THE UNIVERSITY OF MELBOURNE
DEPARTMENT OF COMPUTER SCIENCE AND SOFTWARE ENGINEERING
SEMESTER 2 ASSESSMENT 2010

433-637 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 6 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.

Question 1 [7 marks]

Consider the following MiniZinc predicate:

```
predicate strange(array[int] of var int:  s) =
          forall(i,j in index_set(s))(
                let { var bool:  b1,
                      var bool:  b2 = not(b1) } in
                if i != j then
                      (b1 -> s[i] < s[j]) /\ (b2 -> s[j] < s[i])
                else true endif );
```

(a) Show the conjunction of primitive constraints of the form $b = not(b')$ and $b \rightarrow a < c$ that result from unrolling the MiniZinc constraint

```
constraint strange([x,y,z]);
```

[4 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].

Question 2 [7 marks]

A boy goes to the milk bar and buys 4 items A, B, C, D each at different price and pays \$10. He receives \$2 and 89 cents change. As he leaves the shop the shopkeeper rushes after him, apologising that he had by accident multiplied the costs of the items in dollars to obtain the cost, rather than added them. He then recalculates the sum of the costs of the 4 items and apologises again since the change is the same!

(a) Define a MiniZinc predicate `milkbar` to determine the prices in cents of the 4 items given in the story above. The arguments to the predicate should be the four variables in order A, B, C, D. [3 marks].

(b) There are variable symmetries in the problem: that is if we take a solution we could swap the value of some variables to create another solution. Modify the predicate to remove as many symmetric solutions as possible. [2 marks].

(c) Comment about the size of integers that appear in your model. Do you expect this problem to be solved easily by a finite domain solver? [2 marks].

Question 3 [8 marks]

(a) Explain the difference between domain consistency and bounds consistency. Illustrate your explanation by showing a case where the bounds consistent propagator and the domain consistent propagator for the same constraint have different behaviour. [3 marks].

(b) Give pseudo-code defining a bounds propagator for the constraint:

$x =$ **if** b **then** y **else** z **endif**

where x, y and z are integers and b is a 0-1 integer (representing a Boolean 0=false, 1=true). The meaning of the constraint should be clear, it has solutions $\{x \mapsto 3, b \mapsto 0, y \mapsto 4, z \mapsto 3\}$ and $\{x \mapsto 4, b \mapsto 1, y \mapsto 4, z \mapsto 3\}$ among many others. Make the propagator as strong as possible. Use functions $\mathsf{lb}(v)$ and $\mathsf{ub}(v)$ to access the current bounds of variable v, and functions $\mathsf{setlb}(v, d)$ and $\mathsf{setub}(v, d)$ to set the bounds of a variable. You can assume that the setting functions won't change to a weaker bound: e.g. $\mathsf{setlb}(x,\mathsf{lb}(x) - 1)$ will have no effect. [5 marks].

Question 4 [6 marks]

The global cardinality constraint, `gcc`, has signature

```
gcc(array[int] of var int:  x, array[int] of int:  val,
    array[int] of var int:  count)
```

It constrains that the total number of times a variable in the array x takes the value $val[i]$ equals $count[i]$. For example the constraints

`gcc`$([1,2,2,1,2,3,4,1],[1,2,4,5],[3,3,1,0])$
`gcc`$([1,2,2,1,2,2,4,1],[1,2,4,5],[3,4,1,0])$

both hold while

`gcc`$([1,2,2,1,2,3,4,1],[1,2,3,4],[3,3,0,1])$

does not.

(a) Give a predicate defining the global cardinality constraint `gcc` by decomposition. [3 marks].

(b) The `gcc_closed` constraint is the same as the global cardinality constraint but also requires that all values taken by x appear in the array val. Hence the first solution above for `gcc` is not a solution for `gcc_closed`. Give a predicate definition for `gcc_closed` using the `gcc` constraint in the body. [1 mark].

(c) The global cardinality constraint can be implemented as a global propagator using maximal matching algorithms akin to those used for `alldifferent`. Contrast the advantages and disadvantages of a global propagator definition against a definition by decomposition into simpler primitive constraints in a paragraph. [2 marks].

Question 5 [8 marks]

The simplex tableau at optimality for the linear relaxation of the problem

maximize $2x + y$
subject to
$$3x + 2y \leq 9$$
$$5x + 3y \leq 12$$

where x and y are non-negative integers, is

$$\begin{array}{rcrrr} z & = & 24/5 & -1/5y & -2/5s_2 \\ s_1 & = & 9/5 & -1/5y & +3/5s_2 \\ x & = & 12/5 & -3/5y & +1/5s_2 \end{array}$$

(a) Generate the cutting plane that results from the equation for x using the method in the slides. [3 marks].

(b) Explain in one paragraph how (pure) cutting plane methods can determine the optimal solution of a linear integer program. [2 marks].

(c) Show the two alternative constraints that will be added in a branch and bound search, when applied to the linear integer problem above. [1 mark].

(d) Show the starting tableau for the dual simplex method for resolving the linear program above after the addition of the first of your two alternative constraints from the previous part. Explain how it was obtained. [2 marks].

Question 6 [12 marks]

Given a problem with (idempotent) bounds consistent propagators for the constraints: $b1 \Leftrightarrow y = t$, $b2 \Leftrightarrow y \geq 3$, $z \leq t$, $y + 2 \geq x$, $x + y + t \leq 8$, $b1 \Rightarrow b2$, where the initial domain of each of the variables x, y, z and t is [0..5]. Assume the search strategy makes decisions in the order: $x \geq 4$, $z \geq 2$, and $y \leq 2$.

(a) Show the result of bounds propagation using the propagators: initially, and after every addition of a new constraint by the 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. [4 marks].

(b) In reality a propagation engine will be far more efficient about examining the propagators that may not be at fixpoint. Give a paragraph explaining how a propagation engine avoids looping through all propagators each time in the search. You may illustrate your answer using part of the example problem. [3 marks].

(c) In a lazy clause generation solver propagation is recorded in an implication graph. Give the implication graph for the problem above, and determine the 1UIP nogood that results. Explain where execution backtracks to, and show the result of propagation using the new nogood. [5 marks].

Question 7 [15 marks]

A caterer needs to plan their napkin provision. They can buy new napkins at some price, or use those in stock, and can clean napkins using two services: a fast one which returns the napkins in few days and a slower cheaper one which returns them in more days. Data for the problem is as follows:

```
int:  days;                          % number of days planning
array[1..days] of int:  demand;      % napkin demand on each day
int:  stock;                         % starting stock of napkins
int:  buy;                           % cost to buy a new napkin
int:  fastclean;                     % cost to clean a napkin fast
int:  fasttime;                      % days for fast clean service
int:  slowclean;                     % cost to clean a napkin slow
int:  slowtime;                      % days for slow clean service
```

For example the data file:

```
days = 8;
demand = [4,0,0,2,4,3,4,2];
stock = 3;
buy = 10;
fastclean = 3;
fasttime = 1;
slowclean = 1;
slowtime = 3;
```

describes a situation where there is an 8 day plan, with varying demands, an initial stock of 3, cost per napkin to buy is 10c, cost to fast clean is 3c and returned next day, cost to slow clean is 1c and returned in 3 days.

Expected output is a plan of the form:

```
bought:  [1, 0, 0, 0, 4, 3, 4, 2]
fast:    [0, 0, 0, 0, 0, 0, 0, 0]
slow:    [2, 0, 0, 0, 0, 0, 0, 0]
```

showing the number of napkins bought (at the beginning of the day), fast cleaned and slow cleaned (at the end of the day) on each day. Above illustrates a very expensive solution for the sample data.

(a) Determine a good solution to the problem with sample data by hand, and show its output in the form above. [2 marks].

(b) Give a MiniZinc model for the napkin problem. You do not need to repeat the data declarations. Ensure the output item returns output in the correct form. [8 marks].

(c) Give a good search annotation for the model. Explain why you think this is likely to give good search behaviour. [2 marks].

(d) Discuss which solver would be best for solving this problem: finite domain, mixed integer programming, Boolean satisfiability or lazy clause generation. Argue why your chosen solver is likely to be preferable. [3 marks].

Question 8 [7 marks]

(a) Explain in a paragraph how activity-based search works in a SAT solver. [2 marks].

(b) Explain in a paragraph how impact-based search works in an FD solver. [3 marks].

(c) A comparator gate used in sorting networks takes two inputs a and b and gives two outputs $c = \max(a, b)$ and $d = \min(a, b)$. Give the conjunctive normal form representation of a comparator for Boolean inputs and outputs. [2 marks].