School of Computer Science
University of St Andrews
2018-19
CS4402
Constraint Programming
Practical 2: Constraint Solver Implementation

This Practical comprises 50% of the coursework component of CS4402. It is due on Thursday 18th April at 21:00. The deliverables comprise:

- A report, the contents of which are specified below.
- The Java source code for the constraint solver you will implement.
- The instance files for any CSP instances you use in addition to those provided to test your solver.

The practical will be marked following the standard mark descriptors as given in the Student Handbook (see link below).

Problem Specification

This practical is to design, implement in Java, and test empirically a constraint solver for **binary constraints**. Your solver should employ **2-way branching**, and it should implement both the **Forward Checking** and the **Maintaining Arc Consistency** algorithms. Its search strategy should be **smallest-domain first**, with **ascending** value ordering.

Supplied Files

Accompanying this specification you will find the following Java files, which you should extend and/or modify to produce your submission:

- `BinaryConstraint.java`
- `BinaryCSP.java`
- `BinaryCSPReader.java`
- `BinaryTuple.java`

In addition, you will find ten `.csp` files (a format that can be read by the `BinaryCSPReader`), which contain instances of three problem classes that you should use to test your solver. Note that the `.csp` format assumes that variable domains are specified simply as an integer range. You may wish to extend this.

There are also three Generator Java source files that were used to generate these instances, and can be used to generate more. The Sudoku generator produces the constraints only for the Sudoku puzzle – you will need to edit the domains in the generated .csp file to provide the clues for a particular instance. See the two provided Sudoku instances for examples. Langford's Number Problem may be unfamiliar to you. Its description can be found at CSPLib entry 24: http://www.csplib.org/Problems/prob024/

Basic Solver Design

In designing your solver, you will need to decide upon a suitable representation for variables and domains. A partial implementation of binary extensional constraints is provided in `BinaryConstraint` and `BinaryTuple`. Considerations:

- To implement both Forward Checking and Maintaining Arc Consistency your design must support domain pruning via arc revision. It must also support a mechanism by which domain pruning is undone upon backtracking.
- The main additional consideration for Maintaining Arc Consistency over Forward Checking is the queue. Again, careful implementation is required here. Take care also to **initialise the queue correctly** (see your lecture notes).
- Your implementation of the smallest-domain first heuristic should access the current state of the pruned domains to make its decisions.
- Take care in implementing two-way branching that arc revision is performed on **both branches**.

Your source code should be well commented.

Empirical Evaluation

Using the supplied instances and instance generators, design and run a set of experiments to compare the merits of Forward Checking and Maintaining Arc Consistency. You will need to decide upon one or more sensible bases for comparison, such as time taken, nodes in the search tree, or arc revisions. Your solver will need to be instrumented appropriately to measure these criteria.

There are some ideas for a more extensive empirical evaluation in the Extensions section below.

Extensions

There follow some ideas for extension activities. These are not required, but attempting at least one extension activity is required to gain a mark above 17.

Other Problem Classes

A simple way to extend your empirical evaluation is to use more problem classes. You might look through CSPLib for problems that can easily be expressed as binary CSPs, otherwise you do know, via the dual encoding, how to transform any problem into a binary CSP.

Constraint Types

Extend your solver to support binary intensional constraints, such as equality, disequality or inequality.

Heuristics

Extend your solver to support static variable orderings supplied as a list of variables in the order that they should be assigned. It would be sensible to implement a separate search control file and reader for this information to avoid duplicating large `.csp` files that differ only in the heuristic selected. You might also implement more sophisticated dynamic heuristics, such as Brelaz, and/or other value heuristics.

Report

Your report should have the following sections:

- **Forward Checking**: Describe your implementation of the Forward Checking algorithm, including the data structures you used and your method of supporting domain pruning (and restoration following backtracking). Include here also an account of how you implemented binary branching.
- **Maintaining Arc Consistency**: Similarly, describe how you implemented this algorithm, and in particular how you manage the queue.
- **Empirical Evaluation**: Describe your experimental setup, and record and analyse the output of the empirical evaluation described above.
- **Extensions**: Describe here the extensions you have attempted, if any. Empirical work should be reported as above.

Marking

This practical will be marked following the standard mark descriptors as given in the Student Handbook. There follows further guidance as to what is expected:

- To achieve a mark of 7 or higher: A rudimentary attempt at Forward Checking only. Adequate evaluation and report.
- To achieve a mark of 11 or higher: A reasonable attempt at both Forward Checking and Maintaining Arc Consistency, mostly complete, or complete and with a few flaws. Reasonably well evaluated and reported.
- To achieve a mark of 14 or higher: A good attempt at both Forward Checking and Maintaining Arc Consistency, with only minor flaws. Well evaluated and reported.
- To achieve a mark of 17: A fully correct implementation of both algorithms, very well evaluated and reported.
- To achieve a mark greater than 17: In addition to the requirements for a mark of 17, an attempt at the suggested extension activities.

Pointers

Your attention is drawn to the following:

- Mark Descriptors:
  https://info.cs.st-andrews.ac.uk/student-handbook/learning-teaching/feedback.html
- Lateness:
  https://info.cs.st-andrews.ac.uk/student-handbook/learning-teaching/assessment.html
- Good Academic Practice:
  https://info.cs.st-andrews.ac.uk/student-handbook/academic/gap.html