Modelling with Constraints
Part 2: Formulating Abstract Models

Alan M Frisch
Artificial Intelligence Group
Dept of Computer Science
University of York

with significant contributions from Ian Miguel, Univ of St Andrews

Masterclass

12 December 2011
What is an Abstract Constraint Model?
Writing Abstract FD-Constraint Models

• Writing a what?
• A finite-domain CSP specification in which
  – Each domain is a finite set of finite objects, all of the same type, The type could be:
    • Atomic type: integers, Booleans, ...
    • Compound type: array, sequence, permutation, set, multiset, function, relation, partition, graph,…
  – Constraints and objective function use usual operators on these types of objects: set union, membership of a set or relation, function application,...
Writing Abstract FD-Constraint Models

- Each of the compound types is built using **type constructors**, e.g.:
  - Set of <type>. Examples: set of integer, set of Boolean
  - Function from <type> to <type>
    Example: function from integer to Boolean
- Type constructors can be **nested**, e.g.:
  - set of set of integer
  - set of function from integer to Boolean
- **Every object in a domain must be finite.**
  - So, *set of integer* means *finite set of integer*
  - \{ \{1,2,3\},{} \} is a domain of type *set of integer*.
  - \{ \{x | x is odd\}, \{2,4,6\} \} is not a domain.
Abstract FD-Constraint Models: Magic Square [CSPLib 019] as an Example

• **Given** a positive integer $n$
  
  **Find** *Magic*, an $n$-by-$n$ matrix containing values between 1 and $n^2$

  **Such that**

  every row, column and main diagonal of *Magic*

  has the same sum

  no two cells of *Magic* contain the same value
Reflection

Given a positive integer $n$

Find *Magic*, an $n$-by-$n$ matrix containing values between 1 and $n^2$

Such that

- Every row, column and main diagonal of Magic has the same sum
- No two cells of Magic contain the same value

- The domain of *Magic* is finite: size is $(n^2)^{(n^2)}$
- But only if we fix size of matrix and restrict values of *Magic* to come from a fixed, finite set.
Reflection

**Given** a positive integer \( n \)

**Find** *Magic*, an \( n \)-by-\( n \) matrix containing values between 1 and \( n^2 \)

**Such that**
- Every row, column and main diagonal of Magic has the same sum
- No two cells of Magic contain the same value

- This specifies a problem class, not a single problem instance.
- **Given** specifies the parameters. Each assignment of appropriate values to the parameters is an instance.
- **Find** specifies the decision variables and their domains, usually in terms of the parameters.
- **Such that** specifies the constraints, usually in terms of the parameters and decision variables.
**Reflection**

**Given** a positive integer $n$

**Find** *Magic*, an $n$-by-$n$ matrix containing values between 1 and $n^2$

**Such that**

- Every row, column and main diagonal of *Magic* has the same sum
- No two cells of *Magic* contain the same value

- Could use **Minimize** or **Maximize** to specify an objective function, usually in terms of the parameters and decision variables.
- For example: **Minimize** $\text{Magic}[1,1] + \text{Magic}[n,n]$
Formulating Abstract Constraint Models: Sometimes it is Trivial
Obvious Questions

• How hard is it to take a naturally-arising, well-formed problem statement and formulate it as an abstract constraint model?
• Is it natural to formulate a problem as an abstract constraint model?

Note: We are talking about formulating combinatorial problems not arbitrary problems.
Many Combinatorial Problem Statements are Abstract Constraint Models

- Magic Squares (as we have seen)
- Sudoku
- Many (if not all) of those in *Computers and Intractability: A Guide to the Theory of NP-Completeness* [Garey and Johnson, 1979]
- Knapsack Problem – as we will see
- Magic Sequence – as we will see
- Steiner Triple Systems – as we will see
Knapsack Problem from Garey and Johnson

**INSTANCE:** Finite set $U$, for each $u$ in $U$: a size $s(u) \in \mathbb{Z}^+$, a value $v(u) \in \mathbb{Z}^+$, and positive integers $B$ and $K$.

**QUESTION:** Is there a subset $U' \subseteq U$ such that

$$\sum_{u \in U'} s(u) \leq B \quad \text{and} \quad \sum_{u \in U'} v(u) \geq K$$

Notice: **INSTANCE** gives the parameters: $U$, $s$, $v$, $B$, $K$,
**QUESTION** gives the decision variable: $U'$ gives its domain: power set of $U$ gives two constraints
Steiner Triple Systems  [CSPLib 44]

• Given n, a positive integer
Find a set of n(n-1)/6 triples of elements from 1..n
Such that any pair of triples has at most one
common element.

• Note: an (unordered) triple is a set of size 3.
• If n = 7 a solution is
{ {1, 2, 3}, {1, 4, 5}, {1, 6, 7}, {2, 4, 6}
 {2, 5, 7}, {3, 4, 7}, {3, 5, 6} }
Reflection

- It appears that people – well, at least computer scientists – often think of combinatorial problems as abstract constraint models.
  - problem is to find a combinatorial object(s) of a certain type.
  - so use a decision variable of this type.
Reflection

• Types we have seen so far:
  – matrix of integer: magic square, Sudoku
  – set: knapsack
  – function from elements of a given set to integer: knapsack
  – sequence: magic sequence
  – unordered triple (set of size 3): Steiner triple systems
Reflection

Question:

Why am I making such a big deal about types?

Answer:

The process of building a concrete constraint model will be guided by the types. There will be one or more *modelling rules* for each type.
Formulating Abstract Constraint Models: Deriving Finite Domains
Some Problems Statements are Abstract Constraint Models but Domains Are Not Finite

• Often we can derive bounds to make the domains finite.
• Let’s see three examples:
  – Magic sequence problem
  – Kiselman semi-group problem
  – Golomb ruler problem
Magic Sequence Problem (MSP, CSPLib 19)

• Given \( n \), a non-negative integer. Find a sequence \( S \) of integers \( s_0, \ldots, s_n \) such that there are \( s_i \) occurrences of \( i \) in \( S \) for each \( i \) in \( 0, \ldots, n \).

• If \( n = 9 \), a solution is: \( (6, 2, 1, 0, 0, 0, 1, 0, 0, 0) \)

• This is not a finite domain specification: there is no bound on the elements of the sequence.
Given \( n \), a non-negative integer.
Find a sequence \( S \) of integers \( s_0, \ldots, s_n \)
Such that there are \( s_i \) occurrences of \( i \) in \( S \) for each \( i \) in 0..\( n \).

• No solution sequence of length \( n \) can contain a value greater than \( n \) since the sequence can’t contain more than \( n \) occurrences of any value.
• So we give require that every element of \( S \) is drawn from 0..\( n \).
• With a little more effort we could deduce that \( n-1 \) could not appear in any solution.
Kiselman Semigroup Problem (KSP)

• Given $n$, a positive integer.
  Find a sequence of integers drawn from $1..n$
  Such that between every pair of occurrences of an integer $i$ there exists an integer greater than $i$ and an integer less than $i$.

• If $n = 3$, a solution is 2, 3, 1, 2
• Interest usually focusses on counting the solutions for a given $n$.

• This is not a finite domain specification: there are an infinite number of finite sequences
Derive Finite Domain for KSP

Given $n$, a positive integer
Find a sequence of integers drawn from 1..$n$
Such that between every pair of occurrences of an integer $i$
  there exists an integer greater than $i$ and an integer less than $i$.

- Notice:
  There can be at most 1 occurrence of 1 and $n$.
  There can be at most 2 occurrences of 2 and $n-1$.
  There can be at most 3 occurrences of 3 and $n-2$.
- So, given $n$, we can derive a maximum sequence length:
  - for even $n$: $2(1+2+3+\ldots+n/2) = n(n+2)/4$
  - similar for odd $n$
Golomb Ruler Problem (GRP, CSPLib 006)

• Given a positive integer $n$.
  Find a set of $n$ integer ticks on a ruler
  Such that all inter-tick distances are distinct.
  Minimising the maximum tick

• Applications: x-ray crystallography, radio antenna placement

• This is not a finite domain problem: the domain containing every size-$n$ set of integers is infinite.
Golomb Ruler Problem: Example

Optimal Solution for $n = 4$ ticks
Derive Finite Domain for GRP

Given a positive integer \( n \).
Find a set of \( n \) integer ticks on a ruler
Such that all inter-tick distances are distinct.
Minimising the maximum tick

- It is easy to generate one feasible solution
  - 0, 1, 3, 7, 15, …, \( 2^n - 1 \)
- So every tick must be between 0 and \( 2^n - 1 \)
- Modify the Find statement
  - Find a set of \( n \) integer \((0 .. 2^n-1)\) ticks on a ruler
Formulating Abstract Constraint Models
Sometimes some work is needed
Generating Abstract Constraint Models

• Sometimes we are given a well-defined problem whose formulation is not quite an abstract constraint model.
• Need to fill in some detail.
Graph Colouring Problem: Well-formed Specification

• Given a graph and a positive integer $k$
  Colour each vertex of the graph with one of $k$ colours such that no edge connects two nodes of the same colour.

• Almost an abstract constraint model, but:
  – What is it that must be found? On the face of it, a function from vertices to colours.
  – But what are the colours? Could take them to be integers $1..k$. 
Graph Colouring Problem: Abstract Constraint Model 1

• Given a graph $G$ and a positive integer $k$
  Find a total function $\text{Col}$ from $\text{vertices}(G)$ to $1..k$.
  Such that for every edge $(v_1,v_2) \in G$, $\text{Col}(v_1) \neq \text{Col}(v_2)$

• Notice that $\text{Col}$ has a finite domain (size $|V|^k$)
• Notice that if we permute the colours, $1..k$, then we still have a solution. All such permutations are said to be symmetric to each other.
• Are symmetric solutions really the same? If so, how can we avoid multiple “versions” of the same solution?
Graph Colouring Problem: Abstract Constraint Model 2

- Given a graph G and a positive integer k
  Find a partition P of vertices(G) such that
  - P has no more than k groups
  - for every edge \((v_1,v_2) \in G\), \(v_1\) and \(v_2\) are in different groups of P

- Notice that P has a finite domain.
- Notice that a solution in which P has n groups corresponds to n! solutions of the previous spec.
- This is really a slightly different problem than the previous specification. (Advantage of rigour!)
Graph Colouring Problem: Abstract Constraint Model 3

- Given a graph G and a positive integer k
  Let colours be a set of size k
  Find a total function Col from vertices(G) to colours
  Such that for every edge \((v_1,v_2)\) in G,
  \(\text{Col}(v_1) \neq \text{Col}(v_2)\)

- This is specifying the same problem as the partition specification.
- How to output a solution.
  - Without loss of generality, let the colours be 1..k.
    Col maps vertex1 to 1,……
The Social Golfer Problem (SGP, CSPLib 010)

• In a golf club there are a number of golfers who wish to play together in $g$ groups of size $s$.
• Find a schedule of play for $w$ weeks such that no pair of golfers play together more than once.

• Solution to instance with 3 groups of size 3 over 3 weeks

<table>
<thead>
<tr>
<th>3 groups, size 3</th>
<th>3 weeks</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1, 2, 3]</td>
<td>[1, 4, 7]</td>
</tr>
<tr>
<td>[4, 5, 6]</td>
<td>[2, 5, 8]</td>
</tr>
<tr>
<td>[7, 8, 9]</td>
<td>[3, 6, 9]</td>
</tr>
<tr>
<td>[1, 5, 9]</td>
<td>[1, 5, 9]</td>
</tr>
<tr>
<td>[2, 6, 7]</td>
<td>[2, 6, 7]</td>
</tr>
<tr>
<td>[3, 4, 8]</td>
<td>[3, 4, 8]</td>
</tr>
</tbody>
</table>
SGP: An Abstract Constraint Model

• In each week, we need to **partition** the golfers into groups..
• What about the weeks?
  • A sequence? But what does the order matter?
  • A multiset.
  • In fact, there’s an **implied constraint** here. Can you see it?
• So we can think of the problem as finding a **multiset of partitions of golfers**
• What are the golfers?
  • An unnamed set of size g*s
SGP: An Abstract Constraint Model

• **Given** positive integers $g$, $s$, $w$,

• **Let** golfers be a set of size $g \times s$

• **Find** schedule: a multiset (of size $w$) of partitions (g groups, each with s members) of golfers.

• **Such that**
  
  For every two distinct members, $w_1$ and $w_2$, of schedule
  
  For every group $g_1$ in $w_1$ and group $g_2$ in $w_2$
  
  $|g_1 \cap g_2| \leq 1$
Balanced Incomplete Block Design Problem (BIBD, CSPLib 028)

• A BIBD is an arrangement of \( v \) distinct objects into \( b \) blocks such that
  – each block contains \( k \) distinct objects
  – each object occurs in exactly \( r \) different blocks
  – Every two distinct objects occur together in exactly \( \lambda \) blocks

• Applications: cryptography, experimental design
A BIBD is an arrangement of 6 distinct objects into 10 blocks such that
- each block contains 3 distinct objects
- each object occurs in exactly 5 different blocks
- Every two distinct objects occur together in exactly 2 blocks

```
OBJECTS
0 0 0 0 0 1 1 1 1 1
0 0 1 1 1 0 0 0 1 1
0 1 0 1 1 0 1 1 0 0
1 0 1 0 1 1 0 1 0 0
1 1 0 1 0 1 0 0 0 1
1 1 0 1 0 1 0 0 0 1
1 1 1 0 0 0 1 0 1 0
1 1 1 0 0 0 1 0 1 0

BLOCKS
0 0 0 0 0 1 1 1 1 1
0 0 1 1 1 0 0 0 1 1
0 1 0 1 1 0 1 1 0 0
1 0 1 0 1 1 0 1 0 0
1 1 0 1 0 1 0 0 0 1
1 1 1 0 0 0 1 0 1 0
```
BIBD Problem

• A BIBD is an arrangement of $v$ distinct objects into $b$ blocks such that
  – each block contains $k$ distinct objects
  – each object occurs in exactly $r$ different blocks
  – Every two distinct objects occur together in exactly $\lambda$ blocks

• Each block is a set of objects and we need to find a set of blocks. Thus, need to find a set of sets of objects.

• Alternatively: Notice each block has multiple ($k$) objects and each object occurs in multiple ($r$) blocks. Thus, need to find a relation between blocks and objects.
BIBD Problem: Abstract Constraint Model 1

- A BIBD is an arrangement of $v$ distinct objects into $b$ blocks such that
  - each block contains $k$ distinct objects
  - each object occurs in exactly $r$ different blocks
  - Every two distinct objects occur together in exactly $\lambda$ blocks

- Given $v$, $b$, $r$, $k$, $\lambda$
- Let objects be a set of size $v$
- Find BIBD: a set (of size $b$) of set (of size $k$) of objects
- Such that
  - $\forall o \in \text{objects}, |\{\text{block} \in \text{BIBD} | o \in \text{block}\}| = r$
  - For every two distinct $o_1$ and $o_2$ in objects
    $(\sum_{\text{block} \in \text{BIBD}} \{o_1,o_2\} \subseteq \text{block}) = \lambda \langle \langle$
Reflection

• This spec doesn’t introduce names for the blocks.
• The constraint

$$\forall o \in \text{objects}, \ |\{\text{block} \in \text{BIBD} \mid o \in \text{block}\}| = r$$

could have been written as

for every $o$ in objects, $(\sum_{\text{block} \in \text{BIBD}} o \in \text{block}) = r$

This is useful as constraint languages typically support this but not set comprehension.
BIBD Problem: Abstract Constraint Model 2

• A BIBD is an arrangement of $v$ distinct objects into $b$ blocks such that
  – each block contains $k$ distinct objects
  – each object occurs in exactly $r$ different blocks
  – Every two distinct objects occur together in exactly $\lambda$ blocks

• Given $v$, $b$, $r$, $k$, $\lambda$: positive integers
• Let objects be a set of size $v$ and blocks be a set of size $b$
• Find BIBD, a relation on objects $X$ blocks
• Such that
  – $\forall (b \in \text{blocks}) \ (\sum_{(o \in \text{objects})} \langle o, b \rangle \in \text{BIBD}) = k$
  – $\forall (o \in \text{objects}) \ (\sum_{(b \in \text{blocks})} \langle o, b \rangle \in \text{BIBD}) = r$
  – $\forall (o_1, o_2 \in \text{objects}) \ o_1 \neq o_2 \rightarrow$
    $\sum_{(b \in \text{blocks})} (\langle o_1, b \rangle \in \text{BIBD} \land \langle o_2, b \rangle \in \text{BIBD}) = \lambda$
Reflection

• Generally, a relation on $A \times B$ in which either $A$ or $B$ is an unnamed set can be considered as a set of sets and vice-versa.

• For example, consider the relation
  $\{ \langle a1,b1 \rangle, \langle a1,b2 \rangle, \langle a2,b2 \rangle, \langle a2,b3 \rangle \}$

If $a1$ and $a2$ are arbitrary names (that is, $A$ is really an unnamed set) then this is same as the set of sets
  $\{ \{b1, b2\}, \{b2, b3\} \}$
Formulating Abstract Constraint Models
A Last Resort
Dealing with Infinite Domains

• For the Kiselman Sequence Problem and Golomb Ruler problem we were able to bound the domains to make them finite.
• For some problems either:
  • We cannot derive a bound.
  • Any bound we can derive is so weak as to be useless.
• This is often the case when modelling planning problems.
  • Difficult to tell how many actions are going to be needed to achieve the goal state.
Unbounded Sequences

• Solution: solve a series of CSPs, incrementally increasing the length of the sequence.
• i.e. Build a model for a sequence of length 1 and try to solve it
  • If no solution, repeat for length 2.
  • If no solution, repeat for length 3…
• This way we find a solution with the shortest sequence
Wrapping Up
Types We Have Seen

- matrix of integer: magic square, Sudoku
- set of atoms (objects): knapsack
- set of integer: Golomb ruler problem
- function from atoms (objects) to integer: knapsack
- function from vertices to atoms (colours) or int: graph colouring
- sequence of integer: magic sequence, Kiselman semigroup
- triple (set of size 3) of integer: Steiner triple systems
- graph: graph colouring
- partition of vertices: graph colouring
- multiset of partition of atoms (golfers): social golfers
- set of set of atoms (objects): BIBD
- relation on atoms (blocks) X atoms (objects): BIBD
Further Examples

• A catalog of abstract constraint models written in the ESSENSE language:
  www.cs.york.ac.uk/aig/constraints/AutoModel/Essence/specs120

• These should be readable even without knowledge of ESSENCE
You Try One: The SONET Problem

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

- **Sample Instance**:
  - \texttt{nrings}=2, \texttt{nnodes}=5, \textit{capacity} = 4
  - demand = \{\(n_1 \& n_3\), \(n_1 \& n_4\), \(n_2 \& n_3\), \(n_2 \& n_4\), \(n_3 \& n_5\}\}

- **Solution**:
SONET:

given

nrings, nnodes, capacity : int (1...),
demand : set of set (size 2) of int (1..nnodes)

find

network : mset (size nrings) of

set (maxSize capacity) of int (1..nnodes)

minimising

\[ \sum_{\text{ring} \in \text{network}} |\text{ring}| \]

such that \( \forall \text{pair} \in \text{demand} . \exists \text{ring} \in \text{network} . \text{pair} \subseteq \text{ring} \)