Department of Mathematics and Computer Science
University of Southern Denmark, Odense

Monday 23rd October, 2023
Marco Chiarandini

DM841 - Heuristics and Constraint Programming

Obligatory Assignment 2, Autumn 2023

Deadline: Friday, October 27, 2023 at 16:00.

Preliminary Remarks

- This is the *second* of two preparatory assignments on constraint programming. The assignment is compulsory with pass/fail evaluation.
- The aim of this assignment is to work with pen and paper on the concepts and definitions of local consistency and propagation discussed in class and working further on modeling.
- Each group has to submit the following elements:
  - short report in PDF

  The submission is made by one member of the group only. The PDF document containing the report must be anonymous. The full name of the group members and their SDU usernames must appear in file with name README.md inside the directory asg2/.
- The submission is electronic with the same modalities and submission system as in Assignment 1.

  Organize your documents as follows:

```
-- asg2
| |-- README.txt
| |-- doc
|     |-- main.pdf
```

- You have to hand in one file in PDF format containing your responses to the tasks below. You can handwrite your answers and scan them. Alternatively, source code for the figures and tables is linked in the captions of the figures.

Propagation

The following assignment was posed in a written exam by Prof. Pierre Flener for the course on Constraint Programming at Uppsala University.
Consider the following CSP and its labelled constraints:

$$\mathcal{P} := \langle X := \{v, w, x\}; \mathcal{DE} := \{D(v) = D(w) = D(x) = \{1, \ldots, 9\}\}; \mathcal{C} := \{$$

$$\textsc{Element}(v, x, a = [2, 1, 8, 2, 1, 0]) \equiv x = a_v \quad \text{(c)}$$

$$v + x \leq 7 \quad \text{(d)}$$

$$w \cdot w \leq 5 \quad \text{(e)}$$

$$\textsc{Distinct}(\{v, w, x\})\} \rangle \quad \text{(f)}$$

We will look at the execution of the constraint propagation for this problem. The algorithm uses propagator conditions (events), status messages and keeps track of variables whose domains have been modified.
The following propagation choices are imposed:

- Use **strongly idempotent**[1] propagators achieving **bounds(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 CSP above.
- 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.)

Upon denoting the propagator of constraint C by $p_C(\cdot)$, address the following tasks:

Task 1 Formalize the propagators in a similar way as seen in class:

$p_{\text{Element}}(x, \mathcal{DE}) = p_{\text{Element}}(D)(x) = \{$ $\}$
...

Task 2 In the Constraint-Propag fixpoint algorithm seen in the course (slide 15, lecture "Rules Iteration") one needs to know when propagators are to be executed. For each constraint C, give (without proof) the set PropConds$(\cdot)$ of conditions that should trigger the scheduling of the propagators of $C \in \mathcal{C}$, as a strictly tighter CSP might then be obtained. Each propagator condition is of the form 'Any(α)', 'Fixed(α)', or 'Min/Max(α)', where α is a decision variable of C. Write '(none)' rather than nothing, where appropriate. Write the most general condition (eg, Any() is more general than Min/Max()).

PropConds(c) $= \{$ $\}$

PropConds(d) $= \{$ $\}$

PropConds(e) $= \{$ $\}$

PropConds(f) $= \{$ $\}$

Task 3 Using a procedure like the Constraint-Propag fixpoint algorithm, perform the pre-search propagation to compute the fixpoint of the root of the search tree. Fill in the data in a table like Table 1 for the initialisation (in the first row) and every pre-search step of your Constraint-Propag. (The array argument of the Element constraint is **indexed from 1**, not from 0.) Write the status message of the propagators after their execution as precisely as possible, the options being 'AtFixpt', 'Unknwon' (short for "not known to be at fixpoint"), 'Subsumed' and 'Failed'. In addition, write the modification event that occurs. The modification event is of the form 'None(α)', 'Failed(α)', 'Fixed(α)', 'Min/Max(α)', or 'Any(α)', where α is a decision variable of the propagated constraint. Write '(none)' rather than nothing, where appropriate. Finally, write the dependant propagators that are triggered by the events occurred and how the new queue of propagators will look like.

Task 4 Indicate 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.)

[1] To be strongly idempotent it means that reapplying the propagator to the same problem reached after the first application does not change the problem (unless the problem has changed). Waekly idempotent means that sometimes the propagator may turn out to be idempotent but not always. See second example on slide 8 of "Propagation Events and Implementations".

ELEMENT $\left(\begin{array}{|c|c|c|c|c|c|}\hline 2 & 1 & 8 & 2 & 1 & 0 \\ \hline\end{array}, v, x\right)$ (c), $v + x \leq 7$ (d), $w \cdot w \leq 5$ (e), DISTINCT($\{v, w, x\}$) (f)

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

Table 1: Latex source code for this table.

Task 5 If the pre-search propagation of sub-task 3. has not solved the problem, then draw (**below**) the search tree with **all** the solutions, with pairs of non-subsumed propagator sets and domain extensions as nodes, and decisions (which are constraints) as labelled arcs. The previous choices in this task and the following branching heuristics are imposed:

- Use **largest-minimum** variable selection
  (`int_search( [v,w,x], largest, <assignmentannotation>)` in Minizinc).
- Use **bottom-up** value selection
  (`int_search( [v,w,x], <varchoiceannnotation>, indomain_min)` in Minizinc).

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 Constraint-Propag.

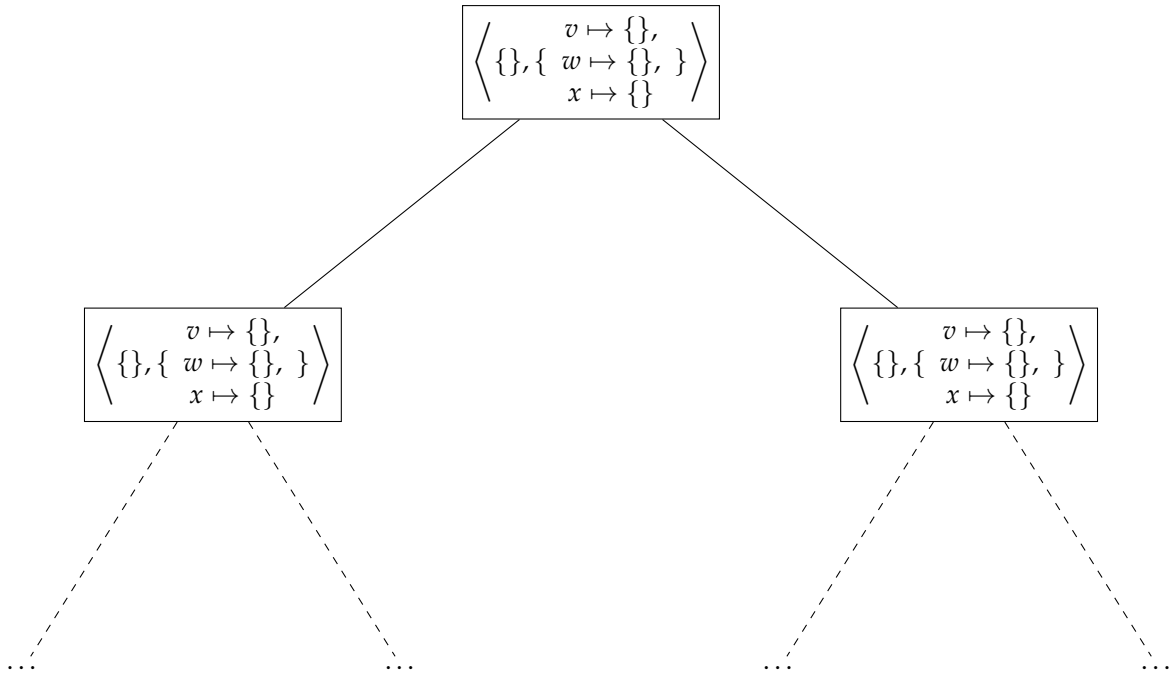

Figure 1: Latex source code for this figure.

1 Modeling

Model the following two problems in constraint programming. You do not need to solve numerical instances.

Task 6

Given a set of patients distributed in a number of hospital zones and an available nursing staff, one must assign a nurse to each patient in such a way that the work is distributed evenly between nurses. Each patient is assigned an acuity level corresponding to the amount of care he requires; the workload of a nurse is defined as the sum of the acuities of the patients he cares for. A nurse can only work in one zone and there are retrictions both on the number of patients assigned to a nurse and on the corresponding workload. We balance the workloads by minimizing their standard deviation.
This problem can be decomposed in two phases: *nurse staffing* that assigns nurses to zones and *nurse-patient assignment* that then assigns patients to nurses.
A variant of this problem has patients grouped into a small number of types and has the acuity associated with each patient type being nurse-dependent. Note that the total workload is no longer known a priori since each patient's acuity now depends on which nurse he is assigned to. There are two objectives to minimize: the total workload and the standard deviation of the workloads.

Task 7

The problem consists in transporting patients to medical appointments given a fleet of vehicles. The main objective is to satisfy as much patient requests as possible. Additional objectives such as minimizing the waiting time or maximum ride time of the patients or towards minimizing costs could be considered.
Each request consists of a forward travel where the patient must be brought to their destination on time for their appointment, a backward travel that must be done after the end of the appointment or both. For back and forth requests, both travels must be done for the request to be satisfied. Patients are associated with categories restricting the vehicles that can take them and might be accompanied. Patients might also require some amount of time to embark/disembark vehicles. The vehicles have a limited capacity which must never be exceeded and are available at fixed times.
As an example, the input data for the problem is provided as JSON files having the following format:

- The "coordType" field indicates if the coordinates are geographical ("Geo") or euclidian ("Eucl").
- The "sameVehicleBackWard" field indicates if the same vehicle has to be used for the two trips of a back and forth request.
- The "maxWaitTime" field contains the duration of the time windows during which the patient has to be transported. It is expressed as a string having the following format: "HHhMM"
- The "places" field contains all the distinct locations of the problem. For each of them:
  - "id" is the id of the location. It corresponds to its position in the list of locations.
  - "lat" and "long" are the coordinates of the location.
  - "category" is the category of the location. Its value can be 0 for a medical center, 1 for a vehicle depot and 2 for a patient location.
- The "vehicles" field contains all the vehicles available in the fleet. Each of them has the following fields:
  - "id" which contains the id of the vehicle.
  - "canTake" which is a set of categories of patients that the vehicle can take.
  - "start" and "end" which are the ids of the starting and ending depots of the vehicle. A -1 value indicates that the vehicle has no depot.
  - "capacity" which is the capacity of the vehicle.
  - "availability" which is a set of time windows when the vehicle is available. Each time window is encoded by a string having the following format: "HHhMM:HHhMM".
- The "patients" field contains all the requests. Each patient/request has the following fields:
  - "id" which contains the id of the request.
  - "category" which indicates the category of the patient
  - "load" which indicates the number of places required in the vehicle during the transport.

  - "start", "destination" and "end" which indicate the ids of respectfully the starting location, medical center and return location for the patient. "start" or "end" can have a -1 value in case of single trip.
  - "rdvTime" and "rdvDuration" indicate the start of the appointment and its duration as strings under the format "HHhMM". In terms of constraints, the patient must be picked up at its starting location for its forward trip at or after rdvTime - maxWaitTime. They must be dropped at their destination before or at rdvTime. They must be picked up for their backward trip at the medical center after or at rdvTime + rdvDuration and must be dropped at their end location before or at rdvTime + rdvDuration + maxWaitTime.
  - "srvDuration" indicates the time needed for the patient to embark/disembark the vehicle. It is encoded as a string under the format "HHhMM".

- The "distMatrix" field contains the distance (in minutes) matrix between the locations. it follows the ids and order of the locations (distMatrix[2][3] is the distance from location 2 to location 3).


```
{
  "coordType" : "Eucl",
  "sameVehicleBackward" : false,
  "maxWaitTime" : "00h30",
  "places" : [ {
    "id" : 0,
    "lat" : -9.765289504081604,
    "long" : 0.09260531673304229,
    "category" : 0
  }, {
    "id" : 1,
    "lat" : 4.673051931629041,
    "long" : 5.956203854277059,
    "category" : 0
  },
  ...
  {
    "id" : 19,
    "lat" : -2.0424908483847424,
    "long" : -1.5015726999113692,
    "category" : 2
  }, {
    "id" : 20,
    "lat" : -0.7019351233585336,
    "long" : 6.7146606740927535,
    "category" : 2
  } ],
  "vehicles" : [ {
    "id" : 21,
    "canTake" : [ 0 ],
    "start" : 4,
    "end" : 4,
    "capacity" : 6,
    "availability" : [ "07h00:20h00" ]
  }, {
    "id" : 22,
    "canTake" : [ 0 ],
    "start" : 4,
    "end" : 4,
    "capacity" : 6,
    "availability" : [ "07h00:20h00" ]
  } ],
  "patients" : [ {
    "id" : 23,
    "category" : 0,
    "load" : 1,
    "start" : 5,
    "destination" : 0,
    "end" : 5,
    "rdvTime" : "10h43",
```

```
    "rdvDuration" : "00h23",
    "srvDuration" : "00h03"
   },
  ...
  {
    "id" : 37,
    "category" : 0,
    "load" : 1,
    "start" : 19,
    "destination" : 3,
    "end" : -1,
    "rdvTime" : "15h56",
    "rdvDuration" : "02h36",
    "srvDuration" : "00h01"
  }, {
    "id" : 38,
    "category" : 0,
    "load" : 2,
    "start" : 20,
    "destination" : 2,
    "end" : 20,
    "rdvTime" : "08h19",
    "rdvDuration" : "02h36",
    "srvDuration" : "00h01"
  } ],
  "distMatrix" : [ [ 0, 16, 19, 9, 10, 14, 18, 18, 12, 12, 7, 11, 3, 7, 17, 8, 4, 11, 5, 8, 11
    ], [ 16, 0, 8, 9, 6, 7, 11, 9, 6, 13, 9, 16, 12, 12, 14, 8, 12, 16, 17, 10, 5 ], [ 19, 8, 0,
     10, 13, 5, 3, 1, 8, 10, 12, 13, 17, 12, 7, 13, 16, 13, 17, 11, 13 ], [ 9, 9, 10, 0, 8, 5,
    9, 9, 3, 6, 2, 7, 7, 3, 9, 5, 7, 8, 8, 1, 8 ], [ 10, 6, 13, 8, 0, 10, 14, 13, 7, 13, 6, 15,
    7, 9, 16, 3, 7, 15, 13, 8, 1 ], [ 14, 7, 5, 5, 10, 0, 5, 4, 3, 7, 7, 10, 12, 8, 7, 8, 11,
    10, 13, 6, 9 ], [ 18, 11, 3, 9, 14, 5, 0, 2, 8, 7, 11, 11, 16, 11, 4, 13, 15, 10, 15, 10, 14
     ], [ 18, 9, 1, 9, 13, 4, 2, 0, 7, 8, 11, 12, 16, 11, 6, 12, 15, 12, 16, 10, 12 ], [ 12, 6,
    8, 3, 7, 3, 8, 7, 0, 8, 4, 10, 9, 6, 9, 5, 9, 10, 11, 5, 6 ], [ 12, 13, 10, 6, 13, 7, 7, 8,
    8, 0, 8, 4, 12, 5, 5, 11, 11, 4, 9, 6, 13 ], [ 7, 9, 12, 2, 6, 7, 11, 11, 4, 8, 0, 9, 5, 3,
    12, 3, 4, 9, 8, 2, 6 ], [ 11, 16, 13, 7, 15, 10, 11, 12, 10, 4, 9, 0, 12, 6, 8, 12, 11, 0,
    7, 7, 15 ], [ 3, 12, 17, 7, 7, 12, 16, 16, 9, 12, 5, 12, 0, 6, 16, 5, 0, 12, 7, 6, 8 ], [ 7,
     12, 12, 3, 9, 8, 11, 11, 6, 5, 3, 6, 6, 0, 10, 6, 6, 6, 5, 2, 10 ], [ 17, 14, 7, 9, 16, 7,
    4, 6, 9, 5, 12, 8, 16, 10, 0, 14, 16, 8, 14, 10, 16 ], [ 8, 8, 13, 5, 3, 8, 13, 12, 5, 11,
    3, 12, 5, 6, 14, 0, 4, 12, 10, 5, 4 ], [ 4, 12, 16, 7, 7, 11, 15, 15, 9, 11, 4, 11, 0, 6,
    16, 4, 0, 12, 7, 6, 8 ], [ 11, 16, 13, 8, 15, 10, 10, 12, 10, 4, 9, 0, 12, 6, 8, 12, 12, 0,
    7, 7, 15 ], [ 5, 17, 17, 8, 13, 13, 15, 16, 11, 9, 8, 7, 7, 5, 14, 10, 7, 7, 0, 7, 14 ], [
    8, 10, 11, 1, 8, 6, 10, 10, 5, 6, 2, 7, 6, 2, 10, 5, 6, 7, 7, 0, 8 ], [ 11, 5, 13, 8, 1, 9,
    14, 12, 6, 13, 6, 15, 8, 10, 16, 4, 8, 15, 14, 8, 0 ] ]
}
```