LUNDS UNIVERSITET
Lunds Tekniska Högskola

Institutionen för datavetenskap
*Krzysztof Kuchcinski*

# Tentamen i kursen
# EDAN01: Constraint-Programmering
# (Constraint programming)

2018-01-12, kl. 14-19

**Sal:**

E:2116

**Hjälpmedel:**

Inga

**Resultat anslås:**

Senast 2018-01-26

**Poänggränser:**

Max 40 p., för 4 krävs ca 25 p, för 5 ca 30 p.

**Jourhavande lärare:**

Krzysztof Kuchcinski, tel. 22 23414

*The answers to the questions can be written in Swedish or English.*

***Lycka till!***

# 1 (3 p.)

Constraint programming defines *node*, *arc* and *path* consistency methods for domain pruning based on constraints.

- Explain these methods shortly and give examples of their applications.
- Give example when arc consistency cannot determine inconsistency while path consistency can do it.
- How these methods are related to k-consistency and strong k-consistency.

# 2 (3 p.)

Discuss briefly variable selection strategies for depth-first-search used in finite domain solvers. In your discussion include the following three methods used in JaCoP:

- SmallestMax
- SmallestDomain
- MostConstrainedStatic

For each method define the selections strategy and the motivation for using the method.

# 3 (3 p.)

Use *bounds consistency* and *domain consistency* to prune the domains of FDVs involved in the constraint $y = 3 * x$, $x :: \{0..15\}, y :: \{0..5, 10..17\}$.

# 4 (4 p.)

In constraint programming over finite domain, depth-first-search algorithm is used to enumerate different solutions for a given problem. Discuss different strategies to control the search by selecting variable ordering and domain assignment.

What is the difference between the following two selection choices for finite domain variables.

- choice between $X \leq val$ and $X > val$
- multiple choice for X corresponding to the values in its domain (i.e., $X = val_1, X = val_2, \ldots, X = val_n$)

Illustrate this on an example of list `[X, Y]` with `X :: {1..2}, Y :: {1..3}` by drawing the search tree with values assigned to `X` and `Y` using the following two assumptions.

- variable selection is *anti-first-fail* (in JaCoP called `LargestDomain`), that is a variable with the largest domain size is selected first,
- values $val$ are selected in an ascending order (from min to max) in both cases.

## 5 (3 p.)

A reified constraints is of form $c \Leftrightarrow B$. It consists of a constraint $c$ together with an attached Boolean variable $B$. The reified constraint $c \Leftrightarrow B$ does not require the constraint c to hold but only that the relationship between the constraint and the value of the Boolean variable B holds. The intention is that $B = 1$ when $c$ holds and $B = 0$ when $\neg c$ holds.

Using the reified constraints define the following constraints.

- implication constraint of the form $constraint_1 \rightarrow constraint_2$, where $\rightarrow$ denotes implication.

  Logical implication has the following truth table, where 0 represents **false** and 1 represents **true**.

  | $p$ | $q$ | $p \rightarrow q$ |
  |---|---|---|
  | 1 | 1 | 1 |
  | 1 | 0 | 0 |
  | 0 | 1 | 1 |
  | 0 | 0 | 1 |

  That is the implication only fails when $constraint_1$ is **true** and $constraint_2$ is **false**.

- bidirectional implication constraint of the form $constraint_1 \leftrightarrow constraint_2$, where $\leftrightarrow$ denotes bidirectional implication.

  Bidirectional implication has the following truth table, where 0 represents **false** and 1 represents **true**.

  | $p$ | $q$ | $p \leftrightarrow q$ |
  |---|---|---|
  | 1 | 1 | 1 |
  | 1 | 0 | 0 |
  | 0 | 1 | 0 |
  | 0 | 0 | 1 |

  That is the bidirectional implication only succeeds when both constraints are **true** or **false**.

## 6 (3 p.)

Explain why finite domain solvers use global constraints. As an example use `Alldifferent` constraints and illustrate the propagation "strength" with different implementations of the consistency methods, such as a method that is equivalent to imposing inequality constraints, bounds consistency and hyper-arc consistency.

## 7 (6 p.)

A curriculum is a set of 20 courses numbered from 1 to 20 that all students must study. Each course must be assigned to one of 4 periods.

Courses have prerequisites and cannot be scheduled before its prerequisites. The following pairs, represented as $(i, j)$, mean that course $j$ must be taken before course $i$, i.e. course $j$ is a prerequisite for course $i$.

$$[(3, 1), (4, 1), (5, 1), (6, 1), (7, 1), (6, 2), (8, 2)]$$

Each course gives a number of points, specified as a list of points
$points = [6, 3, 5, 3, 7, 8, 1, 9, 4, 9, 8, 8, 4, 5, 6, 3, 2, 1, 3, 1]$

Students have to collect at least two points and maximum of 25 points in a given period. They must also study at least two and maximum of 10 courses in each period.

The goal is to assign a period to every course satisfying these criteria, minimising the sum of points assigned to each periods.

# 8 (6 p.)

The jobshop scheduling problem requires to schedule number of jobs on available machines. Each job is a sequence of tasks which need to be executed on different machines (M). The duration of each task is D. The table below specifies a simple job shop scheduling problem which has 3 jobs (each job containing the sequence of 3 tasks) and 3 machines (0, 1, and 2). Formulate the problem using a finite domain constraint solver together schedule length minimization method.

| Job | Task 1 | | Task 2 | | Task 3 | |
|---|---|---|---|---|---|---|
| | M | D | M | D | M | D |
| 1 | 1 | 7 | 0 | 8 | 1 | 9 |
| 2 | 0 | 5 | 2 | 6 | 0 | 11 |
| 3 | 2 | 12 | 1 | 10 | 2 | 3 |

# 9 (3 p.)

Nurse planing in a hospital requires to assign seven nurses to seven days and four shifts in every day. Correct assignment requires that each day there are four different nurses available, one for each shift. Moreover there is a requirement that each shift has different nurses assigned during different days. An example assignment is depicted in the table below.

| | Day 1 | Day 2 | Day 3 | Day 4 | Day 5 | Day 6 | Day 7 |
|---|---|---|---|---|---|---|---|
| Shift 1 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Shift 2 | 2 | 1 | 4 | 3 | 6 | 7 | 5 |
| Shift 3 | 3 | 4 | 1 | 2 | 7 | 5 | 6 |
| Shift 4 | 4 | 5 | 6 | 7 | 1 | 2 | 3 |

Indicate which symmetries exist in this problem and how can they be eliminated.

# 10 (6 p.)

The traveling salesperson needs to visit four cities, A, B, C, and D, to sell products. There is different distance between cities and the salesperson wants to optimize the route. The distance between cities is given in the table. Note, that it is non-symmetrical traveling salesperson problem, i.e. distances are different in different directions.

| | A | B | C | D |
|---|---|---|---|---|
| A | - | 3 | 6 | 41 |
| B | 4 | - | 40 | 5 |
| C | 8 | 42 | - | 4 |
| D | 37 | 6 | 2 | - |

- Write a program using FD solver which finds the minimal distance route which contains all cities.
- Assume that the salesperson distributes parcels in the cities as presented in the table below. Positive integer means that the specified number of parcels is loaded into the car and negative integer means that this number of parcels is unloaded. Find a shortest rout that fulfills car maximum capacity of 5 parcels, i.e., a route that one cannot unload from an empty car and load number of parcels that exceeds car capacity.

| | Cities | | | |
|---|---|---|---|---|
| | A | B | C | D |
| Distribution | 2 | -3 | 4 | -1 |

Write a pseudo-code or minizinc program that models and solves this problem.