# Example exam

**Q1: Small questions**

# **1. What is consistency?**

Adapting the domain of variables of a constraint by using a propagator to remove impossible values from it to make it consistent with the constraint

Eg domain consistency (removing all impossible values) or bounds consistency (removing only min and max values from domain)

Is there even a somewhat clear definition of this? It pops up sometimes but i wouldn’t know what to say

**2. What is a redundant constraint?**

Redundant or implied constraint is a constraint that logically follows from other constraints.  
Adding such constraint might improve solving time (smaller search space), but sometimes it’s just overhead

**3. Explain what the difference is between partial and full value symmetry.**

Q2 is just a full symmetry LOL. If task {t1, t2…} can be only done by specific workers {w1, w2…} it becomes a partial symmetry.

=> isn’t that variable symmetry instead of value symmetry?

-> Isn’t it both?

Var symmetry: we can change the tasks (= swap indices) while maintaining valid solutions

Val symmetry: ?

Don’t know the second one?

-> This is a permutation problem; meaning variables and values can be swapped

Thanks!

**Q2: n workers & n tasks**

# **Consider the following simple assignment problem. It involves n workers and n tasks. Every worker should be assigned to exactly one task and every task should be assigned to exactly one worker. The aim is to maximize the profit (given by the profit array: profit[w,t] is the profit when worker w is doing task t). Consider the following partial MiniZinc model.**

***int: n;***

***set of int: Tasks = 1..n;***

***set of int: Workers = 1..n;***

***array[Workers,Tasks] of int: profit;***

**This assignment problem is known to be a permutation problem.**

**1. What is a permutation problem?**

T4 - slide 7

Nb of values equal to nb of decision vars and each variable takes an unique value.

Two VP can be constructed from this namely <X,D> and <D,X> with X the set of decision variables and D the domain of the variables.

**2. Give 2 viewpoints for the above problem and complete the partial MiniZinc model for each of the two viewpoints**

Viewpoint 1: Assign tasks to workers

*% task task[i] is executed by worker i*

*array[Workers] of var Tasks: task;*

*constraint alldifferent(task);*

*solve maximize sum (w in Workers) (profit[w, task[w]])*

Viewpoint 2: Assign workers to tasks

*% worker worker[i] has task i*

*array[Tasks] of var Workers: worker;*

*constraint alldifferent(worker);*

*solve maximize sum (t in Tasks) (profit[worker[t], t])*

# **3. What is a channeling constraint?**

A channel constraint establishes the coherence of the values of mutually redundant decision variables (e.g., the inverse constraint)

# **4. Give the channeling constraint for your two viewpoints.**

# constraint inverse(worker, task) or, constraint forall (w,t in 1..n) (worker[t] = w <-> task[w] = t)

# JUNE 2021 - A

**Q1: Small questions**

1. **What is a constraint propagator?**

(T8 slide 12)

A constraint propagator for a predicate x removes from the current domain of variables of a x-constraint values that cannot be part of a solution to that constraint   
This is also a super vague question? The definition from the slides doesn’t even seem like a legit sentence?  
A constraint propagator removes values from the current domain that cannot be part of the solution, it maintains consistency because it still satisfies all the constraints + it enforces constraints by pruning values that are inconsistent with given constraints -> by propagation alone, it’s possible to find solutions without having even to search (in general, it narrows down the search space)

1. **What is column symmetry, give a minizinc model as an example.**

(T5 slide 19 & 29 & 34 & 36, T8 slide 31)

n\_queen problem is a column symmetry problem as shown below.

constraint symmetry\_breaking\_constraint(

let {

array[1..n,1..n] of var bool: qb;

} in

forall (i,j in 1..n) ( qb[i,j] <-> (q[i]=j) )

/\

lex\_lesseq(array1d(qb), [ qb[i,j] | i in reverse(1..n), j in 1..n ])

);

=> symm breaking constraint is not asked right?

Instead: The problem has inherent symmetry. That is, for any solution we obtain another solution by any of the 8 symmetries of the chessboard (including the identity) obtained by combinations of rotations by 90 degrees and reflections, this boils down to permutations of the columns.

Slide 9: nqueens doesn’t have column symmetry: only reflection and rotation symmetry.

-> Is reflection symmetry a type of column symmetry?

I think column symmetry means permuting the columns of a valid solution will yield another valid solution. In this point of view, nQueens holds column symmetry under the following situation: use array to sequentially assign column permutations to every queen in each row. It can also considered as a row symmetry (sequentially assign row permutations to every queen in each column)

1. **What is global consistency? show how global consistency evolves in the following problem(An example): x,y,z have the global consistency on their value domain, 1..9**

***x in 1..9, y in 1..9, z in 1..9***

***x+3y<=7***

***2z+3x<=8***

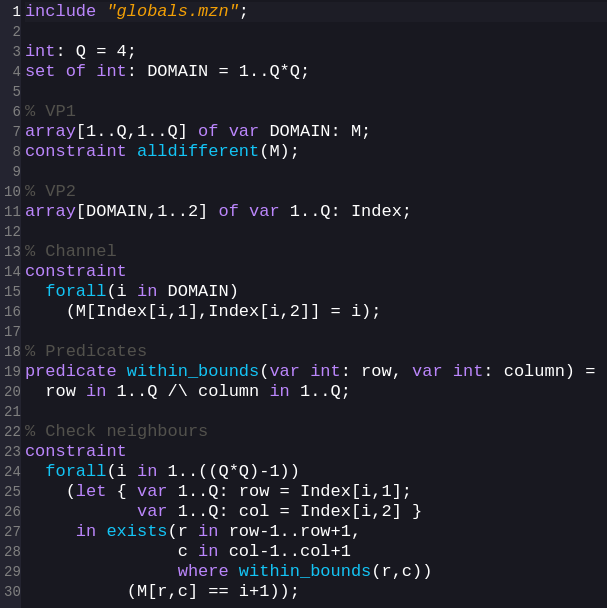
***3x+2y+z<=15***

(T8 slide 12) => What is global consistency??

Global consistency takes multiple constraints into account (all constraints in the problem?). While domain or bounds consistency are applied on one (or a set of) specific constraint(s)

**Q2: A constraint programming question**

**Given a Q\*Q matrix M, the value domain is 1..Q\*Q, design a minizinc model to find a solution such that, every M[i,j] value should have its adjacent value at its neighbor. e.g., M[3,3]=15, then number 14 or 16 should be at one of the following position: M[2,2],M[2,3],M[2,4],M[3,2],M[3,4],M[4,2],M[4,3],M[4,4].**

1. **Give the decision variables and parameters in minizinc.**
2. **Define the constraints and objective function.**

include *"globals.mzn"*;

int: q = 4;

array[1..q\*q] of var 1..q\*q: M;

array[1..q\*q] of 1..q: row = [((i - 1) div q) + 1| i in 1..q\*q];

array[1..q\*q] of 1..q: col = [((i - 1) mod q) + 1 | i in 1..q\*q];

constraint forall(i in 1..q\*q) (exists(o in

1..q\*q) (

((M[i] = M[o] - 1) \/ (M[i] = M[o] + 1)) /\

((row[i] = row[o] - 1) \/ (row[i] = row[o] + 1)) /\

((col[i] = col[o] - 1) \/ (col[i] = col[o] + 1))

));

constraint alldifferent(M[1..q\*q]);

solve satisfy;

—-------------------

Found the following to be an easier solution  
include *"globals.mzn"*;

int:Q=5;

set of int:dom = 1..Q;

array[dom,dom] of var 1..Q\*Q: m;

constraint

forall(i,j in dom)(

exists(x in i-1..i+1 where x >0 /\ x <Q+1, y in j-1..j+1 where y > 0 /\ y < Q+1)(

abs(m[x, y] - m[i, j]) = 1

)

)

;

constraint

all\_different(m);

solve satisfy;

1. **What is the viewpoint? Find two viewpoints for your model.Z**

Map the values to the position in the matrix (VP1) and the position to the values in (VP2)

1. **What is a channeling constraint? Define a channeling constraint in your model.**

A channeling constraint establishes coherence of values of mutually redundant decision variables

The domain of one VP corresponds to the positions of the other, we can use something to the likes of:

% Channelling constraint

constraint forall (p in DOMAIN, i,j in 1..Q) (

(M[Index[p,1],Index[p,2]] = p);

**Q3, Another non-trivial programming question**  
**Given a grid shown as Matrix Q\*P, a square in the grid could be a not passable square, a square with reward values from [1..5],or an empty square. The task is, given a number of the steps user can take, to find the longest path in the grid which have the biggest accumulated reward value, the path should form a loop. The layout of the combination of arrays grid is given already.**

1. **Give the decision variables and parameters in minizinc.**
2. **Define the constraints and objective function.**
3. **Is there a symmetry to break? If yes, define the symmetry breaking constraint.**
4. **Could we use Precomputing to optimize the minizinc model, show how to do that in your minizinc code.**
5. **How to accelerate the solving using heuristics. explain your idea.**

**1+2:**

include *"globals.mzn"*;

int: Q = 2;

int: P = 3;

set of int: rows = 1..Q;

set of int: cols = 1..P;

set of int: cells = 1..Q\*P;

int: not\_passable = 6;

set of int: val = 0..6;

array[rows, cols] of val: grid = [|2, 0, 6

|4, 1, 0|];

array[cells] of var cells: path;

function var int: row(var int: x) = ((x-1) div P)+1;

function var int: col(var int: x) = ((x-1) mod P)+1;

var int: reward = sum(i in cells where path[i] != i) (grid[row(i), col(i)]);

solve maximize reward;

constraint

subcircuit(path)

/\ forall (i in cells) (path[i] != i -> grid[row(i), col(i)] != 6)

/\ forall (i in cells) (path[i] = i \/ path[i] = i+1 \/ path[i] = i-1 \/ path[i] = i+P \/ path[i] = i-P);

### 

**3:** Symmetry: inverse of path

* Rotation symmetry (if start position is not locked)?

**4:** (T2 slide 43,T8 slide 43) Precompute all rows and columns for each cell index

array[1..P\*Q] of 1..P: R = [1,1,1,2,2,2]

array[1..P\*Q] of 1..P: C = [1,2,3,1,2,3]

**5:**

# JUNE 2021 - B

**Q1: Small questions**

1. **What is the propagator in a constraint**

A propagator in a constraint for a predicate x removes from the current domains of the variables of a x-constraint values that cannot be part of a solution to that constraint

1. **What is domain consistency**

(T8)  
Domain consistency is obtained by applying a domain-consistency propagator, deleting all impossible values from the current domains of variables of a constraint

1. **Given the constraint 2y+X = 2w -4 with the domains 1..10 give the domains after domain consistency.**

(T8) -> Lot of work no?

Run 2y+x=2w-4: prune unsupported values of y

Possible for y until 7 => prune y = 8,9,10

Run 2y+x=2w-4: prune unsupported values of x

Impossible for odd numbers => prune all odd x’s

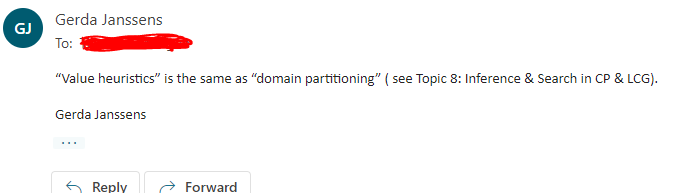
Run 2y+x=2w-4: prune unsupported values of w

Prune w= 1,2,3

Run 2y+x=2w-4: prune unsupported values of y

…

Just to be clear, we stop here right or are we supposed to find more values to prune?

1. **Give a Def for at-most(c,A,v) where in array A there are at most c variables with the value v**

predicate at\_most(int: c, array[int] of var int: A, int: v) =

count(A,v) <= c;

Can A and v not take any type?

1. **Does at most have symmetry**

Yes, array A can be permuted and still be a solution (?)

* I think it depends on what you see as a symmetry in this case. If a symmetry class exists of all candidate solutions leading to the same boolean result of the predicate -> yes, a lot of symmetries

1. **What is the difference between partial and full value symmetry?**

In full value symmetry, all permutations preserve solutions (Full symmetry group S\_n has n! symmetries over a sequence of n elements)  
In partial symmetry, only piecewise permutations preserve solutions.

1. **Illustrate with an example.**

An example of full symmetry: the colours in a graph colouring problem are all interchangeable, since the exact colour doesn’t matter for the problem.

An example of partial symmetry: suppose we are looking for integers a,b,c,d such that a^3+b^3=c^3+d^3. Then we can interchange a and b (and c and d), but we cannot interchange a and c without losing our solution for instance.

=> isn’t your first example value symmetry and the second variable symmetry?

I think so. But I suppose full/partial symmetry can occur in both value and variable symmetry.

Value: Normal <-> Partitioned map colouring

Variable: a^2+b^2+c^2 = 100 <-> a^2+b^2=c^2

You could also just use the first problem ( colouring ) with the different colours partitioned into subsets for partial symm, no? (eg slide 8, slide 27 of topic 5)

**Q2: Depots. Given set of depots, the coordinates, pickups, and dropoffs.**

1. **Give 1d decision variable.**

Lack of details to solve

1. **Complete model.**
2. **Give second viewpoint decision variable**
3. **What is channeling constraint**
4. **Solve with either second viewpoint or a combination of the two.**

**Q3: Customers suppliers tables and dinners. Give a schedule with the constraint that each s and c must sit together exactly once. S and s can sit together maximum once.**

1. **Give parameters**
2. **Give decision variables**
3. **Complete model**

int: s; % customers

int: t % tables

int: d % dinners

% constraint: each s and c must sit together once

What is a dinner? Is it an dinner event? Is it a dining place? Is it one single dinner where s and c can sit together?

1. **Give symmetry breaking constraint**
2. **Create a redundant constraint explain**
3. **Which variables and which heuristic value work the best explain.**

What are they referring to here? Heuristic value?