Constraint Programming (course 1DL440)
Uppsala University – Autumn 2011
Exam

Prepared by Pierre Flener

— Deadline: 13:00 on Monday 19 December 2011 —

Materials: This is a ***closed***-book exam. The usage of electronic devices is ***not*** allowed.

Grading: The grade scale is as follows, when your exam mark is x out of 45 exam points:

Swedish Grade	ECTS Grade	Condition
5	A	$41 \leq x \leq 45$
5	B	$36 \leq x \leq 40$
4	C	$29 \leq x \leq 35$
3	D	$26 \leq x \leq 28$
3	E	$23 \leq x \leq 25$
U	FX	$18 \leq x \leq 22$
U	F	$00 \leq x \leq 17$

Help: Normally, an instructor will attend this exam from 10:00 to 11:00.

Answers: Your answers must be written in English. Provide only the requested information and nothing else. Unreadable, unintelligible, and irrelevant answers will not be considered. Be concise and write each answer immediately behind its question and ***attach extra solution pages only where permitted***: if an answer does not fit into the provided space, then it is unnecessarily long and maybe you should re-read the question. Always show ***all*** the details of your reasoning, and make explicit ***all*** your assumptions, unless otherwise indicated. This question set is double-sided. Circle 'yes' or 'no' below for each question:

Question	Solution Provided?	Max Points	Your Mark
1	yes / no	16	
2	yes / no	10	
3	yes / no	19	
	Total:	45	

Identity: Your anonymous exam code (or name and personal number if you have no code):

. . . .

Question 1: Consistency, Propagation, and Search (16 points)

Consider the following named constraints over the domain $\{1, \ldots, 9\}$:

$$w \cdot w \leq 5 \tag{c}$$

$$v + x \leq 7 \tag{d}$$

$$\textsc{Element}([2, 1, 8, 2, 1, 0], v, x) \tag{e}$$

$$\textsc{Distinct}(\{v, w, x\}) \tag{f}$$

Answer (***only on this page and the next two pages***) the following sub-questions:

A. Using the Propagate algorithm seen in the course, namely the version with propagation conditions and status messages (but without the set *ModVars* of decision variables whose domains have been modified), perform the pre-search propagation to compute the root of the search tree. The following propagation choices are imposed:

- Use ***idempotent*** propagators achieving ***bounds***$(\mathbb{Z})$ consistency for the arithmetic constraints and ***domain*** consistency for the other constraints.
- Post the constraints in the textual order in which they appear above.
- Handle the decision variables in the textual order in which they appear in the stores.
- Use a ***first-in first-out queue*** (FIFO) for implementing the set N of propagators that are not known to be at fixpoint. (Note that N is ***not*** a multi-set.)

In other words, upon denoting the propagator of constraint γ also by γ, do the following:

- For each constraint γ, give (***only in the four lines after this paragraph***, and without proof) the set CondSet(γ) of non-redundant conditions that should trigger the enqueuing of the propagator of γ, as a strictly stronger store might then be obtained. Each propagator condition is of the form any(α), fixed(α), min(α), or max(α), where α is a decision variable of γ (the shortcuts α, $\dot{\alpha}$, $\underline{\alpha}$, and $\overline{\alpha}$, respectively, are also acceptable, here and in the other sub-questions). Write '(none)' rather than nothing, where appropriate. (3 points)

  CondSet(c) = { }

  CondSet(d) = { }

  CondSet(e) = { }

  CondSet(f) = { }
- Fill in the table ***on the next page*** for the initialisation (in the first row) and every pre-search iteration of Propagate, where each status message is as precise as possible, the options being 'subsumed', 'atFixpt', 'unknown' (short for "not known to be at fixpoint"), and 'failed'. Write '(none)' rather than nothing, where appropriate. As specified on the last page of this exam, assume here that the array argument of the $\textsc{Element}$ constraint is ***indexed from* 1**, not from 0. (7 points)
- Indicate (***below***) what changes in your answers to the previous items when instead achieving ***domain*** consistency for ***all*** the constraints. Why? (Note that you are ***not*** asked to fill in another table.) (2 points)

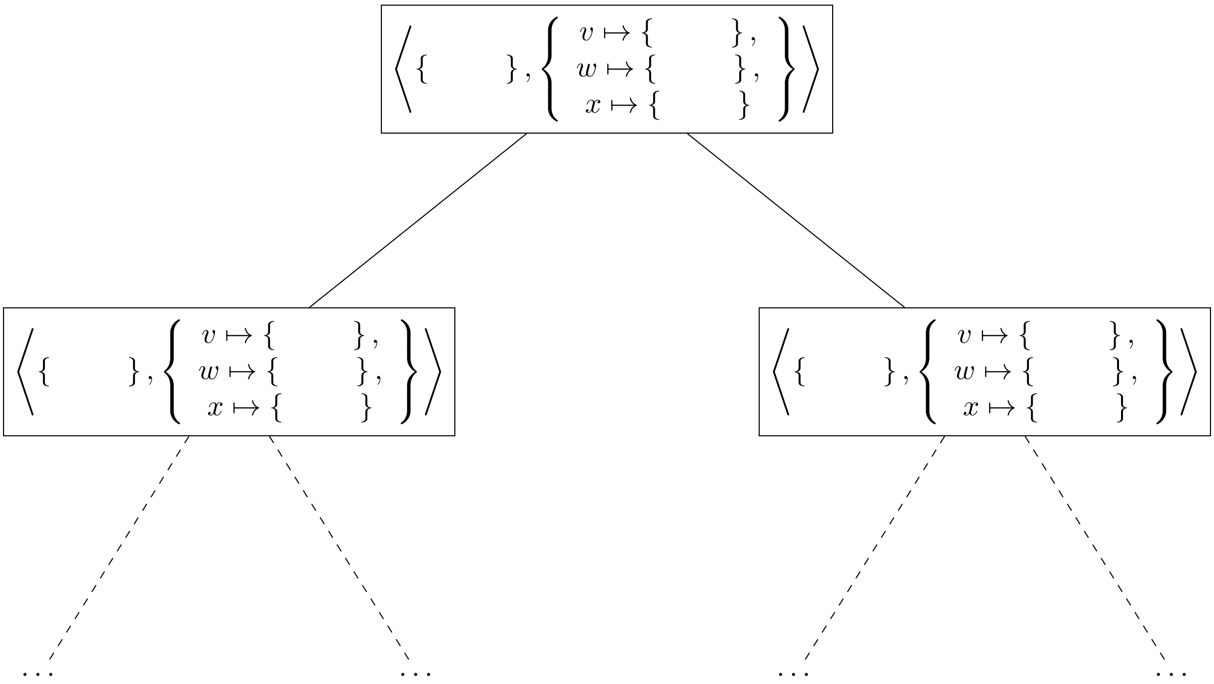
$w \cdot w \leq 5$ (c), $v + x \leq 7$ (d), $\textsc{Element}([2,1,8,2,1,0], v, x)$ (e), $\textsc{Distinct}(\{v,w,x\})$ (f)

Chosen prop.	Resulting store	Status message	Non-redundant raised cond.s	Dependent prop.s *DepProps*	Non-subsumed prop.s P	FIFO queue N of non-fixpt prop.s
(none)	$v \mapsto \{\ \}, w \mapsto \{\ \}, x \mapsto \{\ \}$	(none)	(none)	(none)	$\{\ \}$	$[\ ]$
	$v \mapsto \{\ \}, w \mapsto \{\ \}, x \mapsto \{\ \}$			$\{\ \}$	$\{\ \}$	$[\ ]$
	$v \mapsto \{\ \}, w \mapsto \{\ \}, x \mapsto \{\ \}$			$\{\ \}$	$\{\ \}$	$[\ ]$
	$v \mapsto \{\ \}, w \mapsto \{\ \}, x \mapsto \{\ \}$			$\{\ \}$	$\{\ \}$	$[\ ]$
	$v \mapsto \{\ \}, w \mapsto \{\ \}, x \mapsto \{\ \}$			$\{\ \}$	$\{\ \}$	$[\ ]$
	$v \mapsto \{\ \}, w \mapsto \{\ \}, x \mapsto \{\ \}$			$\{\ \}$	$\{\ \}$	$[\ ]$
	$v \mapsto \{\ \}, w \mapsto \{\ \}, x \mapsto \{\ \}$			$\{\ \}$	$\{\ \}$	$[\ ]$
	$v \mapsto \{\ \}, w \mapsto \{\ \}, x \mapsto \{\ \}$			$\{\ \}$	$\{\ \}$	$[\ ]$
	$v \mapsto \{\ \}, w \mapsto \{\ \}, x \mapsto \{\ \}$			$\{\ \}$	$\{\ \}$	$[\ ]$
	$v \mapsto \{\ \}, w \mapsto \{\ \}, x \mapsto \{\ \}$			$\{\ \}$	$\{\ \}$	$[\ ]$
	$v \mapsto \{\ \}, w \mapsto \{\ \}, x \mapsto \{\ \}$			$\{\ \}$	$\{\ \}$	$[\ ]$
	$v \mapsto \{\ \}, w \mapsto \{\ \}, x \mapsto \{\ \}$			$\{\ \}$	$\{\ \}$	$[\ ]$
	$v \mapsto \{\ \}, w \mapsto \{\ \}, x \mapsto \{\ \}$			$\{\ \}$	$\{\ \}$	$[\ ]$
	$v \mapsto \{\ \}, w \mapsto \{\ \}, x \mapsto \{\ \}$			$\{\ \}$	$\{\ \}$	$[\ ]$
	$v \mapsto \{\ \}, w \mapsto \{\ \}, x \mapsto \{\ \}$			$\{\ \}$	$\{\ \}$	$[\ ]$
	$v \mapsto \{\ \}, w \mapsto \{\ \}, x \mapsto \{\ \}$			$\{\ \}$	$\{\ \}$	$[\ ]$
	$v \mapsto \{\ \}, w \mapsto \{\ \}, x \mapsto \{\ \}$			$\{\ \}$	$\{\ \}$	$[\ ]$
	$v \mapsto \{\ \}, w \mapsto \{\ \}, x \mapsto \{\ \}$			$\{\ \}$	$\{\ \}$	$[\ ]$
	$v \mapsto \{\ \}, w \mapsto \{\ \}, x \mapsto \{\ \}$			$\{\ \}$	$\{\ \}$	$[\ ]$
	$v \mapsto \{\ \}, w \mapsto \{\ \}, x \mapsto \{\ \}$			$\{\ \}$	$\{\ \}$	$[\ ]$
	$v \mapsto \{\ \}, w \mapsto \{\ \}, x \mapsto \{\ \}$			$\{\ \}$	$\{\ \}$	$[\ ]$
	$v \mapsto \{\ \}, w \mapsto \{\ \}, x \mapsto \{\ \}$			$\{\ \}$	$\{\ \}$	$[\ ]$
	$v \mapsto \{\ \}, w \mapsto \{\ \}, x \mapsto \{\ \}$			$\{\ \}$	$\{\ \}$	$[\ ]$
	$v \mapsto \{\ \}, w \mapsto \{\ \}, x \mapsto \{\ \}$			$\{\ \}$	$\{\ \}$	$[\ ]$
	$v \mapsto \{\ \}, w \mapsto \{\ \}, x \mapsto \{\ \}$			$\{\ \}$	$\{\ \}$	$[\ ]$
	$v \mapsto \{\ \}, w \mapsto \{\ \}, x \mapsto \{\ \}$			$\{\ \}$	$\{\ \}$	$[\ ]$
	$v \mapsto \{\ \}, w \mapsto \{\ \}, x \mapsto \{\ \}$			$\{\ \}$	$\{\ \}$	$[\ ]$
	$v \mapsto \{\ \}, w \mapsto \{\ \}, x \mapsto \{\ \}$			$\{\ \}$	$\{\ \}$	$[\ ]$
	$v \mapsto \{\ \}, w \mapsto \{\ \}, x \mapsto \{\ \}$			$\{\ \}$	$\{\ \}$	$[\ ]$
	$v \mapsto \{\ \}, w \mapsto \{\ \}, x \mapsto \{\ \}$			$\{\ \}$	$\{\ \}$	$[\ ]$
	$v \mapsto \{\ \}, w \mapsto \{\ \}, x \mapsto \{\ \}$			$\{\ \}$	$\{\ \}$	$[\ ]$

B. If the pre-search propagation of sub-question 1.A has not solved the problem, then draw (***below***) the search tree, with pairs of non-subsumed propagator sets and constraint stores as nodes, and decisions (which are constraints) as labelled arcs. The propagation choices of sub-question 1.A and the following branching heuristics are imposed:

- Use ***largest-maximum*** variable selection (called INT_VAR_MAX_MAX in *Gecode*).
- Use ***top-down*** value selection (called INT_VAL_MAX in *Gecode*).

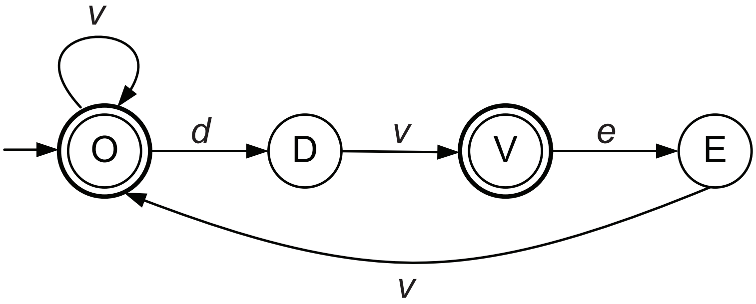
Do ***not*** expand nodes where all propagators are subsumed (whether *Gecode* would detect this or not). Continue to ***use the table on the previous page*** when propagating a branching decision or a constraint, and mark there the starting row of each call to Propagate. (4 points)



The exam continues on the next page!

Question 2: Global Constraints (10 points)

Consider the deterministic finite automaton A below. It describes the legal daily work allocation sequences for one employee. There are values for two work shifts, namely day-shift (d) and evening-shift (e), as well as a value for enjoying a vacation (v). On every day, exactly one of these values must be assigned to the employee. The start state O is marked by a transition entering from nowhere, while states O and V are the accepting states and are marked by double circles. Missing transitions, say from state D upon reading value e, are assumed to go to an implicit failure state, with a self-loop transition for every symbol of the alphabet $\{d, e, v\}$, so that no accepting state is reachable from it.



Answer the following sub-questions about legal work allocation for one employee over five consecutive days. In other words, we examine the propagation to ***domain*** consistency of the constraint $\textsc{Regular}(A, [x_1, x_2, x_3, x_4, x_5])$, starting from the store $\{x_1, x_2, x_3, x_4, x_5 \mapsto \{d, e, v\}\}$:

A. Perform (***on a separate page***) the forward phase of constructing the layered graph. (2 points)

B. Perform (***on the same separate page, with another colour***) the backward phase (without minimisation) of constructing the layered graph. (2 points)

C. Indicate (***below***) all the solutions to this constraint: (1 point)

D. Indicate (***below***) the domain-consistent store obtained from the layered graph: (1 point)

$\{x_1 \mapsto \{\quad\},\ x_2 \mapsto \{\quad\},\ x_3 \mapsto \{\quad\},\ x_4 \mapsto \{\quad\},\ x_5 \mapsto \{\quad\}\}$

E. Explain (***below***) how one makes this propagator incremental: (1 point)

F. Continuing from your store of sub-question 2.D, perform (***on the same separate page, with yet another colour***) what happens when branching or the propagator of some other constraint gives $x_4 \neq v$ and thus wakes up the propagator. Also indicate (***below***) the new resulting domain-consistent store: (2 points)

$\{x_1 \mapsto \{\quad\},\ x_2 \mapsto \{\quad\},\ x_3 \mapsto \{\quad\},\ x_4 \mapsto \{\quad\},\ x_5 \mapsto \{\quad\}\}$

G. Consider the assignments that are permitted by your store of sub-question 2.F: Are they all solutions? Why? (1 point)

Question 3: Modelling (19 points)

Consider the **Progressive Party (PP)** problem. Given a set G of m guest boats, where guest boat $g \in G$ has crew size $size[g]$, and given a set H of ℓ host boats, where host boat $h \in H$ has spare capacity $cap[h]$ (indicating the number of people that can be hosted, other than the crew of h itself), construct a schedule where the crews of the guest boats party at the host boats over a given set $P = \{1, \ldots, n\}$ of party periods such that the following constraints are satisfied:

(party-once-a-period) In each period, the crew of each guest boat parties at some host boat.

(host-at-most-once) The crew of any guest boat parties at any host boat in at most one period.

(meet-at-most-once) The crews of any two distinct guest boats party together at any host boat in at most one period.

(capacity) The spare capacity of any host boat is not exceeded in any period.

Note that some host boats may have no guest crews in some periods, and that guest crews may eventually not visit every host boat.

Answer the following sub-questions (***not here, but on attached separate pages***):

A. Model the PP problem for ***any*** instance. Show how some, if any, of the constraints named above are automatically enforced by your choice of decision variables. (Hint: Try to achieve this for at least one named constraint, via a two-dimensional matrix of scalar decision variables.) Relate each of your constraints to one of the named constraints above, or declare it to be a channelling constraint. (Hint: Recall that with reification one can transform disjunction into integer inequality.) To ease our grading (and your thinking), use mnemonic names, that is use g, g_1, g_2 to denote guest crews of G, as well as h, h_1, h_2 to denote host boats of H, and p to denote a period of P. (12 points)

B. Identify the variable and value symmetries in your ***model***. Show how some, if any, of the symmetries of the ***problem*** are broken by your model. (3 points)

C. Break as many of the symmetries of your model as reasonable. (2 points)

D. Argue for suitable branching heuristics. (2 points)

First read the modelling instructions on the next page!
(They are the same as in the homeworks.)

We will not grade anything written on this or the next page!

Modelling Instructions for Question 3

Your model should be clear and comprehensible, say such that your classmates can understand and implement it without difficulty. Write it in pseudo-code, as in the lecture slides and homeworks. The instance data, as well as the decision variables and their domains, must be declared and their semantics must be given ***in English***, and every constraint must be annotated with an ***English paraphrase***. You may use standard mathematical notation and logical notation, such as (but not limited to) the following:

- $M[i, j]$, to designate the element in row i and column j of a matrix M; similarly for arrays and matrices of any other number of dimensions. You may use $\star$ or intervals to extract an entire slice of a matrix. If some index is an integer decision variable, then you must ***also*** model the constraint using the ELEMENT constraint.
- **sum**(i **in** S) $f(i)$, to designate the sum over all i in set S of the numerical expressions $f(i)$.
- **for all** i **in** S : $c(i)$, to express that for all i in set S the constraint $c(i)$ must hold; we refer to the whole statement as a *quantified constraint*.
- $\wedge$ or & (logical *and*); you may assume an implicit logical *and* between any two (quantified) constraints.
- $\Leftrightarrow$ (is logically equivalent to): you may ***only*** use this connective for the reification of a constraint $c(\dots)$ by a Boolean decision variable b (denoted by $c(\dots) \Leftrightarrow b$).

Note that you may ***not*** use full logic: you may neither use $\vee$ (logical *or*), $\Rightarrow$ (logically implies), or $\Leftrightarrow$ (is logically equivalent to) between two (quantified) constraints, nor use **exists** i **in** S : $c(i)$ to express that there must exist at least one i in set S such that the (quantified) constraint $c(i)$ holds, nor apply $\neg$ (logical negation) to a (quantified) constraint.
If you wrap the implicitly reifying Iverson brackets around a constraint $c(\dots)$ in order to formulate a higher-order constraint $\gamma([c(\dots)])$, then you must ***also*** model that higher-order constraint using explicit reification of $c(\dots)$ by a Boolean decision variable b.
You may use the following global constraints, as well as any others seen in the course:

- DISTINCT($\{x_1, \dots, x_n\}$), also known as ALLDIFFERENT($\{x_1, \dots, x_n\}$), requires that any two decision variables x_i and x_j with distinct indices take distinct values.
- ELEMENT($[a_1, \dots, a_n], x, y$), where $a_1, \dots, a_n, x, y$ are integers or integer decision variables, requires that y be equal to the element at position x (counting from 1) of the array $\langle a_1, \dots, a_n \rangle$, that is $a_x = y$.
- GCC($\{x_1, \dots, x_n\}, [v_1, \dots, v_m], [\ell_1, \dots, \ell_m], [u_1, \dots, u_m]$) requires that the number of decision variables among $\{x_1, \dots, x_n\}$ that take the constant value v_j be between the integer lower bound ℓ_j and integer upper bound u_j inclusive, for all $j \in \{1, \dots, m\}$.
- $[x_1, \dots, x_n] \leq_{\text{lex}} [y_1, \dots, y_n]$ requires that the decision-variable array $[x_1, \dots, x_n]$ be lexicographically smaller than or equal to the decision-variable array $[y_1, \dots, y_n]$.
- LINEAR($[a_1, \dots, a_n], [x_1, \dots, x_n], R, d$) requires that the scalar product of the integer array $[a_1, \dots, a_n]$ with the decision-variable array $[x_1, \dots, x_n]$ be in relation R with the integer d, where $R \in \{<, \leq, =, \neq, \geq, >\}$, that is $\left(\sum_{i=1}^{n} a_i \cdot x_i\right) R\ d$.

as well as ***all*** non-global constraints.

Good Luck!