Constraint Programming

Constraint Modelling – Thinking Abstractly
The Return of Modelling
A constraint model maps the features of a combinatorial problem onto the features of a constraint satisfaction problem (CSP).
Model a problem for a particular constraint solver

1. Ill-defined problem statement
2. Well-defined problem statement
3. Solver-independent Constraint Model
4. Solver-specific Constraint Model

- We usually consider constraint modelling to be mapping 2 to 3.
An Extra Step

1. Ill-defined problem statement
2. Well-defined problem statement
3. Solver-independent Constraint Model
4. Solver-specific Constraint Model

When viewed **abstractly**, many combinatorial problems that we wish to tackle with constraint solving exhibit **common features**.

Abstractly: **above** level at which constraint modelling decisions made.⁵
An Extra Step

1. Ill-defined problem statement

2. Well-defined problem statement

3. Solver-independent Constraint Model

4. Solver-specific Constraint Model

• E.g Many problems require us to find objects such as:
  • (Multi-)sets
  • Relations
  • Functions

• These are patterns in the problems we want to solve.

• We can write our abstract constraint model in terms of these patterns. But…
An Extra Step

1. Ill-defined problem statement

2. Well-defined problem statement

3. Solver-independent Constraint Model

4. Solver-specific Constraint Model

- E.g. Many problems require us to find objects such as:
  - (Multi-)sets
  - Relations
  - Functions
- Typically, not supported directly by constraint solvers.
- So need to model them as constrained collections of more primitive objects.
An Extra Step

1. Ill-defined problem statement
2. Well-defined problem statement
A. Abstract Constraint Model
3. Solver-independent Constraint Model
4. Solver-specific Constraint Model

- E.g Many problems require us to find objects such as:
  - (Multi-)sets
  - Relations
  - Functions
- Develop corresponding **modelling patterns** for representing and constraining these combinatorial objects.
- We can **reduce effort** required when modelling a new problem.
We will use the same skeleton as CSPs:

- **Given** parameters
- **Find decision** variables
- **Such that** constraints

Much richer set of types of decision variable:

- Sequence, set, multiset, …
- Constraints & objective function use usual operators on these types of objects:
  - Set union, membership of a set or relation, function application.
Patterns in Abstract Constraint Models

- We will look at a number of individual patterns.
- We will then look at how these patterns can be combined to model more complex problems.
Sequences
A sequence is an ordered list of elements. In the sense that a sequence has a first element, a second element, etc. Repetition is allowed.

Examples:
- 0, 1, 1, 2, 3, 5, 8, 13.
- Turn right, drive $\frac{1}{4}$ mile, turn right, drive $\frac{1}{2}$ mile, turn left.
Where does the Sequence Pattern Occur?

- Planning Problems:
  - Find a **sequence of actions** to transform an initial state into a goal state.
  - Example: Peg Solitaire (CSPLib 38).

![Sequence Pattern Diagram](www.csplib.org)
Where does the Sequence Pattern Occur?

• Communications:
  • Low Autocorrelation Binary Sequences (CSPLib 5).

• Mathematics:
  • Langford’s Problem (CSPLib 24).
  • Error-Correcting Codes (CSPLib 36).

• Puzzles:
  • Magic Sequences (CSPLib 19).
Fixed-length Sequences

- Problems of the form:
  - Given $n$,
  - Find a sequence of objects of length $n$,
  - Such that …
Fixed-length Sequences

• Example (Magic Sequence, CSPLib 19):
  • Given \( n \).
  • 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
Fixed-length Sequences

- Problems of the form:
  - Given $n$,
  - Find a sequence of objects of length $n$, such that …

- Most straightforward model: use an array of decision variables indexed 1..$n$. Domains are the objects to be found.

- Example, find a sequence of $n$ digits:

```
DigitsArray 1 2 3 4 n
 0..9 0..9 0..9 0..9 ...
```
Fixed-length Sequences

Example: Magic Sequence

- **Given** \( n \), a non-negative integer
- **Find** a sequence \( S \) of integers \( s_0, \ldots, s_n \) each of which is between 0 and \( n \)
- **Such that** there are \( s_i \) occurrences of \( i \) in \( S \) for each \( i \) in \( 0, \ldots, n \).

\[
\begin{array}{cccccc}
0 & 1 & 2 & 3 & \ldots & n \\
\hline
s': & 0..n & 0..n & 0..n & 0..n & \ldots & 0..n
\end{array}
\]

Constraints:
For all \( i \) in 0..\( n \).
No. of occurrences of \( i \) in \( S' \) is \( S'[i] \)
Bounded-length Sequences

- Problems of the form:
  - Given $n$,
  - Find a sequence of objects of length at most $n$,
  - Such that …
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

• We are usually interested in counting the solutions
  for a given \( n \).

• This is not a finite domain specification: there are an
  infinite number of finite sequences
Problem: Infinite Domains

- The KSP as stated is not finite.
  - Find a sequence of integers drawn from 1..n
  - The domain of this “sequence” variable is infinitely large.

- Why is this a problem?
  - Because we want to map our abstract constraint specification down to a finite-domain CSP.
  - With infinite domains in the abstract specification, this will not be possible.
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 4 occurrences of 3 and \( n-2 \).

• So, given \( n \), we can derive a maximum sequence length:
• For even \( n \): \( 1+2+4+8+\ldots+2^{n/2-1} = 2^{n/2+1}-2 \)
• Similarly for odd \( n \).
• Hence, domain of the sequence variable is finite.
Bounded-length Sequences

Given \( n \) a positive integer
Find a sequence of at most \( 2^{n/2+1}-2 \) 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 \( i \).

Again, we can use an array indexed \( 1.. 2^{n/2+1}-2 \):

\[
S': \begin{array}{cccc}
1 & 2 & 3 & 4 \\
\vdots & \ddots & \ddots & \ddots \\
\end{array}
\]

Problem: What if a solution has length less than \( 2^{n/2+1}-2 \)?
Example: The empty sequence is always a solution to this problem.
Bounded-length Sequences

Given \( n \) a positive integer
Find a sequence of at most \( 2^{n/2+1}-2 \) 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 \( i \).

**Problem**: What if a solution has length less than \( 2^{n/2+1}-2 \)?

**Solution**: Use a dummy value in the domain. In this case we use 0:

\[
\begin{array}{cccccc}
1 & 2 & 3 & 4 & \cdots & 2^{n/2+1}-2 \\
S': & 0..n & 0..n & 0..n & 0..n & \cdots & 0..n
\end{array}
\]
Bounded-length Sequences

Given \( n \) a positive integer

Find a sequence of at most \( 2^{n/2+1}-2 \) 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 \( i \).

Constraint:

For all \( i \) in \( 1..2^{n/2+1}-2 \). For all \( j \) in \( i+1..2^{n/2+1}-2 \).

\( S'[i] = S'[j] \neq 0 \rightarrow \)

exists \( ls, gt \) in \( i+1..j-1 \).

\( (S'[ls] \neq 0 \land S'[ls] < S'[i]) \land S'[gt] > S'[i]) \)
Bounded-length Sequences

- So, for $n = 4$ and the solution 1, 2 the variables might be assigned:

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
</table>

**Problem**: They might also be assigned

| 1 | 0 | 0 | 2 | 0 | 0 |

Adding the dummy value has created **equivalence classes** of assignments.
Bounded-length Sequences

• Adding the dummy value has created equivalence classes of assignments.
• Solution: choose a canonical element from each class.
• E.g. all 0s must appear at the end of the sequence:
  • For all \( i \) in \( 1..2^{n/2+1}-3 \) . \((S'[i] = 0) \rightarrow (S'[i + 1] = 0)\)
• This constraint rejects all other equivalents.
Something to Note

• It is **very common** when modelling an abstract object to **introduce equivalences** during modelling.

• Need to be aware of this happening, and of the measure used to counter it.
Unbounded Sequences: Dealing with Infinite Domains

• For the Kiselman problem, we were able to bound the sequence length (relatively) straightforwardly.

• For some problems either:
  • We cannot derive a bound.
  • Any bound we can derive is so weak as to be useless.
Unbounded Sequences: Dealing with Infinite Domains

• This is an example of a situation where we have to deal with an infinite domain.
• 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. Try and find a solution for a sequence of length 1.
  - If no solution, try length 2.
  - If no solution, try length 3 …
- This way we find a solution with the shortest sequence.
Permutations

• Some problems involve finding a sequence of elements where:
  • The elements in the sequence are known
  • Their arrangement is not.
• I.e. find a permutation of the sequence.
Permutations

Example: The Travelling Salesman Problem

• Given a network of cities, known distances between every pair of cities, and a starting city.
• Find shortest route that visits all points, returns to start.

Image from www.jimloy.com
Permutations

Example: The Car Sequencing Problem (CSPLib 1)
- Given a set of cars that must be manufactured
- Find an order in which they should go down the conveyor belt such that...

Figure from Foundations of Constraint Satisfaction [Tsang, 1993]
Permutations: First Viewpoint

- Assume that the elements of the permutation are distinct.
- First viewpoint is as fixed-length. If permutation contains elements a, ..., f:

<table>
<thead>
<tr>
<th>Perm1</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>a..f</td>
<td>a..f</td>
<td>a..f</td>
<td>a..f</td>
<td>a..f</td>
<td>a..f</td>
<td>a..f</td>
</tr>
</tbody>
</table>

**Constraint**: All-different(Perm1)
Permutations: Second Viewpoint

• Alternatively, we know the elements that appear in the sequence.

• So we can index by those elements:

  \[
  \begin{array}{ccccccc}
  \text{Perm2} & a & b & c & d & e & f \\
  \hline
  1..6 & 1..6 & 1..6 & 1..6 & 1..6 & 1..6 \\
  \end{array}
  \]

  **Constraint:** All-different(Perm2)

• Domain values represent the position in the sequence an element is in. So “badcef” would be:

  \[
  \begin{array}{ccccccc}
  \text{Perm2} & a & b & c & d & e & f \\
  \hline
  2 & 1 & 4 & 3 & 5 & 6 \\
  \end{array}
  \]
Permutations: Which Viewpoint to Use?

- Depends on the constraints on the permutation.
- Example:: a and b must be adjacent.

<table>
<thead>
<tr>
<th>Perm1</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>a..f</td>
<td>a..f</td>
<td>a..f</td>
<td>a..f</td>
<td>a..f</td>
<td>a..f</td>
<td>a..f</td>
</tr>
</tbody>
</table>

For all i in 2..5. Perm1[i] = a →

(Perm1[i-1] = b ∨ Perm1[i+1] = b)

repeat with a and b swapped
Permutations: Which Viewpoint to Choose?

- Depends on the constraints on the permutation.
- Example: a and b must be adjacent.

<table>
<thead>
<tr>
<th>Perm2</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1..6</td>
<td>1..6</td>
<td>1..6</td>
<td>1..6</td>
<td>1..6</td>
<td>1..6</td>
</tr>
</tbody>
</table>

\[ | \text{Perm2}[a] - \text{Perm2}[b] | = 1\]
Permutations: Which Viewpoint to Choose?

- Depends on the constraints on the permutation.
- Example: The first three letters of the sequence must form an English word.

<table>
<thead>
<tr>
<th>Perm1</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>a..f</td>
<td>a..f</td>
<td>a..f</td>
<td>a..f</td>
<td>a..f</td>
<td>a..f</td>
</tr>
</tbody>
</table>

Just need a table constraint on the first three variables in Perm1 that allows “bad”, “cad”, “fad”, …
Permutations: Which Viewpoint to Choose?

• Depends on the constraints on the permutation.

• Example: The first three letters of the sequence must form an English word.

<table>
<thead>
<tr>
<th>Perm2</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1..6</td>
<td>1..6</td>
<td>1..6</td>
<td>1..6</td>
<td>1..6</td>
<td>1..6</td>
</tr>
</tbody>
</table>

Horrible: (Perm2[a] = 1 and Perm2[c] = 2 and Perm2[e] = 3) or …
Sequences: Summary

• Fixed-length.
• Bounded-length.
• Unbounded.
• Permutations.
• Try some of the problems from CSPLib!