



THE UNIVERSITY OF

MELBOURNE

Library Course Work Collections

Author/s:

Computing and Information Systems

Title:

Constraint Programming, 2015 Semester 2, COMP90046

Date:

2015

Persistent Link:

<http://hdl.handle.net/11343/91167>

THE UNIVERSITY OF MELBOURNE
DEPARTMENT OF COMPUTING AND INFORMATION SYSTEMS
SEMESTER 2 ASSESSMENT 2015

COMP90046 Constraint Programming

TIME ALLOWED: 3 HOURS
READING TIME: 15 MINUTES

Authorized materials: Books and calculators are not permitted.

Instructions to Invigilators: One 21 page script. Exam paper may leave the room.

Instructions to students:

This exam counts for 70% of your final grade. There are 8 pages and 8 questions for a total of 70 marks. Attempt to answer all of the questions. Values are indicated for each question and subquestion — be careful to allocate your time according to the value of each question.

This paper should be reproduced and lodged with the Baillieu Library

Question 1 [4 marks]

How many integers between 1 and 200 have digits that sum up to 9? The answer is 17 being {9, 18, 27, 36, 45, 54, 63, 72, 81, 90, 108, 117, 126, 144, 153, 162, 171, 180}

We consider the generalization of this problem: how many integers between l and u sum up to s .

- (a) Write a MiniZinc model that, given l , u and s , determines a number n which is in the range $l..u$ and whose digits sum to s .

You can assume that l , u and s are all positive integers. The data format should be

```
int: l;  
int: u;  
int: s;
```

For example the data for the problem above is

```
l = 1;  
u = 200;  
s = 9;
```

Ensure that any decision variables you define have tight upper and lower bounds. To help you, given a number k the number of digits d in k is given by

```
int: d = ceil( ln(k+1) / ln(10) );
```

e.g. for $k = 1000$, $d = 4$ and for $k = 999$, $d = 3$, and in MiniZinc exponentiation is written as `pow(x , y) = x^y` . The model should output n . [3 marks].

- (b) Explain how you can use your model to determine the number of integers between l and u that add up to s . [1 mark].

Question 2 [8 marks]

In 1779 Euler proposed the following “36 Officer Problem”: given 6 regiments each with 6 officers of 6 different ranks, is it possible to arrange the 36 officers in a square of 6×6 , so that no row or column contains a duplicate rank or regiment. The mathematician Gastor proved it impossible in 1901.

We consider a generalization of the problem: given n regiments each with n officers of n different ranks, is it possible to arrange the n^2 officers in a $n \times n$ square so that no row or column contains a duplicate rank or regiment. It turns out for many n this is possible. Here is a solution for $n = 4$, shown as (regiment, rank) pairs. Note how no regiment or rank is duplicated in any row or column and each pair occurs exactly once.

(4,1) (3,2) (1,3) (2,4)
(2,2) (1,1) (3,4) (4,3)
(3,3) (4,4) (2,1) (1,2)
(1,4) (2,3) (4,2) (3,1)

- (a) Write a MiniZinc model that given n determines a solution to the $n \times n$ officer problem, if possible. The data format is simply

```
int: n;
```

For example the data for the example above is

```
n = 4;
```

Ensure that any decision variables you define have tight upper and lower bounds. Note you do not have to provide an output item! [6 marks].

- (b) Since the regiments and ranks are interchangeable, there are significant symmetries in the problem. Illustrate a symmetry breaking constraint that could be added to your model to improve the solving behavior. Explain why it is correct! [2 marks].

Question 3 [7 marks]

In a steel mill there are a number of tasks to schedule on a single roller. Each task has a required duration in minutes. There are two kinds of tasks: “hot” tasks and “cold” tasks. Assume data is given in the form:

```
int: n;                                % number of tasks
set of int: TASK = 1..n;
array[TASK] of int: duration;          % duration of each task (mins)
set of TASK: cold;                     % which tasks are cold
set of TASK: hot = TASK diff cold;    % which tasks are hot
int: makespan;                         % total rolling time available
```

The principal decision variables are

```
array[TASK] of var 0..makespan: start; % start time of each task
var 0..makespan: endtime;              % when rolling finishes
```

- (a) Give a MiniZinc model to ensure that all tasks are finished before `endtime` and no two tasks overlap in the schedule, and the `endtime` is minimized. There is no need to repeat the above declarations. [3 marks].
- (b) A further constraint is that a “cold” task cannot be performed on the roller immediately after a “hot” task. If the previous task to a “cold” task is “hot” then we must wait at least 30 minutes after the “hot” task ends before we can start the “cold” task, to let the roller cool. Give additions to the MiniZinc model to ensure the “hot”/”cold” constraint is not violated. [4 marks].

Question 4 [10 marks]

- (a) If the solver used does not support a constraint directly it needs to be decomposed into simpler constraints. Here are two predicate definitions for the constraint `mymin(x, y, z)` which encodes $x = \min(y, z)$.

```
predicate mymin(var int: x, var int: y, var int: z) = %%(A)
    x <= y /\ x <= z /\ (x = y \/ x = z);
predicate mymin(var int: x, var int: y, var int: z) = %%(B)
    (y <= z /\ x = y) \/ (y > z /\ x = z);
```

Explain which decomposition is likely to be better and why? [2 marks].

- (b) A strict valley in a sequence of numbers is a element of the sequence which is strictly less than both its two neighbors. End points of the sequence can never be strict valleys. For example the sequence [1,1,4,8,2,7,3,3,6,1] has one strict valley at the 5th position e.g. 8,2,7. The combinatorial function

```
function var int: strictvalley(array[int] of var int: x)
```

returns the number of strict valleys in the sequence x . Give a MiniZinc function definition for `strictvalley`. [3 marks].

- (c) Explain if the function definition you gave above can be used in a negative context, and why this is the case. [1 mark].
- (d) In order to run a MiniZinc model using a MIP solver, all variables and constraints must be eventually mapped to integer variables and linear integer constraints. In one sentence each briefly informally describe how MiniZinc maps
- Boolean variables: `var bool: b`
 - fixed array lookup: `x = a[y]`, where a is a fixed array of ints.
 - global constraints: e.g. `alldifferent`
- [4 marks].

Question 5 [7 marks]

Consider the following partial model for a nurse rostering problem

```
int: k; % number of nurses
set of int: NURSE = 1..k;
int: m; % number of days
set of int: DAY = 1..m;
set of int: SHIFT = 1..3;
int: day = 1; int: night = 2; int: dayoff = 3;
int: o; % nurses required on day shift
int: l; % lower bound for nurses on nightshift
```

```

int: u; % upper bound for nurses on nightshift

array[NURSE, DAY] of var SHIFT: x;

constraint forall(n in NURSE, d in 1..m-2)
    ( x[n,d] = night /\ x[n,d+1] = night -> x[n,d+2] = dayoff /\
      x[n,d] = night -> x[n,d+1] != day );
constraint forall(n in NURSE)
    ( x[n,m-1] = night -> x[n,m] != day );

solve satisfy;

```

The model has constraints to ensure that after two night shifts in a row, a nurse must have a dayoff, and that directly after a night shift, a nurse cannot take a day shift

- Write constraints for this model that ensure that on every day there are exactly o nurses assigned to a day shift, and between l and u nurses assigned to a night shift. Write the constraint in the most efficient form you can. [3 marks].
- After adding correct constraints of the previous section that constrain the number of nurses on particular shifts, you find the model returns UNSATISFIABLE for all the data sets. Explain in a paragraph what process you would use to find the error in the model. [2 marks].
- Explain where the error is in the above model, and how to correct it. [2 marks].

Question 6 [7 marks]

Given a problem with for the constraints: $y = \text{abs}(x)$, $y = 2z$, $x \neq 0$, $z \neq 1$ and initial domains $D(x) = D(y) = D(z) = -7..7$ and the search decision $x \leq 0$.

- Show the result of idempotent bounds propagation for the constraints: initially, and after the addition of the new constraint by search. Assume that at each stage in the search the propagation loop continually examines all propagators in the order shown until a fixpoint is reached. Show the domains of all variables after any change and the propagator which caused it, e.g.

constraint	$D(x)$	$D(y)$	$D(z)$
	$-7..7$	$-7..7$	$-7..7$
$y = \text{abs}(x)$?	?	?

Recall that bounds propagators only examine and update bounds. [4 marks].

- Show the result of domain propagation for the constraints: initially, and after the addition of the new constraint by search. Assume that at each stage in the search the propagation loop continually examines all propagators in the order shown until a fixpoint is reached. Show the domains of all variables after any change and the propagator which caused it. [3 marks].

Question 7 [11 marks]

- (a) Consider the following MiniZinc predicate:

```
predicate strange(array[int] of var int: x, int: y) =  
  forall(i,j in index_set(x))(  
    if i < j then x[i] + y <= x[j]  
    else true endif );
```

Show the conjunction of primitive constraints of the form $u + v \leq w$ from unrolling the MiniZinc constraint

```
constraint strange([a,b,c,d],3);
```

[3 marks].

- (b) The definition of **strange** is not as simple as it could be and generates redundant constraints. Give a new MiniZinc definition of **strange** that is as efficient as possible but still has the same logical meaning. [3 marks].
- (c) Briefly explain (in at most two sentences) the *relational semantics* used by MiniZinc. [2 marks].
- (d) Compiling the MiniZinc model

```
array[1..5] of var 1..5: x;  
var 3..6: u;  
include "alldifferent.mzn";  
constraint alldifferent(i in 1..5 where i < u)(x[i]);  
solve satisfy;
```

gives an error message

```
MiniZinc: type error: no function or predicate with this signature  
found: 'alldifferent(array[int] of var opt int)'
```

Explain why this error arises, and how you could rewrite the constraint to avoid the error. [3 marks].

Question 8 [16 marks]

An aircargo planning operation has a single plane to shift cargoes around. There are N airports to visit, and E flight legs from airport to airport. There are C cargos which have to be delivered from one airport to another. The plane can carry only one cargo at a time. This is a planning problem with *maxsteps* actions that can be taken, and the possible actions are

Action	shorthand	integer representation
do nothing	donothing	0
pickup cargo c	pickup c	c
move to airport n	move n	$C + n$
drop current cargo	drop	$C + N + 1$

Actions are represented by integers $0..C + N + 1$.

Data for the problem is as follows:

```

int: N;                                % number of airports
set of int: NODE = 1..N;               % airports
int: E;                                % number of flight routes
set of int: EDGE = 1..E;               % routes
array[EDGE] of NODE: fst;               % first airport
array[EDGE] of NODE: snd;               % second airport

int: C;                                % number of cargoes
set of int: CARGO = 1..C;               % cargoes
array[CARGO] of NODE: from;             % cargo pickup place
array[CARGO] of NODE: to;               % cargo dropoff place

set of int: ACT = 0..C+N+1;             % possible actions a, (egs where C = 2, N = 6)
int: donothing = 0;                     % donothing,          0 represents donothing
set of int: PICKUP = 1..C;              % pickup a,          e.g. 2 represents pickup 2
set of int: MOVE = C+1..C+N;            % move (a - C), e.g. 6 represents move 4
int: drop = C + N + 1;                  % drop,              9 represents drop

int: maxsteps;                           % max actions
set of int: STEP = 0..maxsteps;          % steps

NODE: start;                             % aircraft start

```

For example the data file:

```

N = 6;
E = 8+8;
fst = [1,1,2,2,3,3,4,5] ++ [2,4,3,6,4,5,5,6];
snd = [2,4,3,6,4,5,5,6] ++ [1,1,2,2,3,3,4,5];

```

```

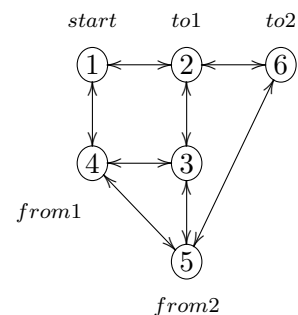
C = 2;
from = [4,5];
to   = [2,6];

```

```
maxsteps = 20;
```

```
start = 1;
```

describes a situation of six airports with 8 possible routes, that are bidirectional, 2 cargoes and the plane starting at airport 1. The airport network is shown to the right



The objective of the plan is to deliver all the cargoes from their pickup position to their dropoff position. A cargo can only be picked up at its pickup place, and only dropped off at its dropoff place. The plane can only move from airport a to airport b if there is an edge (a, b) .

The expected output is a sequence of actions, and the number of steps taken, for example

```
act = [6, 1, 3, 4, 9, 5, 7, 2, 8, 9, 0, 0, 0, 0, 0, 0, 0, 0];
steps = 10;
```

This indicates a plan of move 4, pickup 1, move 1, move 2, drop, move 3, move 5, pickup 2, move 6, drop followed by doing nothing. It takes 10 actions. Clearly each action is allowed: moves are along edges, pickups occur at the right place, and drop offs occur at the right place, and all cargoes are delivered.

- (a) The important state of the problem is: the position of the plane, what it is carrying (if anything), and what cargoes have been delivered. The key decisions are the action at each step, and the number of steps required. Give MiniZinc variable declarations for the key decisions and state representation of the problem. [3 marks].
- (b) Define a MiniZinc predicate that updates the position of the plane given the next action. It should ensure that if the action is a move action then it is legitimate, and if the action is not a move action, that the position does not change. [3 marks].
- (c) Define a MiniZinc predicate that updates state for what the plane is carrying, and which cargoes are delivered depending on the next action. It should make sure that pickup and drop actions are legitimate, e.g. you can't pickup a cargo that has been delivered, you can only pickup a cargo from its “from” position, you only drop a cargo at its “to” position. For actions that do not change this state it should ensure the state is unchanged. [4 marks].
- (d) Define a MiniZinc constraint that ensures that the plan delivers all cargoes. [1 marks].
- (e) Give a correct solve item for the model, together with a search annotation that will make a CP solver effective at tackling your model. In a sentence justify why the search annotation is likely to be useful. [2 marks].
- (f) Complete your MiniZinc model to give a full solution to the problem. [3 marks].