

Masterclass

Modelling with Constraints

Part 4: Formulating Concrete Models (nested types)

Alan M Frisch

Artificial Intelligence Group

Dept of Computer Science

University of York

with significant contributions from Ian Miguel, Univ of St Andrews

12 December 2011

Recap

- We have seen how abstract constraint models can be reformulated as concrete constraint models that are equivalent in a certain sense.
- The reformulation is driven by the types. The types we have considered so far are
 - sequences of τ , including permutations
 - sets of τ
 - relations on $\tau_1 \times \tau_2$
 - functions on from τ_1 to τ_2 , including total, surjective, injective and bijective functionswhere τ is **atomic** (Boolean or integer).

Now Consider “Nested” Types

- Now consider the case where τ is non-atomic.
- These are called nested types.
- For example:
 - sequence of set of τ
 - set of set of τ
 - set of partition of τ

Basic Idea

- A set of integers is represented by a 1-dimensional array.
- So a set of sets of integers is represented by a 1-dimensional array in which each element is a 1-dimensional array --- that is, a 2-dimensional array.
- The basic process is to combine the representations we have seen so far.

Nesting

Abstract Models Often Employ Nested Types

Recall the Steiner Triple Problem:

Given n , a positive integer

Find a set of $n(n-1)/6$ triples of elements from $1, \dots, n$

Such that any pair of triples have at most one
common element.

This abstract model asks to find a set of triples,
where a triple is a set of size 3

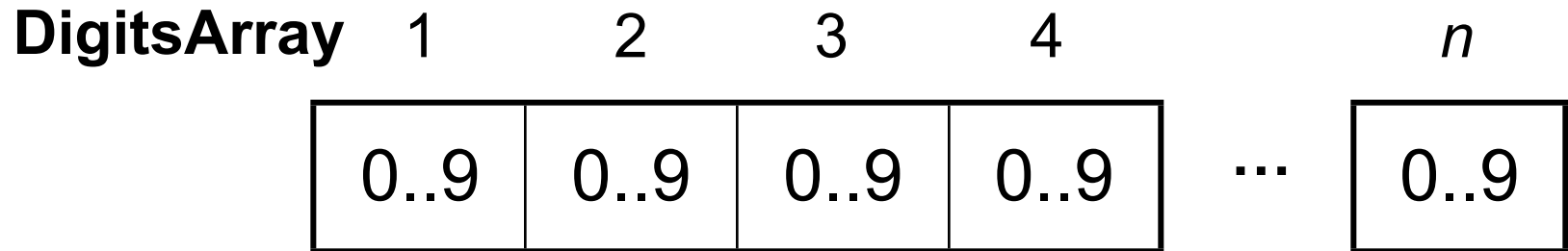
Abstract Models Often Employ Nested Types

- Planning Problems:
 - Find a **sequence of actions** to transform an initial state into a goal state.
- When a planning problem allows actions to be performed in parallel in a single step, it is natural to characterise it as a **sequence of sets** of actions.
- In a logistics problem, an action may involve transporting a set of packages. So a plan is a sequence of sets.

Nesting Inside Sequences

Nesting Inside Sequences

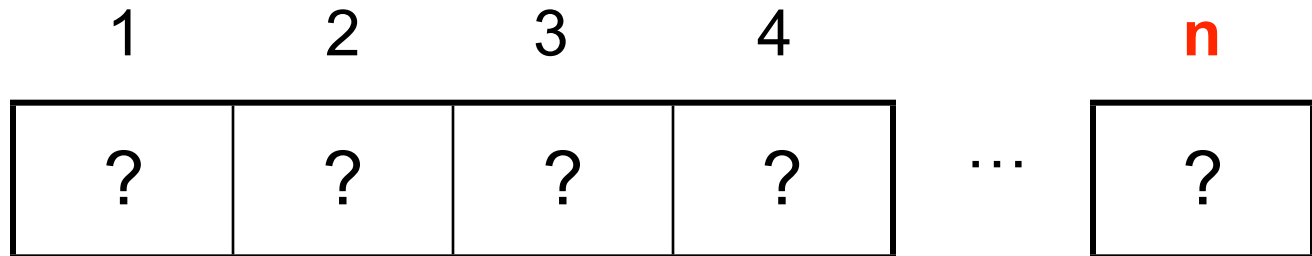
- Recall how we modelled a sequence of length n by an array of decision variables indexed $1..n$. Domains are the objects to be found.
- Example, find a sequence of n digits:



Nesting Inside Sequences

- Assume now that we must find a sequence of sets, functions, relations, ...
- A single variable at each index is not adequate to represent the compound object at that position.

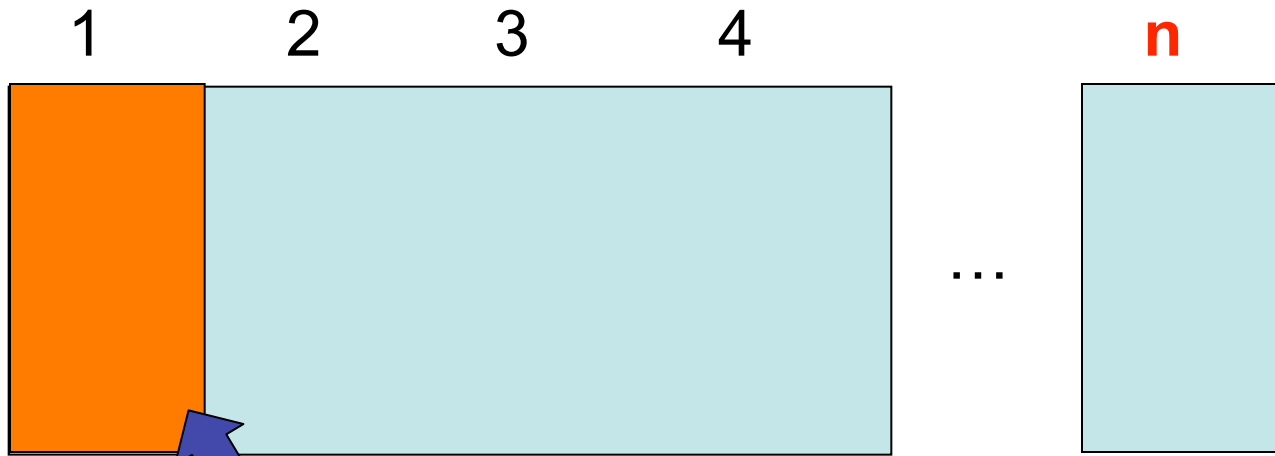
NestedSeqArray



Nesting Inside Sequences

- Simple solution:
 - Extend the dimension of the array.

NestedSeqArray



E.g. 2 dimensions.

We now have a **column** of variables to represent our set, relation, function ...

Example: Sequence of Sets

Given n, k

Find a sequence of length n of sets of size ≤ 2 drawn from $1..k$.

- Let's use the **explicit** representation of each set and 0 as the dummy value:

	1	2	3	4	...	n
1	0..k	0..k	0..k	0..k	...	0..k
2	0..k	0..k	0..k	0..k	...	0..k

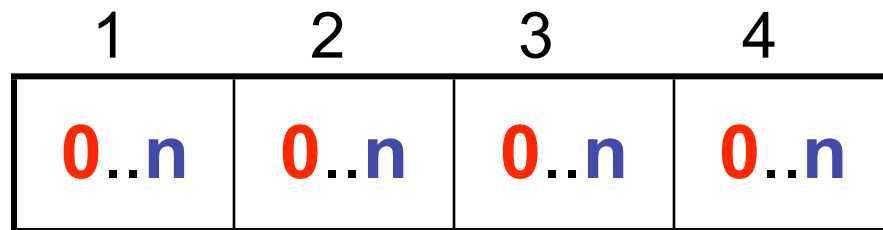
Constraint:

Each column is in ascending order

Nesting Inside Sequences

- What if the sequence has bounded length?
- Recall that in the non-nested case we used a dummy value:

KisSequence



Nesting Inside Sequences

- Use the same approach here.
- Instead of dummy values could use auxiliary **switch** variables to indicate whether the corresponding column is part of the sequence.
- Be careful of introducing equivalence classes of assignments.

	1	2	3	4	...	n
Seq	0..k	0..k	0..k	0..k	...	0..k
Switches	0, 1	0, 1	0, 1	0, 1	...	0, 1

Nesting Inside Sets

Nesting Inside Sets

- Being asked to find a set of some other object is common, so it is worth considering how to model this type of problem.
- Now we must choose how to model the outer type (e.g. explicit vs occurrence model of sets) **as well** as the inner.

Nested Sets

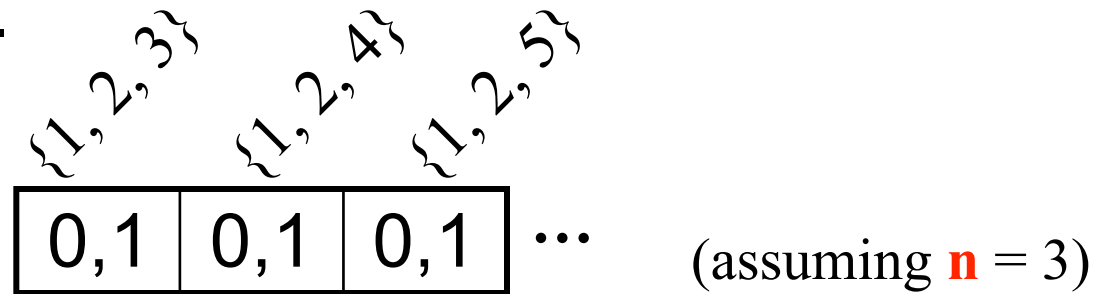
Given m, n
Find a set (size m) of sets of n digits

- From what we have seen so far, we have three possibilities:
 1. An occurrence representation.
 2. Outer: Explicit. Inner: Occurrence.
 3. Outer: Explicit. Inner: Explicit.

Nesting Inside Sets: Occurrence

Given m, n
Find a set (size m) of sets of n digits

- Remember: explicit representation of a set is an array whose index values are the possible elements of the set.
- In this case use an array indexed by the possible sets of n digits.

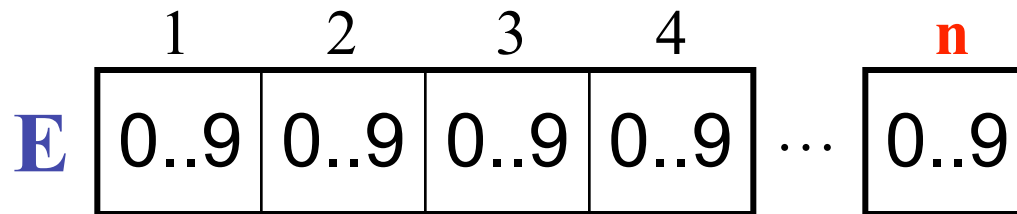


This is often not feasible.

Typically, when dealing with nesting the outer layers are represented **explicitly**.

Nesting Inside Sets: Outer Explicit

- Recall the explicit representation of a fixed-cardinality set of digits:

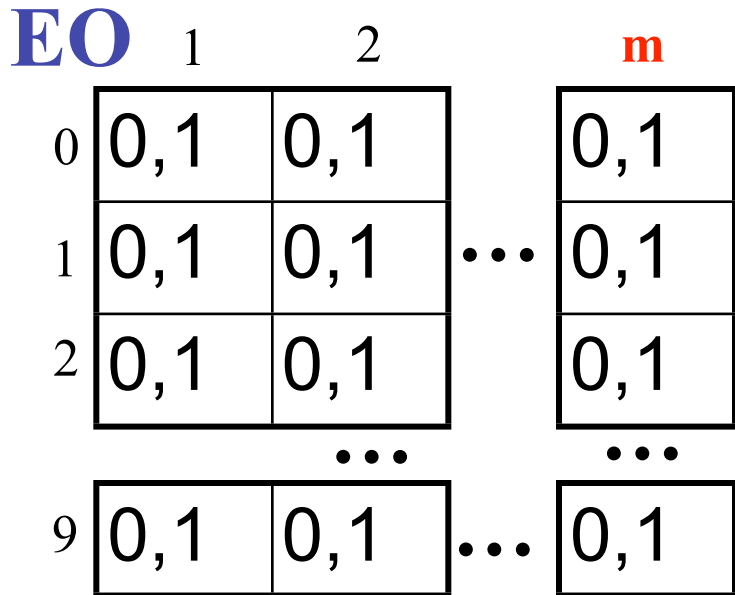


- As in the sequence example, we extend the dimension of **E** according to the representation we choose for the inner set.
- Also need to ensure that the elements of the outer set are **distinct**.

Nesting Inside Sets: Explicit/Occurrence

Given m, n
Find a set (size m) of sets of n digits

- Consider an occurrence representation for the inner sets.



Constraints:

Each column sums to n

No two columns are identical

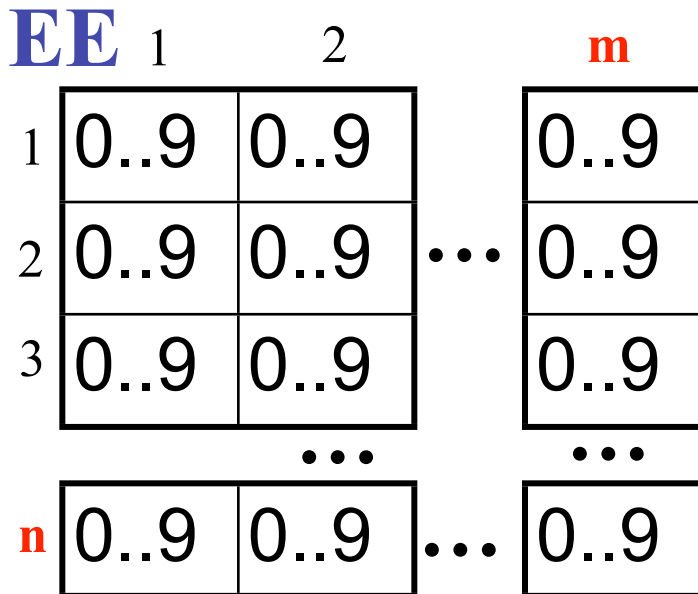
But what about

equivalence classes?

Nesting Inside Sets: Explicit/Explicit

Given m, n
Find a set (size m) of sets of n digits

- Let's consider an explicit representation for the inner sets.



Constraints:

No two columns are the same.

Each column is in ascending order.

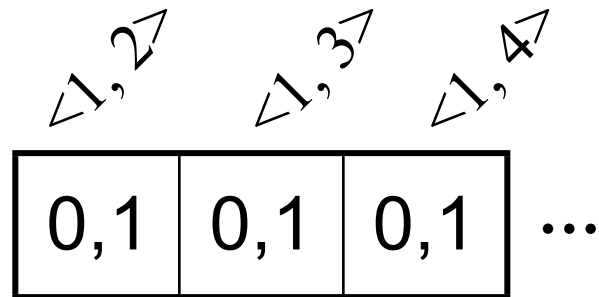
But what about **equivalence classes?**

Relations as Sets of Tuples

- We can also view relations as **sets of tuples**.
- Recall our example:
 - Find a relation R between sets $A = \{1, 2, 3\}$ and $B = \{2, 3, 4\}$ such that...
- What happens when we try and model this from the perspective of a set of tuples?

Relations as Sets of Tuples: Occurrence

- We have an array indexed by the possible tuples:



- Basically same as the occurrence representation we came up with directly:

		A		
		1	2	3
B	2	0, 1	0, 1	0, 1
	3	0, 1	0, 1	0, 1
	4	0, 1	0, 1	0, 1

Relations as Sets of Tuples: Explicit

Find a relation R between sets $A = \{1, 2, 3\}$ and $B = \{2, 3, 4\}$
Such that...

- Maximum number of tuples is 9. Invoke the bounded-cardinality set pattern:

	1	2	3	4	...	9
1	1..3	1..3	1..3	1..3	...	1..3
2	2..4	2..4	2..4	2..4	...	2..4

Constraint: columns are all different.

What about **equivalence classes**?

What if the relation allows fewer than the full 9 tuples?

The Social Golfers Problem

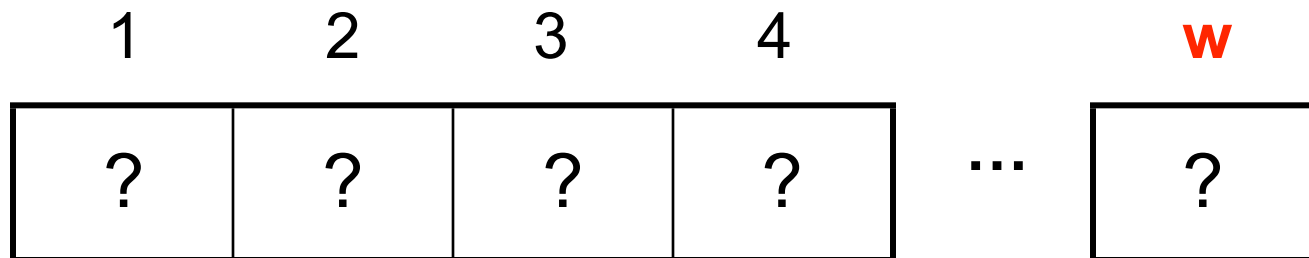
SGP: An Abstract Constraint Model

- **Given** positive integers g, s, w ,
- **Let** golfers be a set of size $g \cdot s$
- **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
 - $|g_1 \cap g_2| \leq 1$

Since we haven't considered multisets, let's use the inference that we are looking for a set of partitions.

SGP: Representing the Outer Set

- We have seen explicit and occurrence representations of sets.
- The set contains complex objects (partitions).
- Indexing an array by the possible partitions of golfers doesn't seem appealing.
- So let's try an explicit model with one entry for each of the **w** weeks:



SGP: The Partitions

- In each week we want to partition the golfers into **g** groups of size **s**.
- That is, a set of cardinality **g** of disjoint sets of cardinality **s**
- As per the previous discussion, probably sensible to represent the outer set explicitly.
- The inner set could be occurrence or explicit. Here we'll talk about an explicit/explicit representation.

SGP: The Partitions

(explicit/explicit representation)

- Let n = number of golfers = $g * s$.

partition of golfers
representing the
golfers by 1..n
groups by 1..g

		Groups			
		1	2	...	g
golfers in group	1	1..n	1..n	...	1..n
	2	1..n	1..n	...	1..n
	3	1..n	1..n	...	1..n

	s	1..n	1..n	...	1..n

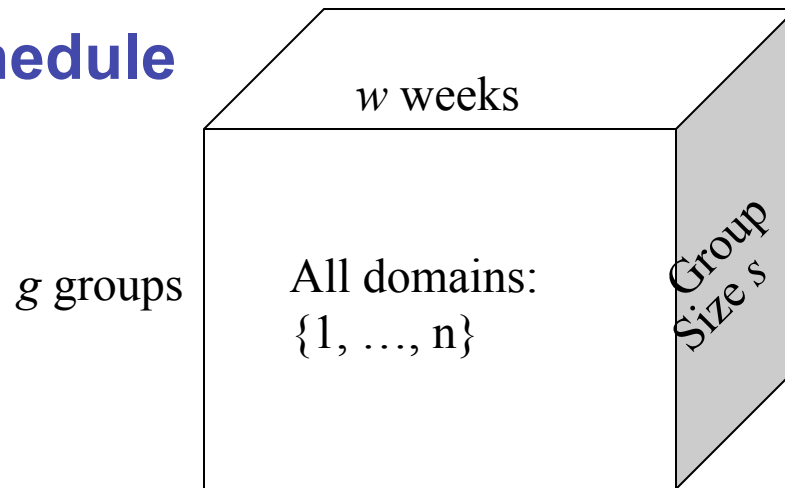
Constraints:

- The elements of this matrix are all different. (assures a partition)
- Each column is in ascending order. (remove equivalences)

A Multiset of Partitions of Golfers

Putting a **week** (a partition represented as a 2D array) into each slot of the explicit set representation, we obtain a 3D array.

Schedule



NB $n = g \times s$ is
number of golfers

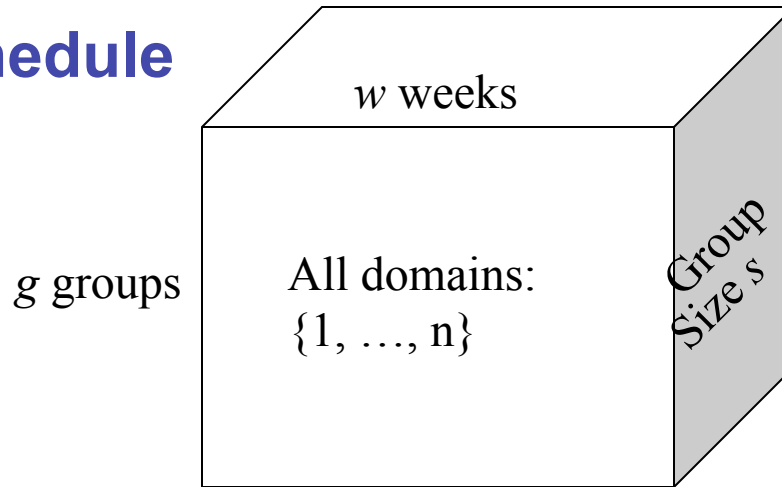
We can **order the weeks** in strict lexicographic order to counter the equivalence of assignments obtained by permuting the weeks and to ensure distinct elements.

To do this we need to flatten each week (a 2D matrix) into a 1D matrix.

A Multiset of Partitions of Golfers

- Need to ensure no pair of golfers meet more than once.

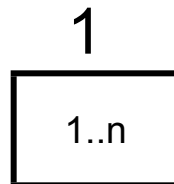
Schedule



NB $n = g \times s$ is number of golfers

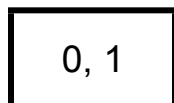
Equivalently: size of intersection of each pair of groups is at most 1. Invoking our intersection pattern:

Intersection



Sum of switches is at most 1

Switches



Nesting Summary

- Modelling problems involving nested combinatorial objects can be quite tricky.
- Using the patterns we've been looking at can help you to do it **systematically**.
- It can also help in spotting **equivalence classes** of assignments as you introduce them.
 - Which can be substantially easier than trying to detect them after the fact.