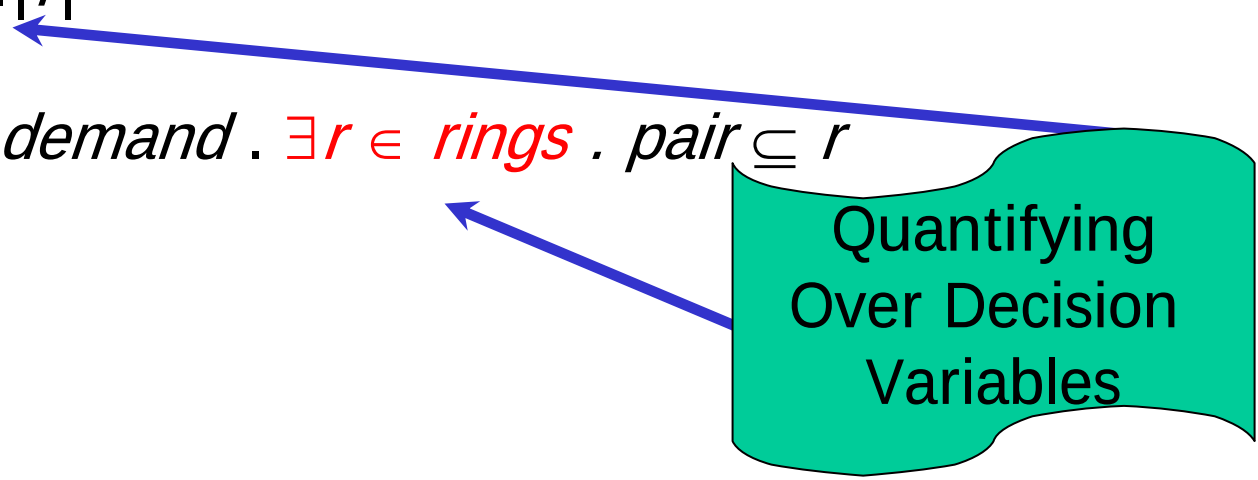
ESSENCE Supports Quantification over Decision Variables

given *n*:int, *m*:int, *c*:int
demand: set of set (size 2) of int (1..*m*)

find *rings*: mset (size *n*) of set (maxsize *c*) of int (1..*m*)

minimising $\sum_{r \in rings} .|r|$

such that $\forall pair \in demand . \exists r \in rings . pair \subseteq r$

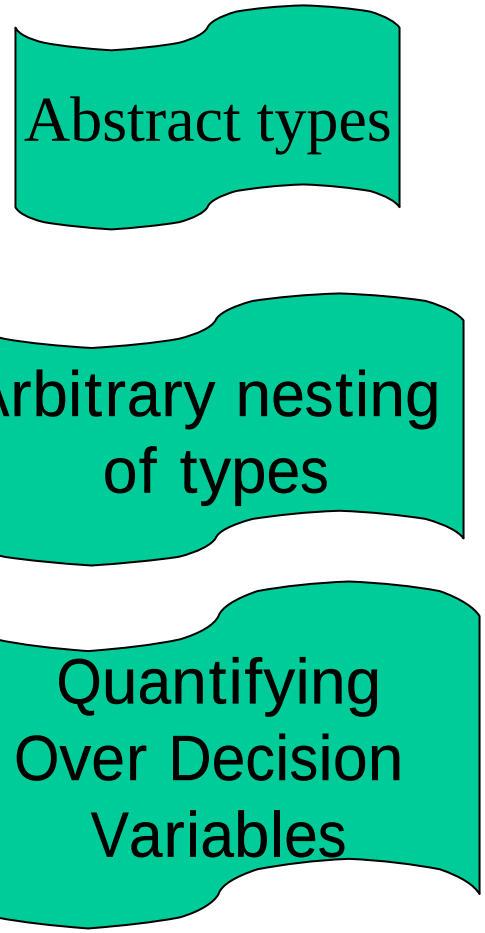


How Usable is Essence?

- Specifications of ~50 problems found in the CSP literature written by an undergraduate with **no** background in constraint programming.

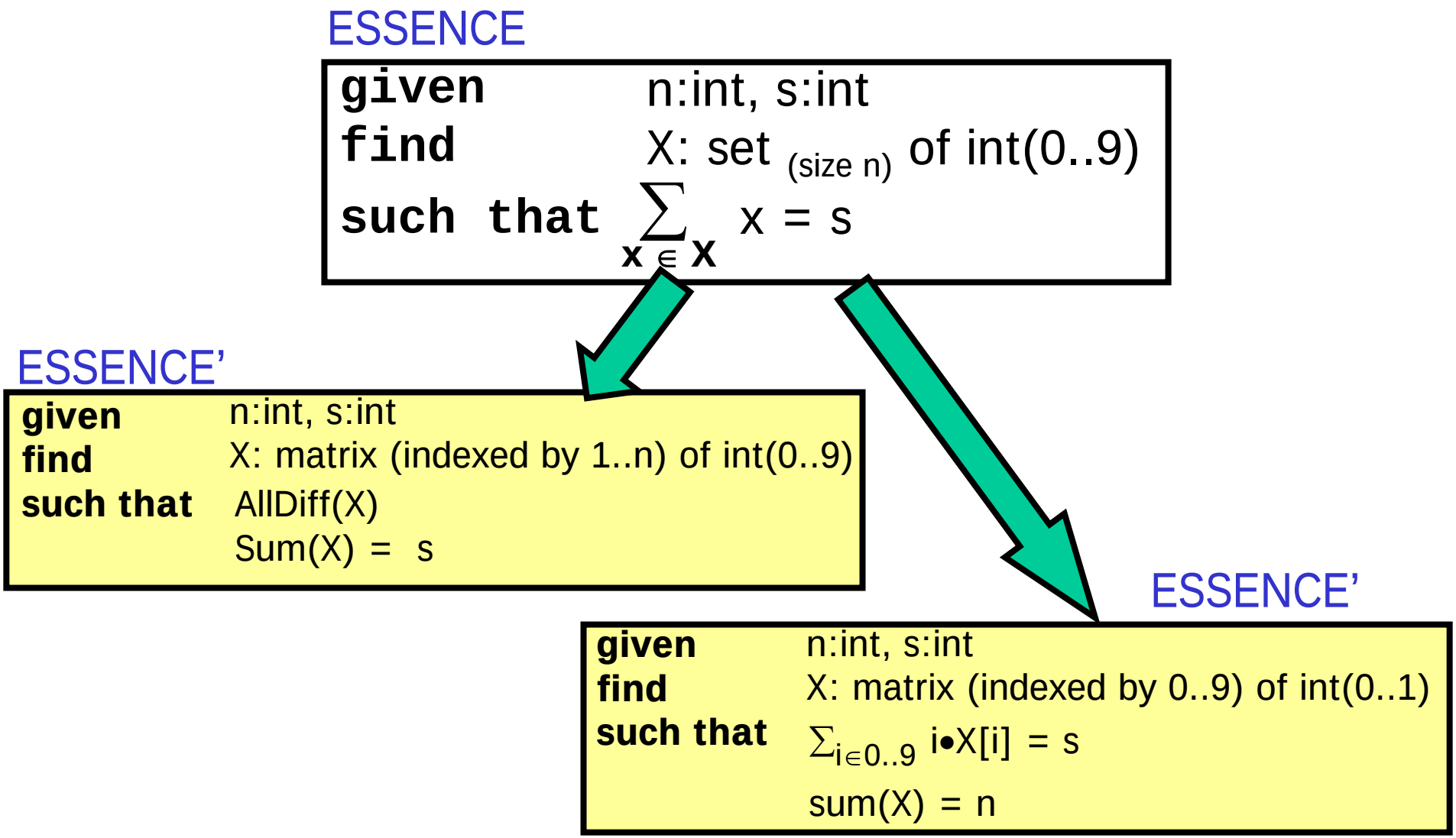
- URL: http://www.cs.york.ac.uk/aig/constraints/

ESSENCE'



- ESSENCE' has a level of abstraction similar to existing constraint languages (OPL, Solver, ECL^{i}PS^{e},…)

Formalisation of the Modelling Problem



Part 4

Automated Identification of Symmetries Introduced by Modelling

CONJURE Identifies Symmetries

- CONJURE annotates the models it produces with the symmetries it introduces.
- As a model is generated, each refinement rule generates appropriate annotations

Refinement of the SONET Problem

Refine: rings:mset (size nrings) of
set (size capacity) of Nodes

Refine	**mset (size n) of τ**	**Explicit**
To	a matrix (indexed by1..n) of refine(τ)	
Sym	permutations of the index values of the matrix symmetries introduced by refine(τ)	

Refine	**set (size n) of τ**	**Occurrence**
To	a matrix (indexed by τ) of bool	
Sym	none	

provided τ is bool, finite set of int, enumerated type

Model of the SONET Problem

Rings

Nodes

0..1	0..1	0..1	0..1
0..1	0..1	0..1	0..1
0..1	0..1	0..1	0..1
0..1	0..1	0..1	0..1
0..1	0..1	0..1	0..1
0..1	0..1	0..1	0..1

Symmetries:

- Permute the 2nd index (columns)

Refinement of the Golfers Problem

Refine: schedule: mset (size nweeks) of
regpart (size ngroups) of golfers

Refine **mset (size n) of τ** **Explicit**
To a matrix (indexed by1..n) of refine(τ)
Sym permutations of the index values of the matrix
each element of the matrix independently has
every symmetry of refine**(τ)**

Refine **regpart (size n) of τ**
To a matrix (indexed by 1..n, 1..|tau| / n) of refine(τ)
Sym permutations of the values of the 1st index
within each row, permutations of values of 2nd index

Refine **newtype (size n)**
To 1..n

Model Social Golfers Problem

weeks \ groups			
	[1,2,3]	[4,5,6]	[7,8,9]
	[1,4,7]	[2,5,8]	[3,6,9]
	[1,5,9]	[2,6,7]	[3,4,8]

Symmetries

- Permutations of the values
- Permutations of the first index (rows)
- Within each row, permutations of the 2nd index (cols)
- Within each cell, permutations of the 3rd index

Conclusion

- Symmetry enters a model from two sources:
  - It is inherent in the problem
  - It is introduced by the modelling process
- The modelling process can be formalised and automated [IJCAI-05].
- A formal/automated account of modelling should provide a formal/automated account of the symmetries introduced by modelling.

Further Information

- http://www.cs.york.ac.uk/aig/constraints/
  - Our papers
  - Catalogue of CONJURE rules
  - Syntax and semantics of ESSENCE version 1
  - Catalogue of ~50 problems specified in ESSENCE and other constraint languages