Constraint Programming (course 1DL440)
Uppsala University – Autumn 2013
Exam

Prepared by Pierre Flener

— Deadline: 13:00 on Friday 20 December 2013 —

Materials: This is a ***closed***-book exam. Electronic devices are ***dis***allowed.
Grading: Your grade is as follows, when your mark is x out of 75 points:

Swedish Grade	Condition
5	$64 \leq x \leq 75$
4	$51 \leq x \leq 63$
3	$38 \leq x \leq 50$
U	$00 \leq x \leq 37$

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 (unless explicitly not requested), and make explicit ***all*** your assumptions. This question set is double-sided. Do ***not*** write anything into the following table:

Question	Max Points	Your Mark
1	27	
2	17	
3	31	
Total	75	

Identity: Your anonymous exam code (or name and personal number if you have none):

☐☐☐☐☐☐ ……………………………………………………

Question 1: Consistency, Propagation, and Search (27 points)

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

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

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

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

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

Perform (***only on this page and the next two pages***) the following sequence of tasks:

A. Using the Propagate fixpoint algorithm seen in the course, namely the version with propagator 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***($\boldsymbol{D}$) 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 Q of propagators that are not known to be at fixpoint. (Note that Q is ***not*** a multi-set.)

In other words, upon denoting the propagator of constraint γ also by γ, perform the following sequence of sub-tasks:

(a) For each constraint γ, give (***only in the lines after this paragraph***, and without proof) the set PropConds(γ) of conditions that should trigger the scheduling 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)', or 'Bounded(α)', where α is a decision variable of γ. Write '(none)' rather than nothing, where appropriate. (5 points)

PropConds(c) = { }

PropConds(d) = { }

PropConds(e) = { }

PropConds(f) = { }

(b) Fill in the table ***on the next page*** for the initialisation (in the first row) and every pre-search iteration of Propagate. Each status message is as precise as possible, the options being 'Subsumed', 'AtFixpt', 'Unknown' (short for "not known to be at fixpoint"), and 'Failed'. Each modification event is of the form 'None(α)', 'Failed(α)', 'Fixed(α)', 'Bounded(α)', or 'Any(α)', where α is a decision variable of the propagated constraint. Write '(none)' rather than nothing, where appropriate. The array argument of the Element constraint is ***indexed from* 1**, not from 0. (12 points)

(c) Indicate (***below***) what changes in your answers to the previous sub-tasks when instead achieving ***domain*** consistency for ***all*** the constraints. Why? (Note that you are ***not*** asked to fill in another table.) (3 points)

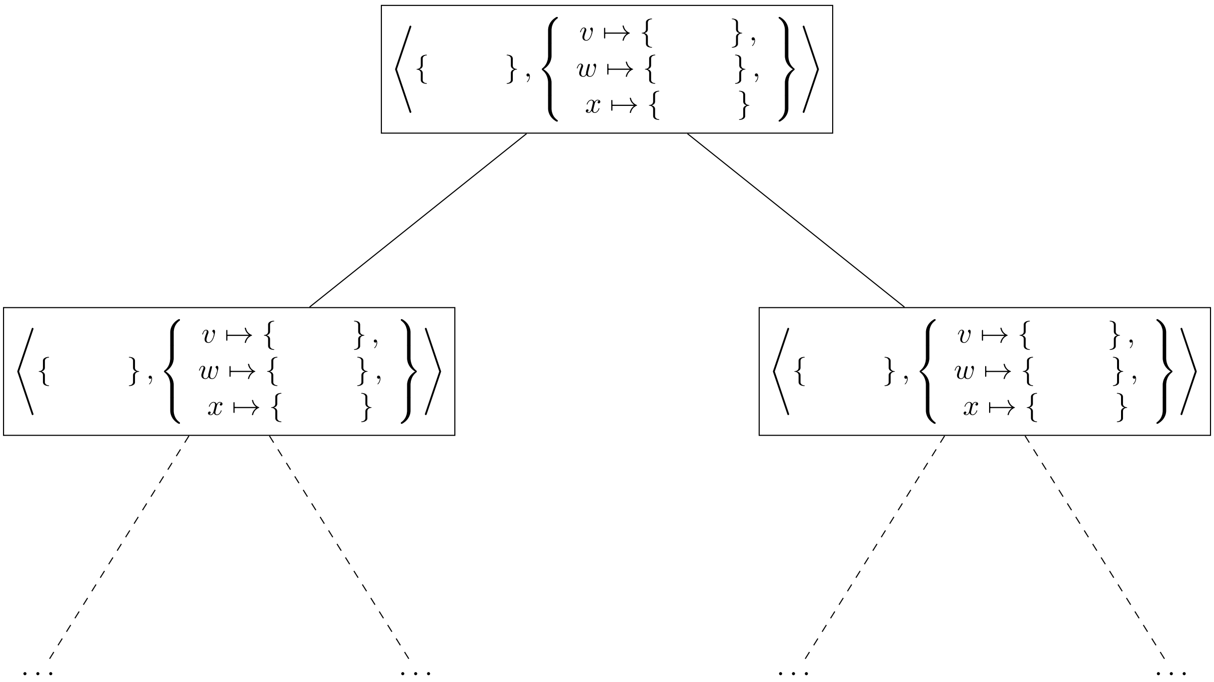
$\textsc{Element}\left(\begin{array}{cccccc}1&2&3&4&5&6\\ \boxed{2}&\boxed{1}&\boxed{8}&\boxed{2}&\boxed{1}&\boxed{0}\end{array}, v, x\right)$ (c), $v + x \leq 7$ (d), $w \cdot w \leq 5$ (e), $\textsc{Distinct}(\{v, w, x\})$ (f)

Chosen prop.	Resulting store	Status message	Modification events	Dependent prop.s *DepProps*	Non-subsumed propagators *R*	FIFO queue *Q* of propagators
(none)	$v \mapsto \{\quad\}, w \mapsto \{\quad\}, x \mapsto \{\quad\}$	(none)	(none)	(none)	$\{\quad\}$	$[\quad]$
	$v \mapsto \{\quad\}, w \mapsto \{\quad\}, x \mapsto \{\quad\}$			$\{\quad\}$	$\{\quad\}$	$[\quad]$
	$v \mapsto \{\quad\}, w \mapsto \{\quad\}, x \mapsto \{\quad\}$			$\{\quad\}$	$\{\quad\}$	$[\quad]$
	$v \mapsto \{\quad\}, w \mapsto \{\quad\}, x \mapsto \{\quad\}$			$\{\quad\}$	$\{\quad\}$	$[\quad]$
	$v \mapsto \{\quad\}, w \mapsto \{\quad\}, x \mapsto \{\quad\}$			$\{\quad\}$	$\{\quad\}$	$[\quad]$
	$v \mapsto \{\quad\}, w \mapsto \{\quad\}, x \mapsto \{\quad\}$			$\{\quad\}$	$\{\quad\}$	$[\quad]$
	$v \mapsto \{\quad\}, w \mapsto \{\quad\}, x \mapsto \{\quad\}$			$\{\quad\}$	$\{\quad\}$	$[\quad]$
	$v \mapsto \{\quad\}, w \mapsto \{\quad\}, x \mapsto \{\quad\}$			$\{\quad\}$	$\{\quad\}$	$[\quad]$
	$v \mapsto \{\quad\}, w \mapsto \{\quad\}, x \mapsto \{\quad\}$			$\{\quad\}$	$\{\quad\}$	$[\quad]$
	$v \mapsto \{\quad\}, w \mapsto \{\quad\}, x \mapsto \{\quad\}$			$\{\quad\}$	$\{\quad\}$	$[\quad]$
	$v \mapsto \{\quad\}, w \mapsto \{\quad\}, x \mapsto \{\quad\}$			$\{\quad\}$	$\{\quad\}$	$[\quad]$
	$v \mapsto \{\quad\}, w \mapsto \{\quad\}, x \mapsto \{\quad\}$			$\{\quad\}$	$\{\quad\}$	$[\quad]$
	$v \mapsto \{\quad\}, w \mapsto \{\quad\}, x \mapsto \{\quad\}$			$\{\quad\}$	$\{\quad\}$	$[\quad]$
	$v \mapsto \{\quad\}, w \mapsto \{\quad\}, x \mapsto \{\quad\}$			$\{\quad\}$	$\{\quad\}$	$[\quad]$
	$v \mapsto \{\quad\}, w \mapsto \{\quad\}, x \mapsto \{\quad\}$			$\{\quad\}$	$\{\quad\}$	$[\quad]$
	$v \mapsto \{\quad\}, w \mapsto \{\quad\}, x \mapsto \{\quad\}$			$\{\quad\}$	$\{\quad\}$	$[\quad]$
	$v \mapsto \{\quad\}, w \mapsto \{\quad\}, x \mapsto \{\quad\}$			$\{\quad\}$	$\{\quad\}$	$[\quad]$
	$v \mapsto \{\quad\}, w \mapsto \{\quad\}, x \mapsto \{\quad\}$			$\{\quad\}$	$\{\quad\}$	$[\quad]$
	$v \mapsto \{\quad\}, w \mapsto \{\quad\}, x \mapsto \{\quad\}$			$\{\quad\}$	$\{\quad\}$	$[\quad]$
	$v \mapsto \{\quad\}, w \mapsto \{\quad\}, x \mapsto \{\quad\}$			$\{\quad\}$	$\{\quad\}$	$[\quad]$
	$v \mapsto \{\quad\}, w \mapsto \{\quad\}, x \mapsto \{\quad\}$			$\{\quad\}$	$\{\quad\}$	$[\quad]$
	$v \mapsto \{\quad\}, w \mapsto \{\quad\}, x \mapsto \{\quad\}$			$\{\quad\}$	$\{\quad\}$	$[\quad]$
	$v \mapsto \{\quad\}, w \mapsto \{\quad\}, x \mapsto \{\quad\}$			$\{\quad\}$	$\{\quad\}$	$[\quad]$
	$v \mapsto \{\quad\}, w \mapsto \{\quad\}, x \mapsto \{\quad\}$			$\{\quad\}$	$\{\quad\}$	$[\quad]$
	$v \mapsto \{\quad\}, w \mapsto \{\quad\}, x \mapsto \{\quad\}$			$\{\quad\}$	$\{\quad\}$	$[\quad]$
	$v \mapsto \{\quad\}, w \mapsto \{\quad\}, x \mapsto \{\quad\}$			$\{\quad\}$	$\{\quad\}$	$[\quad]$
	$v \mapsto \{\quad\}, w \mapsto \{\quad\}, x \mapsto \{\quad\}$			$\{\quad\}$	$\{\quad\}$	$[\quad]$
	$v \mapsto \{\quad\}, w \mapsto \{\quad\}, x \mapsto \{\quad\}$			$\{\quad\}$	$\{\quad\}$	$[\quad]$
	$v \mapsto \{\quad\}, w \mapsto \{\quad\}, x \mapsto \{\quad\}$			$\{\quad\}$	$\{\quad\}$	$[\quad]$
	$v \mapsto \{\quad\}, w \mapsto \{\quad\}, x \mapsto \{\quad\}$			$\{\quad\}$	$\{\quad\}$	$[\quad]$
	$v \mapsto \{\quad\}, w \mapsto \{\quad\}, x \mapsto \{\quad\}$			$\{\quad\}$	$\{\quad\}$	$[\quad]$

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

- Use ***largest-minimum*** variable selection (called INT_VAR_MAX_MIN in *Gecode*).
- Use ***bottom-up*** value selection (called INT_VAL_MIN in *Gecode*).

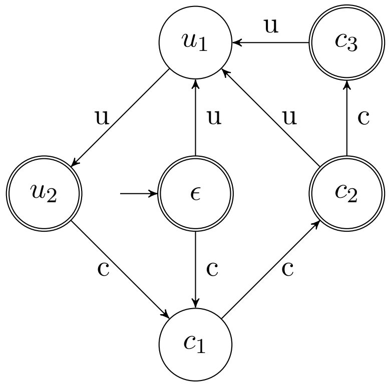
Continue to ***use the table on the previous page*** when propagating a branching decision or a constraint, and mark there the starting row of each call to Propagate. (7 points)



The exam continues on the next page!

Question 2: Global Constraints (17 points)

Consider the finite automaton A below. It describes the language L of words over the alphabet $\Sigma = \{\mathrm{c}, \mathrm{u}\}$ where symbol ‘c’ occurs 2 or 3 consecutive times each time it appears, and symbol ‘u’ occurs exactly 2 consecutive times each time it appears. For instance, “ccuuccc” $\in L$, but “cuu” $\notin L$. The start state ϵ is marked by a transition entering from nowhere; the accepting states ϵ, c_2, c_3, and u_2 are marked by double circles; the non-accepting states are marked by single circles. Missing transitions, say from state c_1 upon reading symbol ‘u’, are assumed to go to an implicit non-accepting state, with a self-loop transition for every symbol of the alphabet, so that no accepting state is reachable from that state. For a sequence X of decision variables over Σ, the automaton A can be used to model as Regular(X, A) the described constraint Stretch$(X, [\mathrm{c}, \mathrm{u}], [2, 2], [3, 2])$, where $[2, 2]$ (respectively $[3, 2]$) contains the minimum (respectively maximum) lengths of stretches in X of the corresponding symbols in $[\mathrm{c}, \mathrm{u}]$.



Let us propagate to ***domain*** consistency the constraint Regular$([x_1, x_2, x_3, x_4, x_5], A)$, starting from the store $\{x_1, x_2, x_3, x_4, x_5 \mapsto \Sigma\}$. Perform the following sequence of tasks:

A. Perform (***on an attached separate page***) the forward phase of constructing the layered graph needed by this propagator. (4 points)

B. Perform (***on the same separate page, using another colour***) the backward phase (without minimisation) of constructing the layered graph. (4 points)

C. Calculate (***below***), using the layered graph, all the solutions to this constraint. (2 points)

D. Indicate (***below***) the constraint store and status message after the pruning phase: (2 points)

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

E. Explain (***below***) how one makes this propagator incremental: (2 points)

F. Continuing from your store of task D, perform (***on the same separate page, using yet another colour***) what happens when branching or the propagator of some other constraint gives $x_4 \neq \mathrm{u}$ and thus eventually wakes up the Regular propagator. Also indicate (***below***) the resulting constraint store and status message: (3 points)

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

Question 3: Modelling (31 points)

Consider the **nurse scheduling problem (NSP)** for hospitals. Given N nurses and a scheduling horizon of D days, the objective is to assign a shift to each nurse on each day. The possible shifts are 'm' (morning), 'a' (afternoon), 'e' (evening), and 'o' (off-duty). Let $S = \{\text{m}, \text{a}, \text{e}, \text{o}\}$. The hospital constraints and the nurse labour union constraints are:

(coverage) For each day d and shift s: there are at least $\mathit{Coverage}[d, s]$ assigned nurses.

(work) For each nurse n and shift s: there are at least $\mathit{Occur}[\text{min}, s]$ and at most $\mathit{Occur}[\text{max}, s]$ assignments of n to s across the D days.

(stretch) For each nurse n and shift s: if n is assigned to s, then there are at least $\mathit{Stretch}[\text{min}, s]$ and at most $\mathit{Stretch}[\text{max}, s]$ consecutive such assignments across the D days.

For instance, for $D = 7$ days and $N = 25$ nurses, the mentioned matrix parameters could be:

Coverage	m	a	e	o
day 1	3	3	2	0
day 2	0	1	2	0
day 3	3	3	1	0
day 4	2	1	1	0
day 5	3	2	1	0
day 6	0	1	2	0
day 7	2	1	1	0

Occur	m	a	e	o
min	0	0	0	1
max	6	6	3	5

Stretch	m	a	e	o
min	1	1	2	1
max	4	4	3	5

Perform the following sequence of tasks (***on attached separate pages***):

A. Model the NSP for ***any*** instance. Show how some, if any, of the constraints above are automatically enforced by your choice of decision variables. Relate each of your constraints to one of the constraints above, or declare it to be a channelling constraint. To ease our grading (and your thinking), use mnemonic names, that is use n to denote nurses, d to denote days, and s to denote shifts. **Hint:** Read the first paragraph of Question 2. ***Read the modelling instructions (of the homeworks) on the next page!*** (20 points)

B. Identify the symmetries of the ***problem***, of the ***instance*** given above, and of your ***model***. Show how some, if any, of the problem symmetries are broken by your model. (6 points)

C. Break as many of the model and instance symmetries as reasonable. (3 points)

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

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 (even reifying Booleans) 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 (but not programming-language-specific lower-ASCII 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 & or **and** (but not &&); you may assume an implicit such 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!