Modelling for Combinatorial Optimisation (1DL451) and Constraint Programming (1DL442)
Uppsala University – Autumn 2024
MiniZinc Exercises (optional)

Prepared by Gustav Björdal and Pierre Flener

Version of 16th August 2024

Comprehensions are a crucial part of efficient MiniZinc models. Here are exercises to help you get started with writing complex comprehensions in the assignments and project. For a more basic introduction to comprehensions, see Section 2.2.1 of the MiniZinc Handbook at `https://www.minizinc.org/doc-latest/en/modelling2.html`. Skeleton code for the exercises is at `https://pierre-flener.github.io/courses/COCP/assignments/exercises`.

1 Repeating Numbers

Consider an array `A` of `m` parameters, and a parameter `n`. Define, using an array comprehension, the array `B = [A[1], A[1], ..., A[2], A[2], ..., A[m], A[m],...]` of length `m*n` containing `n` repetitions of each element of `A`.
Example: `A = [1,2,1]` and `n = 2` should give `B = [1,1,2,2,1,1]`.

Listing 1: Skeleton code for Ex1.mzn

```
int: n = 2;
int: m = 3;
array[1..m] of int: A = [1,2,1];

array[1..m*n] of int: B = [... | ...];
```

2 Repeating Numbers Again

Consider an array `A` of `m` parameters, and a parameter `n`. Define, using an array comprehension, the array `B = [A[1], A[2], ..., A[m], A[1], A[2], ... A[m], ...]` of length `m*n` consisting of `n` concatenations of `A`.
Example: `A = [1,2,3]` and `n = 3` should give `B = [1,2,3,1,2,3,1,2,3]`.

Listing 2: Skeleton code for Ex2.mzn

```
int: n = 3;
int: m = 3;
array[1..m] of int: A = [1,2,3];

array[1..m*n] of int: B = [ ... | ... ];
```

3 Counting

Consider a 2D array A of n rows and m columns of parameters. Define, using an array comprehension, the array B of length n where at each index i in B is the number of occurrences of the value 5 in row i of A.
Example: `A = [|1,2,3|4,5,6|7,8,9|5,5,5|]` should give `B = [0,1,0,3]`.

Listing 3: Skeleton code for Ex3.mzn

```
int: n = 4;
int: m = 3;
array[1..n, 1..m] of int: A = [|1,2,3|4,5,6|7,8,9|5,5,5|];

array[1..n] of int: B = [ ... | ... ];
```

4 Expression on Variables

Consider an array X of n variables in the domain `1..n*n`. Define, using an array comprehension, the array Y of variables denoting the absolute differences in each unordered pair of separate variables in X. Define also the index set of Y (or use the keyword `int`). By constraining both X and Y to each have all-different values, we get a model for the Golomb ruler (see https://mathworld.wolfram.com/GolombRuler.html).
Example: If `X = [8,4,2,1]`, then Y should contain the values $4, 6, 7, 2, 3, 1$, in some order.

Listing 4: Skeleton code for Ex4.mzn

```
include "globals.mzn";

int: n = 4;

array[1..n] of var 1..n*n: X;
array[...] of var int: Y = [... | ...];

constraint all_different(X);
constraint all_different(Y);
```

5 Sorting

Consider two arrays A and B of n parameters. Define, using an array comprehension, the array C containing the elements of B but sorted as if it was A. That is `B[i]` occurs before `B[j]` in C if and only if `A[i] <= A[j]`. **Hint:** Use `arg_sort` in the generator of the comprehension.
Example: `A = [2,1,3]` and `B = [9,7,5]` should give `C = [7,9,5]`.

Listing 5: Skeleton code for Ex5.mzn

```
include "globals.mzn";
int: n = 3;
array[1..n] of int: A = [2,1,3];
array[1..n] of int: B = [9,7,5];

array[1..n] of int: C = [... | ...];
```

6 2D Array

Consider an integer parameter `n`. Define, using an array comprehension, the 2D array `A` with 3 columns, the rows being `[f,s,f*n+s]` for each `f` and `s` with `1 <= f < s <=n`.
Hint: Use `array2d` in order to cast a generated 1D array into a 2D one.
Example: `n=4` should give `[|1,2,6|1,3,7|1,4,8|2,3,11|2,4,12|3,4,16|]`.

Listing 6: Skeleton code for Ex6.mzn

```
int: n = 4;

array[...,...] of ...: A = ...;
```