# Thinking Abstractly About Constraint Modelling II

Ian Miguel
ianm@cs.st-andrews.ac.uk

# Previously on the X-files…

- When viewed **abstractly**, many combinatorial problems that we wish to tackle with constraint solving exhibit **common features**.
- By recognising these commonly-occurring patterns, and
- Developing corresponding **modelling patterns** for representing and constraining these combinatorial objects,
- We can **reduce effort** required when modelling a new problem.

# Previously on the X-files…

- We saw a number of individual patterns:
  - Sequences.
  - (Multi-)Sets.
  - Relations.
  - Functions.

# Previously on the X-files…

- We saw how modelling can introduce **equivalence classes** of assignments

**KisSequence**

| 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|
| **1** | **2** | **0** | **0** | **0** | **0** |

**KisSequence**

| 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|
| **1** | **0** | **0** | **2** | **0** | **0** |

- Need to be aware of this happening, know how to counter it.
- Reduces the need for detection of such equivalences.

# In This Episode

- We will see how these individual patterns can be **combined** to model more complex problems.

# Nesting

# Nesting Overview

- We've seen how to model several combinatorial objects.
- Often, problems require us to find one combinatorial object **nested** inside another.
  - A set of sets,
  - A sequence of functions…

# How Common Are Problems Involving Nesting?

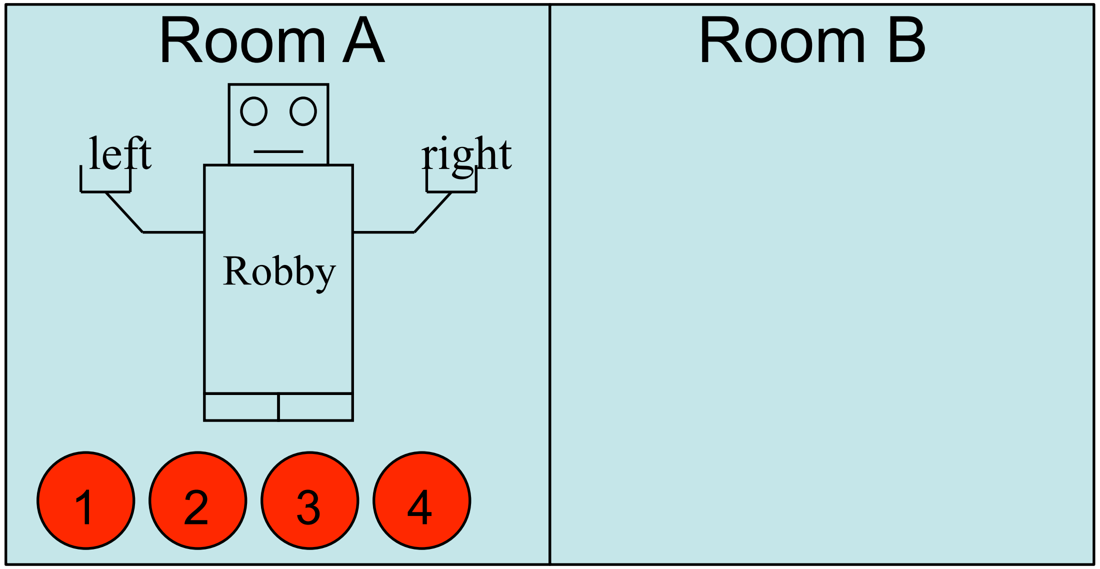
- Very.
- Recall the Steiner Triple (CSPLib 44) problem:
  - Given n, find a set of n(n-1)/6 triples of elements from 1,…,n such that any pair of triples have at most one common element.
  - If n = 7:
    - {{1, 2, 3}, {1, 4, 5}, {1, 6, 7}, {2, 4, 6}, {2, 5, 7}, 3, 4, 7}, {3, 5, 6}}
  - This is a **set of sets** (the triples).

# How Common Are Problems Involving Nesting?

- Planning Problems:
  - Find a **sequence of actions** to transform an initial state into a goal state.
- When a planning problem allows us actions to be performed in parallel in a single step, it is natural to characterise it as a **sequence of sets** of actions.

# How Common Are Problems Involving Nesting?

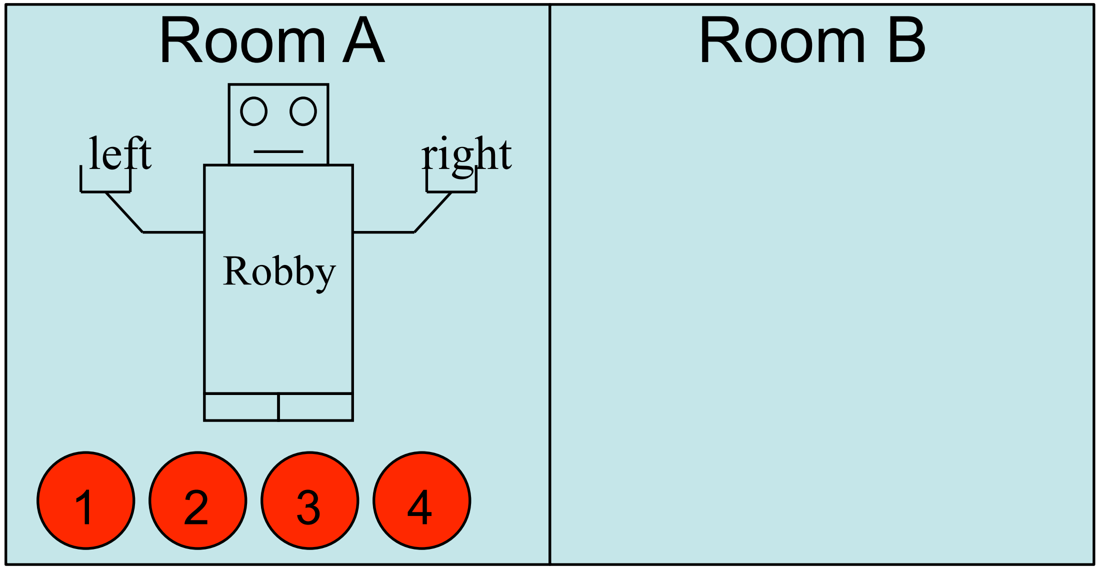
- Example: The Gripper Problem




- **Goal**: All balls in Room B.
- **Operators**: pick up, put down, move.

# How Common Are Problems Involving Nesting?

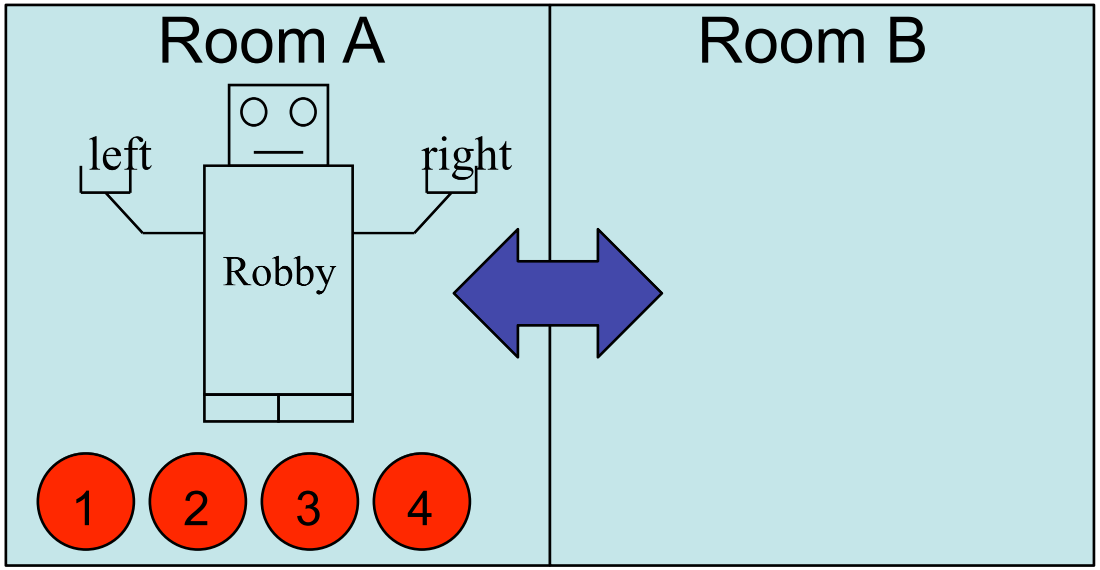
- Example: The Gripper Problem




- Since Robby has two grippers, in a single step of the plan he can pick up/put down two balls.

# How Common Are Problems Involving Nesting?

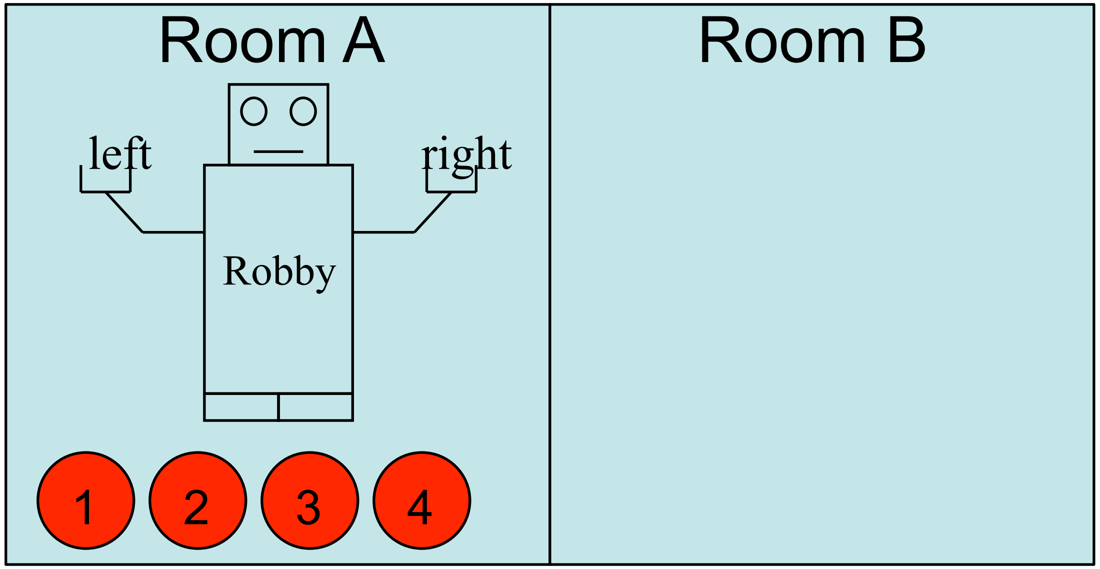
- Example: The Gripper Problem




- When Robby moves, he can't pick up/put down.
- So at most **2** actions per step.

# How Common Are Problems Involving Nesting?

- Example: The Gripper Problem




- Natural to characterise this problem as finding a **sequence of sets** of maximum cardinality 2.

# How Common Are Problems Involving Nesting?

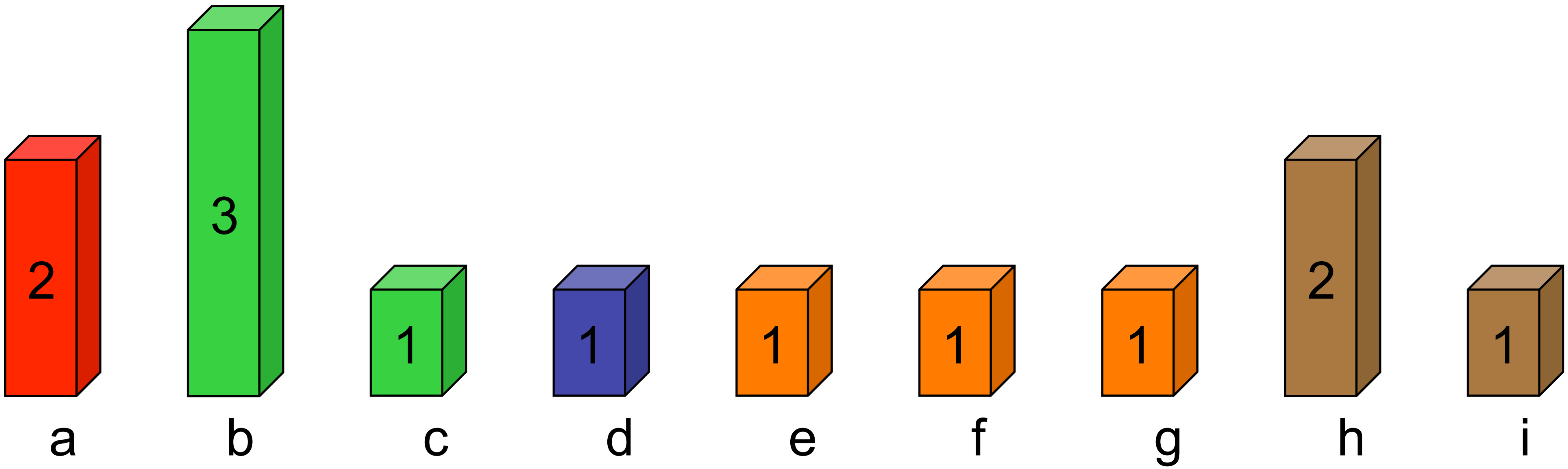
- Example: Steel Mill Slab Design (CSPLib 38).
- The mill can make $\sigma$ different slab sizes.
- Given *d* input orders with:
  - A *colour* (route through the mill).
  - A *weight*.
- Pack orders onto slabs such that the total slab capacity is minimised, subject to:
  - Capacity constraints.
  - Colour constraints.

# How Common Are Problems Involving Nesting?

- Example: Steel Mill Slab Design (CSPLib 38).
- **Capacity:**
  - Total weight of orders assigned to a slab cannot exceed slab capacity.
- **Colour:**
  - Each slab can contain at most $p$ of $k$ total colours.
  - Reason: expensive to cut slabs up to send them to different parts of the mill.

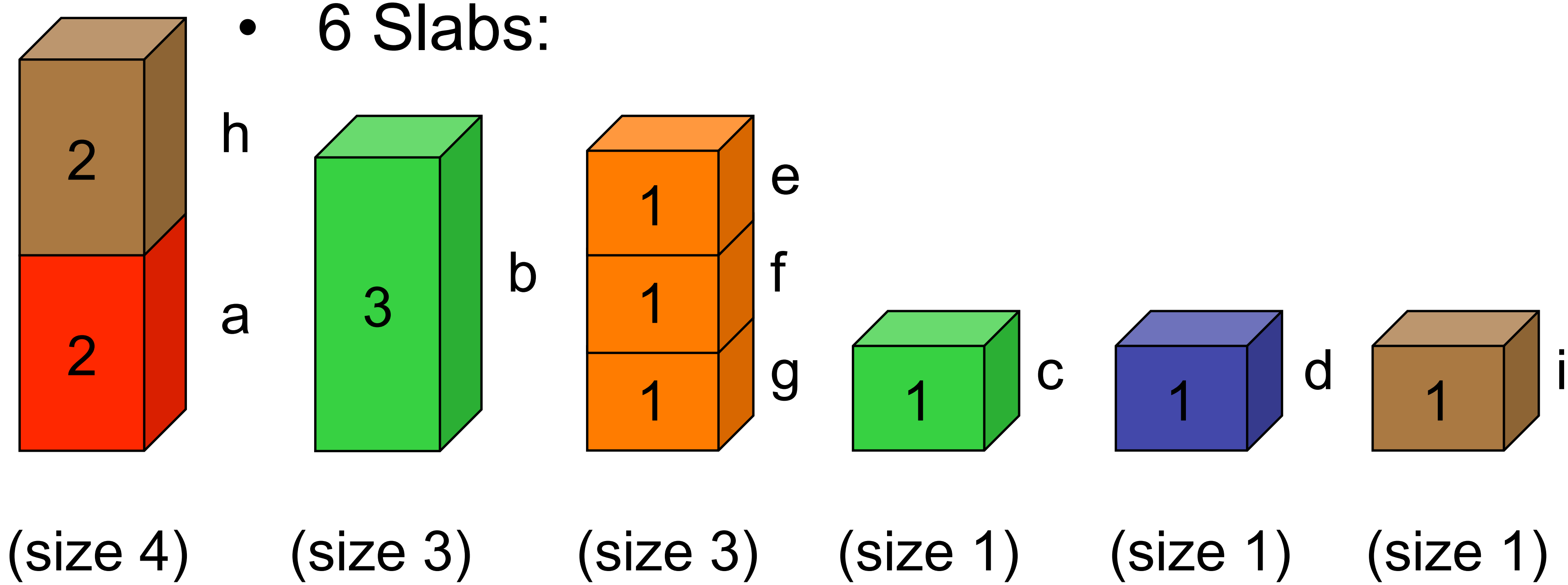
# How Common Are Problems Involving Nesting?

- Example: Steel Mill Slab Design (CSPLib 38).
- Slab Sizes: {1, 3, 4} ($\sigma$ = 3)
- Orders: {$o_a$, …, $o_i$} ($d$ = 9)
- Colours: {red, green, blue, orange, brown} ($k$ = 5)
- $p$ = 2



# How Common Are Problems Involving Nesting?

- Example: Steel Mill Slab Design (CSPLib 38).
- Slab Sizes: {1, 3, 4} ($\sigma = 3$)
- Orders: {$o_a$, …, $o_i$} ($d = 9$)
- Colours: {red, green, blue, orange, brown} ($k = 5$)
- $p = 2$
- 6 Slabs:



# How Common Are Problems Involving Nesting?

- Example: Steel Mill Slab Design (CSPLib 38).
- A slab can be represented as a **set** of orders.
- We must also determine the size of each slab.
- So this problem can be characterised as a **function from sets of orders to the set of sizes**.
- The function is **partial**, since not all possible sets of orders will be mapped to a slab size.

The Template design problem (CSPLib 2) can be characterised similarly.

# Nesting Inside Sequences

# Nesting Inside Sequences

- Recall how we modelled fixed-length sequences.
- 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 | … | 0..9 |

# Nesting Inside Sequences

- Assume now that we must find a sequence of sets, functions, relations, …
- We can no longer use a single variable at each index to represent the object at that position.
  - Because 1 variable is not enough to represent our set, function or relation.

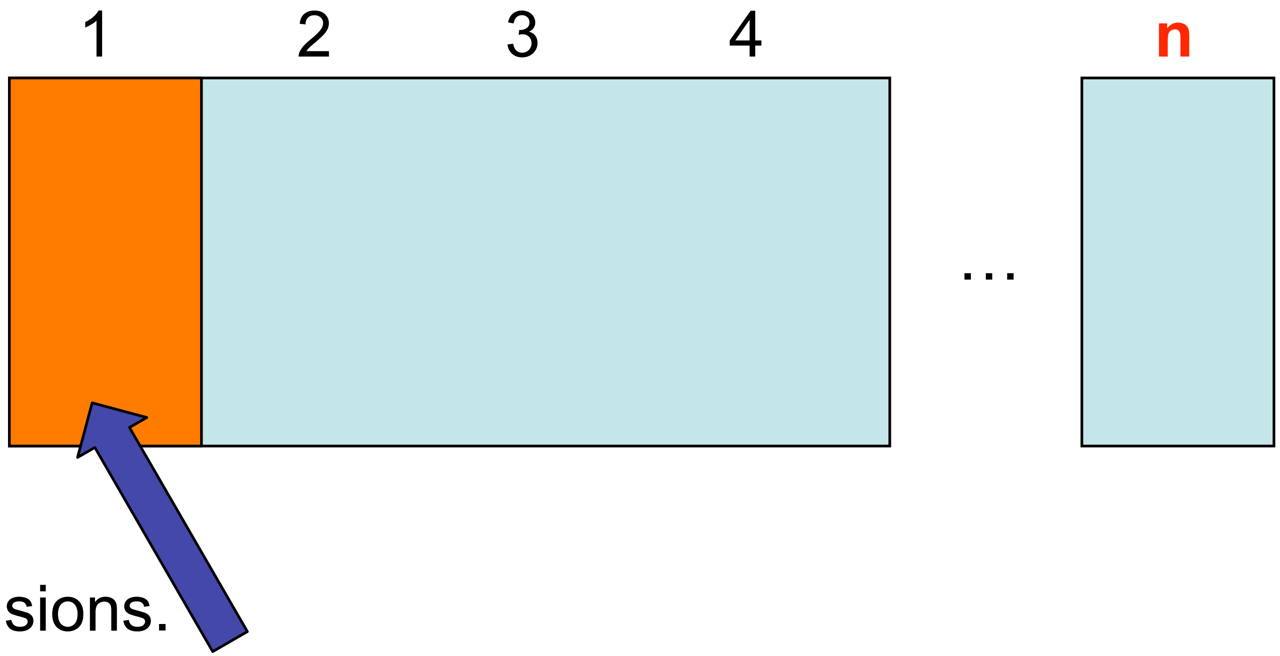
**NestedSeqArray**

| 1 | 2 | 3 | 4 | | **n** |
|---|---|---|---|---|---|
| ? | ? | ? | ? | … | ? |

# Nesting Inside Sequences

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

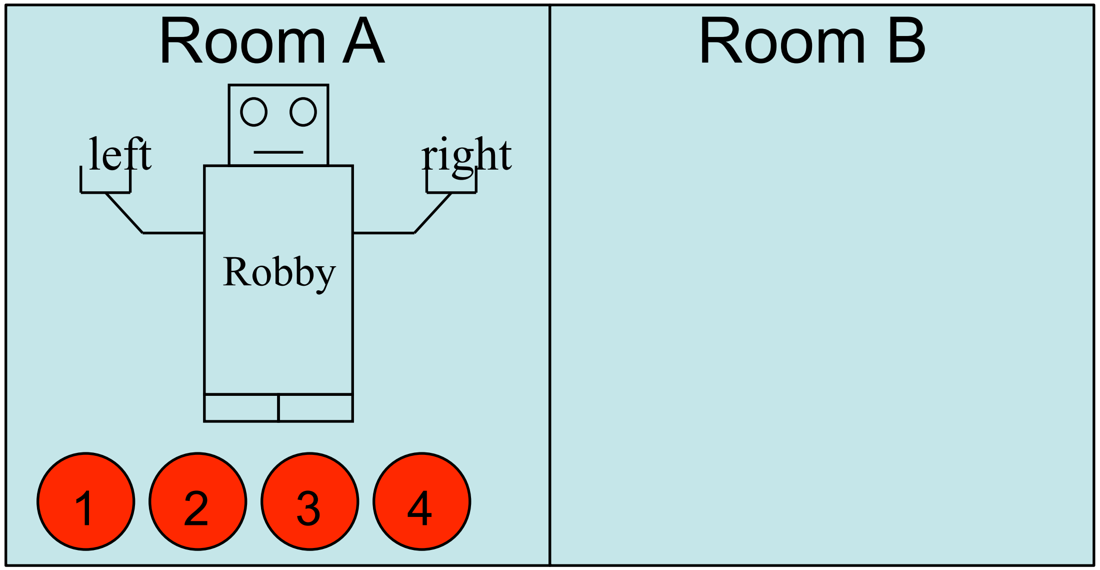
**NestedSeqArray**




E.g. 2 dimensions.
We now have a **column** of variables to
represent our set, relation, function …

# Nesting Inside Sequences

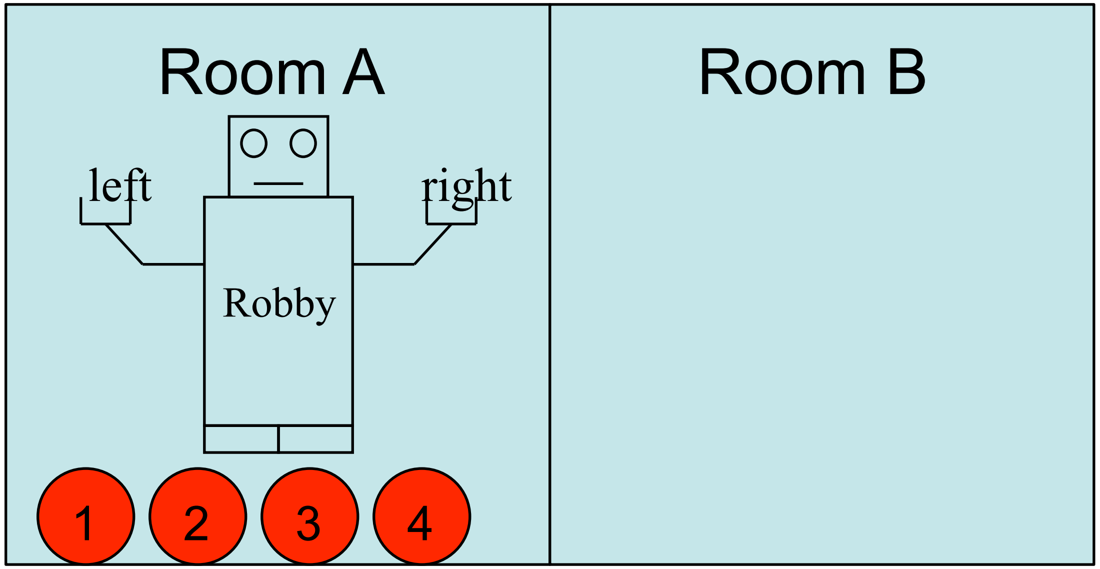
- To illustrate, consider modelling a sequence of sets.




- Returning to the Gripper problem, assume that we are looking for a plan of length **n**.

# Nesting Inside Sequences

- We saw before that a set of cardinality at most two can be used to model the actions performed at each step.




- Need elements for moving, pick up/drop balls with each of the two grippers.
- Assume we use integers 1..**k** to represent these actions.
- (I'm glossing over details here).

# Nesting Inside Sequences

- So, we have a sequence of length **n** of sets of cardinality at most 2 drawn from 1..**k**.
- Let's start by looking at the **occurrence** representation:

Each column represents the occurrence representation of a set. Constraints?

| | 1 | 2 | 3 | 4 | | **n** |
|---|---|---|---|---|---|---|
| 1 | 0, 1 | 0, 1 | 0, 1 | 0, 1 | … | 0, 1 |
| 2 | 0, 1 | 0, 1 | 0, 1 | 0, 1 | | 0, 1 |
| | | | … | | | … |
| **k** | 0, 1 | 0, 1 | 0, 1 | 0, 1 | … | 0, 1 |

# Nesting Inside Sequences

- So, we have a sequence of length **n** of sets of cardinality at most 2 drawn from 1..**k**.
- Now let's look at the **explicit** representation.

Each column represents the explicit representation of a set. Constraints?

| | 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 |

# Nesting Inside Sequences

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

**KisSequence**

| 1 | 2 | 3 | 4 |
|---|---|---|---|
| 0..n | 0..n | 0..n | 0..n |

# Nesting Inside Sequences

- We can use the same approach here (careful not to use the same dummy value as the explicit model of the inner sets).
- Could also use auxiliary **switch** variables to indicate whether the corresponding column is part of the sequence.
- Again, careful of introducing equivalence classes of assignments.

| | 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 |
| 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

Consider the following simple problem class:

- Given **m**, **n**.
- Find a cardinality-**m** **set of sets** of **n** digits such that …
- 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

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

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| **O** | 0, 1 | 0, 1 | 0, 1 | 0, 1 | 0, 1 | 0, 1 | 0, 1 | 0, 1 | 0, 1 | 0, 1 |

- We have an index per possible element of the set.

# Nesting Inside Sets: Occurrence

Can we take the same approach here?

- Given **m**, **n**.
- Find a cardinality-**m** **set of sets** of **n** digits such that…

Introduce an array indexed by the possible sets of **n** digits!

| {1, 2, 3} | {1, 2, 4} | {1, 2, 5} |
|---|---|---|
| 0,1 | 0,1 | 0,1 |

… (assuming **n** = 3)

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:

| | 1 | 2 | 3 | 4 | | n |
|---|---|---|---|---|---|---|
| **E** | 0..9 | 0..9 | 0..9 | 0..9 | … | 0..9 |

- Similarly to the sequence example, we extend the dimension of **E** according to the representation we choose for the inner set.
- We're also going to have to be careful to make sure the elements of the outer set are **distinct**.

# Nesting Inside Sets: Explicit/Occurrence

- Given **m**, **n**.
- Find a cardinality-**m** **set of sets** of **n** digits such that…
- Let's consider an occurrence representation for the inner sets.

**EO**

| | 1 | 2 | … | **m** |
|---|---|---|---|---|
| 0 | 0,1 | 0,1 | | 0,1 |
| 1 | 0,1 | 0,1 | … | 0,1 |
| 2 | 0,1 | 0,1 | | 0,1 |
| | | … | | … |
| 9 | 0,1 | 0,1 | … | 0,1 |

Constraints:

Sum(col i of **EO**) = **n**
(foreach *i* in 1..**m**)

Scalar-prod(col i of **EO**,
col j of **EO**) ≠ **n**
(foreach {*i*, *j*} in 1..**m**)

But what about **equivalence classes?**

# Nesting Inside Sets: Explicit/Explicit

- Given **m**, **n**.
- Find a cardinality-**m** **set of sets** of **n** digits such that…
- Let's consider an occurrence representation for the inner sets.

| **EE** | 1 | 2 | | m |
|---|---|---|---|---|
| 1 | 0..9 | 0..9 | | 0..9 |
| 2 | 0..9 | 0..9 | … | 0..9 |
| 3 | 0..9 | 0..9 | | 0..9 |
| | | … | | … |
| n | 0..9 | 0..9 | … | 0..9 |

Constraints:

Col i of **EE** $<_{lex}$ Col j of **EE** $\vee$
Col i of **EE** $>_{lex}$ Col j of **EE**
(foreach $\{i, j\}$ in 1..**m**)
AllDiff on columns.

But what about **equivalence classes?**

# Relations as Sets of Tuples

- Last time we looked at a couple of ways of modelling relations.
- 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:

| $\langle 1,2\rangle$ | $\langle 1,3\rangle$ | $\langle 1,4\rangle$ | |
|---|---|---|---|
| 0,1 | 0,1 | 0,1 | … |

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

| | | A | | |
|---|---|---|---|---|
| | | 1 | 2 | 3 |
| | 2 | 0, 1 | 0, 1 | 0, 1 |
| B | 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 our bounded-cardinality set pattern:

| | 1 | 2 | 3 | 4 | … |
|---|---|---|---|---|---|
| 1 | 1..3 | 1..3 | 1..3 | 1..3 | |
| 2 | 2..4 | 2..4 | 2..4 | 2..4 | |

What about **equivalence classes**?
What if the relation allows fewer than the full 9 tuples?

# The Social Golfers Problem

# The Social Golfers Problem

- 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.

# The Social Golfers Problem: Modelling

- In each week, we need to **partition** the golfers into groups.
  - A partition is a set of sets. No pair of inner sets have an element in common.
- 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**.

# Golfers: Representing the Outer Multiset

- We have seen explicit and occurrence representations of multisets.
- The multiset contains complex objects (partitions).
- Indexing an array by the possible partitions of golfers doesn't seem appealing.
- So let's try an explicit model:

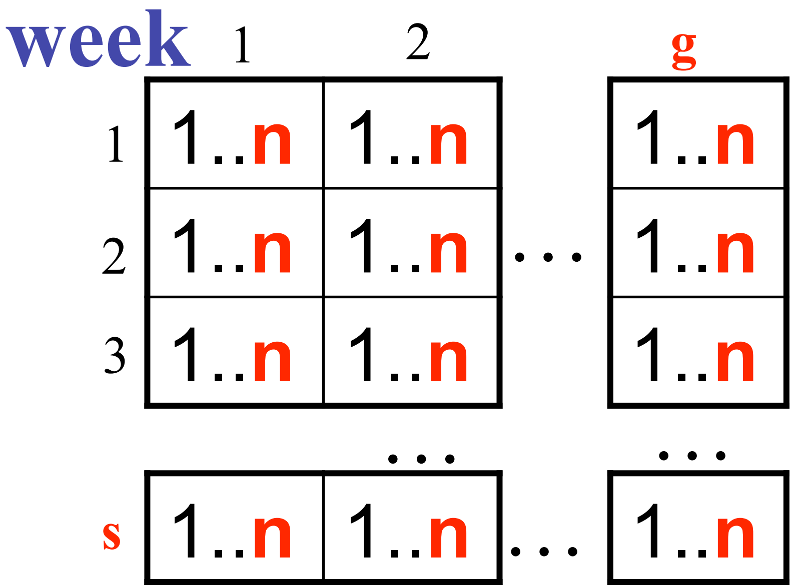
| 1 | 2 | 3 | 4 | | w |
|---|---|---|---|---|---|
| ? | ? | ? | ? | … | ? |

# Golfers: 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 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.

# Golfers: The Partitions

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

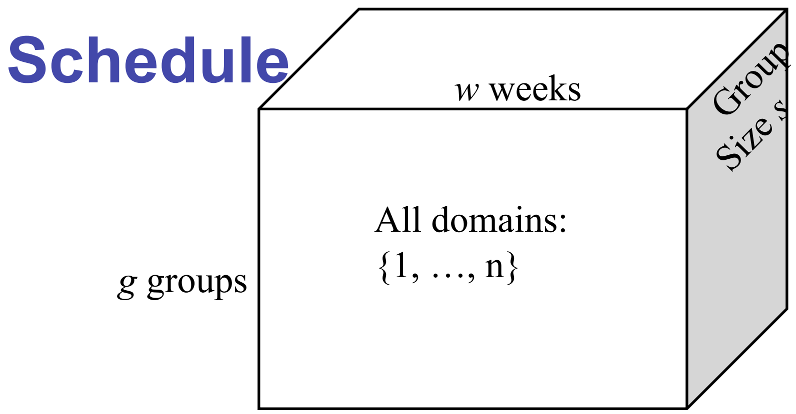



Since a week is a partition, what can we say about the elements of **week**?

What about **equivalence classes**?

# A Multiset of Partitions of Golfers

- If we put **week** into each slot of our multiset representation, we obtain a 3d array:

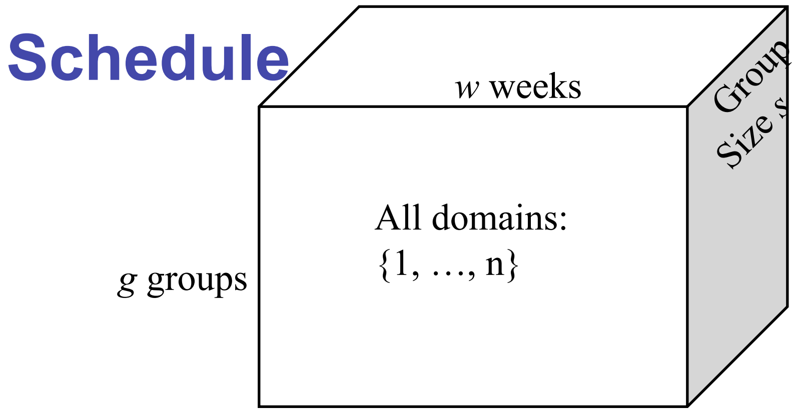



NB n = g x s is no of golfers

We can **order the weeks** lexicographically to counter the equivalence of assignments obtained by permuting the weeks.

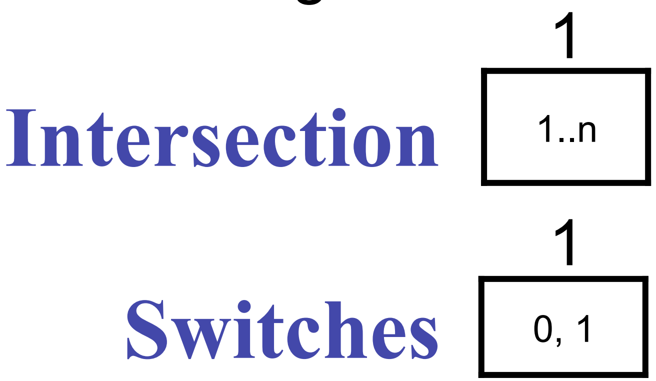
# A Multiset of Partitions of Golfers

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




NB $n = g \times s$ is no of golfers

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




Sum of switches is at most 1

# Social Golfers

- Solution to the instance with 3 groups (size 3) over 3 weeks:

| | 3 groups, size 3 | | |
|---|---|---|---|
| 3 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] |

We've missed an equivalence class! Can you spot it?
Hint: we saw something similar in the BIBD.

# 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 cheaper than trying to detect them after the fact.

And Finally:

# The Golomb Ruler Challenge

# The Golomb Ruler Problem

- NB This is a type of **Graceful Graph**.
- Given:
  - A positive integer ***n***.
- Find:
  - A set of ***n*** integer ticks on a ruler of length ***m***.
- Such that:
  - All inter-tick distances are distinct.
- Minimising:
  - ***m***.

# Modelling the Golomb Ruler

| | 1 | 2 | 3 | 4 | | **n** |
|---|---|---|---|---|---|---|
| **T** | 0..**n**$^2$ | 0..**n**$^2$ | 0..**n**$^2$ | 0..**n**$^2$ | … | 0..**n**$^2$ |

- All inter-tick distances are distinct:
  - **T**$[j]$ – **T**$[i] \neq$ **T**$[k]$ – **T**$[l]$
    for each $\{i, j\}$, $\{k, l\}$ drawn from 1..**n**,
    such that $\{i, j\} \neq \{k, l\}$, $i < j$, $k < l$
    again, exploiting ascending order.
- Objective:
  - Minimise(**T**[**n**])
    Again, exploiting ascending order.

# Modelling the Golomb Ruler

- **A Challenge**:
- Can you see how to model this problem using the occurrence representation?
- This does require a little sleight of hand…

# Golomb Ruler: Occurrence Model

- Recall that in our explicit model, the elements of the set are 0..$n^2$.

| | 1 | 2 | 3 | 4 | | n |
|---|---|---|---|---|---|---|
| T | 0..$n^2$ | 0..$n^2$ | 0..$n^2$ | 0..$n^2$ | … | 0..$n^2$ |

- Invoking our occurrence representation pattern, we begin with an array **O** indexed 0..$n^2$:

| | 0 | 1 | 2 | 3 | | $n^2$ |
|---|---|---|---|---|---|---|
| O | 0,1 | 0,1 | 0,1 | 0,1 | … | 0,1 |

# Golomb Ruler: Occurrence Model

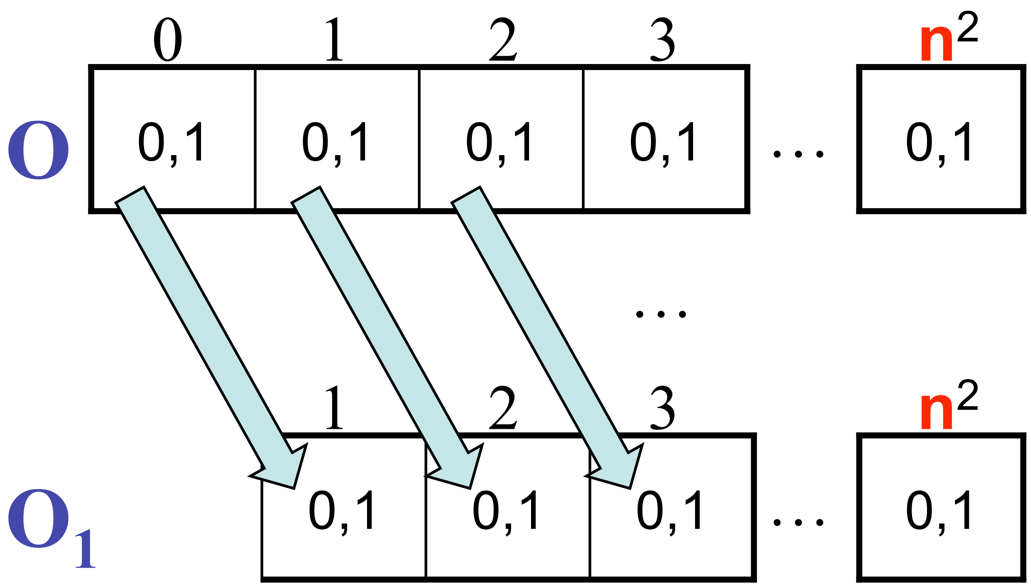
- How can we express the distinct distances constraint?
- Consider a partial assignment:

| | 0 | 1 | 2 | 3 | | $n^2$ |
|---|---|---|---|---|---|---|
| **O** | 1 | 1 | 0,1 | 0,1 | … | 0,1 |

- We now know that no other pair of adjacent variables can be assigned 1.
- How can we express these constraints?

# Golomb Ruler: Occurrence Model

- Consider an array $O_1$, which contains the **same** variables as $O$, shifted one position right.



# Golomb Ruler: Occurrence Model

- Now let’s assign some variables:

| | 0 | 1 | 2 | 3 | | $n^2$ |
|---|---|---|---|---|---|---|
| **O** | 1 | 1 | 1 | 0,1 | … | 0,1 |

| | 1 | 2 | 3 | | $n^2$ |
|---|---|---|---|---|---|
| $\mathbf{O_1}$ | 1 | 1 | 1 | … | 0,1 |

Scalar product: 2

- Whereas:

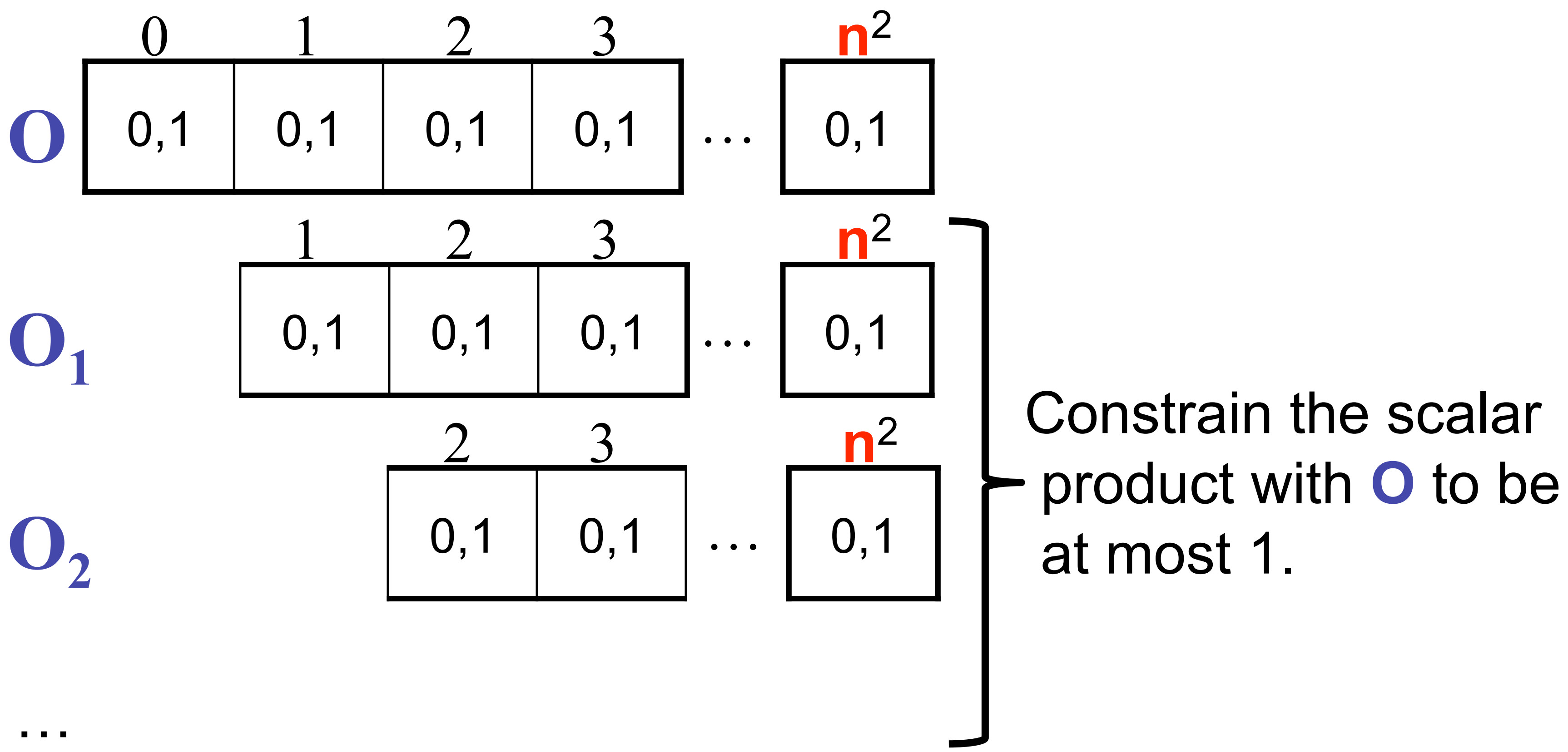
| | 0 | 1 | 2 | 3 | 4 | | $n^2$ |
|---|---|---|---|---|---|---|---|
| **O** | 1 | 1 | 0,1 | 1 | 0, 1 | … | 0,1 |

| | 1 | 2 | 3 | 4 | | $n^2$ |
|---|---|---|---|---|---|---|
| $\mathbf{O_1}$ | 1 | 1 | 0,1 | 1 | … | 0,1 |

Scalar product: 1

# Golomb Ruler: Occurrence Model

- Now consider adding one such array per difference:



# Golomb Ruler: Occurrence Model

- (Perhaps) you're thinking:
  - That's a lot of extra variables!
- In fact, we've introduced **no** extra variables.
- Just re-used **existing** variables in new arrays.

# Golomb Ruler: Occurrence Model

- But what about the objective?

| | 0 | 1 | 2 | 3 | | $n^2$ |
|---|---|---|---|---|---|---|
| **O** | 0,1 | 0,1 | 0,1 | 0,1 | … | 0,1 |

- Tricky because we are trying to minimise the index of the last "1" assignment.
- One way is to solve a series of problems, increasing the size of **O**.
  - As soon as we have a solution, it is optimal.

# Golomb Ruler: Discussion

- Most constraint models of this problem for the literature focus on the explicit representation of the set.
- Build on this model by adding **auxiliary variables** and **implied constraints**.
  - Barbara M. Smith, Kostas Stergiou, Toby Walsh: Using Auxiliary Variables and Implied Constraints to Model Non-Binary Problems. AAAI/IAAI 2000: 182-187
- Distributed effort to find large GRs looks more like this occurrence model.
  - The power of bit-shifting.