First Exam for Course 1DL023:

Constraint Technology for Solving Combinatorial Problems

Autumn 2008, Uppsala University, Sweden

Prepared by Pierre Flener

Friday 17 October 2008, from 14:00 to 19:00, in Polacksbacken

Cover Sheet

Materials: This is an individual, written, closed-world exam: no written or printed material (except dictionaries) is allowed, nor any electronic devices. Preferably use a pencil.

Grading: The grade scale is as follows, when your exam mark (plus any bonus points from the assignments) is x out of 100 exam points:

Swedish Grade	ECTS Grade	Condition
5	A	$90 \leq x \leq 100$
5	B	$80 \leq x \leq 89$
4	C	$65 \leq x \leq 79$
3	D	$58 \leq x \leq 64$
3	E	$50 \leq x \leq 57$
U	FX	$40 \leq x \leq 49$
U	F	$0 \leq x \leq 39$

Help: Normally, an instructor will attend this exam from 16:00 to 17:00.

Answers: Your answers must be written in English. You *must* show how you reached each answer, unless stated otherwise in the question. Unreadable or unclear answers get zero points. The answer to each question must be started on a new sheet; you can use both sides of each sheet. Write your name on each answer sheet. This cover sheet must be handed in together with your answers. Circle below which questions you have actually addressed:

Question	Solution Provided?	Points	Your Mark
1	yes / no	10	
2	yes / no	15	
3	yes / no	30	
4	yes / no	20	
5	yes / no	25	
	Total:	100	

Identity: Name & Civic Registration Number: ....

Question 1: Constraint Technology

If you met the instructor again in a few years and wanted to tell him what you remembered from this course, how would you succinctly formulate (within the allocated space on this page):

A. The essential features of **constraint-based** complete search. (5 points)

B. The essential features of **constraint-based** local search. (3 points)

C. The relationship between these essential features of these two approaches to combinatorial problem solving offered by constraint technology. (2 points)

Question 2: Stores and Propagators

For each of the functions below on stores:

a. State (without proof) whether it is a propagator or not, and give a counterexample if not.

b. Identify (without proof) the implemented constraint, if it is a propagator.

c. State (without proof) whether it is idempotent or not, and give a counterexample if not.

d. State (without proof) whether it is subsumed (entailed) or not after the first invocation, and give a counterexample if not.

Let $\setminus$ denote set subtraction (hence $S_1 \setminus S_2$ denotes the subset of elements of set S_1 that do not belong to set S_2).

A. (4 points)

$$p_1(s) = \left\{ \begin{array}{l} x \mapsto \{n \in s(x) \mid n \leq 2\}, \\ y \mapsto \{n \in s(y) \mid n \geq 3\} \end{array} \right\}$$

B. (5 points)

$$p_2(s) = \left\{ \begin{array}{l} x \mapsto s(x) \setminus \{1\}, \\ y \mapsto \text{if } s(y) \subseteq s(x) \text{ then } s(y) \text{ else } s(y) \setminus \{2,3,4\} \end{array} \right\}$$

C. (6 points)

$$p_3(s) = \left\{ \begin{array}{l} x \mapsto \{n \in s(x) \mid \min(s(z)) - \max(s(y)) \leq n \leq \max(s(z)) - \min(s(y))\}, \\ y \mapsto \{n \in s(y) \mid \min(s(z)) - \max(s(x)) \leq n \leq \max(s(z)) - \min(s(x))\}, \\ z \mapsto \{n \in s(z) \mid \min(s(x)) + \min(s(y)) \leq n \leq \max(s(x)) + \max(s(y))\} \end{array} \right\}$$

Question 3: Consistency, Propagation, and Search

Quoting from the web-page of a very interesting course:

> *There are two assignments to do during the course. [...] Each assignment is worth* 10 *assignment points. [...] There is a mandatory project to do by the end of the course. [...] The project is worth* 20 *project points. [...] The two assignments and the project are worth* 3 *higher-education credits (ECTS credits): Pass, if you get at least* 12 *project points and your sum of assignment points and project points is at least* 22*; Fail, otherwise.*

Perform the following tasks:

A. Model the problem of finding numbers of points for the assignments and project that yield the 3 credits. Follow the modelling instructions at the bottom of Question 5. (Do **not** call any of the decision variables P, p, c, or N, with or without subscript, because these identifiers have another meaning in the remainder of the question.) (4 points)

B. Using the *propagate* algorithm seen in the course, namely the version with events (also known as propagation conditions) and status messages (but without the set MV of modified decision variables), perform the pre-search propagation. (5 points)

   The following propagation choices are imposed: Achieve **bounds consistency** for all the constraints. Post the constraints in the order in which they appear in the quoted text above. Use a **first-in first-out queue** for implementing the collection N of propagators that are not known to be at fixpoint. Consider a node of the search tree to be *solved* when all its propagators are entailed (subsumed).

   In other words, letting p_i be the propagator of the i^{th} constraint c_i:

   (a) Give (without proof) the smallest set $es(p_i)$ of events that trigger the enqueuing of propagator p_i, for each i, so that a strictly stronger store is obtained.
   (b) Give the initial store.
   (c) Give the initial value of the set P of all non-subsumed propagators.
   (d) Give the initial value of the queue N of propagators not known to be at fixpoint.
   (e) Give the following information after every iteration of *propagate*:

      Picked propagator: ...
      Store: ...
      Status message: 'subsumed', 'at fixpoint', or 'not known to be at fixpoint'
      Raised events: $\{\dots\} \subseteq \{any(?), fix(?), min(?), max(?)\}$
      Set of dependent propagators: $DP = \{\dots\}$
      $P = \{\dots\}$
      $N = [\dots]$

C. If the pre-search propagation has not solved the problem, draw the entire search tree, from the moment where a student has earned 2 points on the first assignment and knows that s/he will not have the time for answering a project question worth 4 points. (21 points)

   The propagation choices above and the following branching heuristics are imposed: Use the smallest-domain variable ordering heuristic (INT_VAR_SIZE_MIN in Gecode/J). Use the lower-half-first value ordering heuristic (INT_VAL_SPLIT_MIN in Gecode/J).

   In other words, after propagating the additional information, interleave branching with propagation. Use the answer template of sub-question (Be) when propagating.

Question 4: Propagation for the *distinct* Global Constraint

Consider the instance below of the Sudoku puzzle, where the task is to fill the cells of the 9×9 grid with integer values in the range $1, \ldots, 9$ such that the cells of each row, each column, and each highlighted 3×3 block are pairwise different:

	6		1		4		5	
		8	3		5	6		
2								1
8			4		7			6
		6				3		
7			9		1			4
5								2
		7	2		6	9		
	4		5		8		7	

Perform the following **sequence** of tasks, on the straightforward constraint model of this problem (seen in the course):

A. Propagate each constraint to **value consistency**; only indicate the domains in the resulting store for the decision variables of the top-left 3×3 block. (We only do this first task in order to make the following task more interesting than it would otherwise be.) (3 points)

B. Propagate the constraint on the top-left 3×3 block to **domain consistency** using Jean-Charles Régin's propagator (1994), showing in full detail (with colours, if you want) what you are doing. Is there any aspect of the underlying theorem by Claude Berge (1970) that is unnecessary? Why? (12 points)

C. What does Régin's propagator do when there are fewer values than decision variables? Why? (2 points)

D. Is Régin's propagator idempotent? Why? (1 point)

E. When can Régin's propagator signal subsumption? Why? (2 points)

Question 5: Modelling

Consider the **Tourist Site Competition (TSC)** problem. Given three non-negative integers r, k, λ and given a set *Cities* of touristic cities and a set *Judges* of judges, construct an assignment of judges to cities such that the following constraints are satisfied:

(constant jury) Every touristic city is visited by r judges.

(constant load) Every judge visits k touristic cities.

(fairness) Every pair of touristic cities is visited by λ common judges.

For instance, solutions exist for $|\mathit{Cities}| = 7 = |\mathit{Judges}|$ with $r = 3 = k$ and $\lambda = 1$.

Answer the following sub-questions:

A. Model the TSC problem for **any** instance. Show how some, if any, of the constraints named above are automatically enforced by your choice of decision variables. Relate each of your constraints to one of the named constraints above, or declare it to be a channelling constraint. (16 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. (3 points)

D. Give suitable branching heuristics. (3 points)

Your model should be clear and comprehensible, say such that your classmates can understand and implement it without difficulty. Write it in pseudocode, as on the modelling lecture slides. **The instance data, as well as the decision variables and their domains must be declared**, possibly in mathematical notation, **and their meanings must be given.** You may use standard mathematical and logical notations, such as $M[i, j]$ (to designate the element in row i and column j of a matrix M; **you can only use this notation when each index is a constant**, or $\star$ in case you want to extract an entire slice of the matrix), $\sum_{i \in S} e(i)$ (to designate the sum over all i in the set/enumeration/range S of the numerical expressions $e(i)$), $\forall i \in S : \alpha(i)$ (to express that for all i in S, the formula $\alpha(i)$ is true), $\wedge$ (and), $\leftrightarrow$ (is equivalent to; use this for reification or channelling), $\rightarrow$ (implies; use this for channelling), and so on. Try hard to avoid $\vee$ (or). You may **not** use $\exists i \in S : \alpha(i)$ (to express that there exists an i in S such that $\alpha(i)$ is true), **nor** $\exists! i \in S : \alpha(i)$ (to express that there exists exactly one i in S such that $\alpha(i)$ is true), nor negation. You can **only** use the following global constraints:

- *element*$(\langle i_1, \ldots, i_n \rangle, x, y)$, where $i_1, \ldots, i_n, x, y$ are integers or decision variables, enforces that y is equal to the x^{th} element of the sequence $\langle i_1, \ldots, i_n \rangle$, that is $i_x = y$.
- *gcc*$(\langle x_1, \ldots, x_n \rangle, \langle v_1, \ldots, v_m \rangle, \langle min_1, \ldots, min_m \rangle, \langle max_1, \ldots, max_m \rangle)$ enforces that the number of elements in the decision-variable sequence $\langle x_1, \ldots, x_n \rangle$ that take the constant value v_j is between the integer constants min_j and max_j inclusive, for all $j \in 1, \ldots, m$.
- *lex*$(\langle x_1, \ldots, x_n \rangle, \langle y_1, \ldots, y_n \rangle)$ enforces that the decision-variable sequence $\langle x_1, \ldots, x_n \rangle$ is lexicographically smaller than or equal to the decision-variable sequence $\langle y_1, \ldots, y_n \rangle$.
- *linear*$(\langle c_1, \ldots, c_n \rangle, \langle x_1, \ldots, x_n \rangle, R, d)$ enforces that the scalar product of the integer sequence $\langle c_1, \ldots, c_n \rangle$ with the decision-variable sequence $\langle x_1, \ldots, x_n \rangle$ is in relation R with the integer d, where $R \in \{<, \leq, =, \neq, \geq, >\}$, that is $(\sum_{i=1}^{n} c_i \cdot x_i)\ R\ d$.