School of Computer Science
University of St Andrews
2017-18
CS4402
Constraint Programming
Practical 2: Constraint Solver Implementation

This Practical comprises 50% of the practical component of CS4402. It is due on Wednesday 18th April at 21:00.

The deliverables comprise:

- A report.
- 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.

Your source code should be well commented. Your report should document your solver design decisions in detail. It should also justify your experimental design, describe it in sufficient detail that a competent reader could repeat your experiments, and contain a detailed analysis of your experimental results.

Basic Specification

This practical is to design, implement in Java, and test empirically a Forward Checking constraint solver for **binary constraints**. Your solver should employ **2-way branching**.

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. To implement Forward Checking your design must support domain pruning via arc revision. It must also support a mechanism by which domain pruning is undone upon backtracking. Take care in implementing two-way branching that arc revision is performed on both branches.

Heuristics

Extend your solver to support static variable orderings supplied as a list of variables in the order that they should be assigned. Also extend your solver to support the dynamic smallest domain first heuristic. 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.

Empirical Evaluation

Using the supplied instances and instance generators, design and run a set of experiments to compare the merits of various static variables orderings (of your own choosing) with smallest domain first. 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.

Extensions

Extend your solver to perform MAC (Maintaining Arc Consistency) as an option alongside Forward Checking. Extend your empirical evaluation by comparing MAC with Forward Checking.

Further possible extensions include:

- Support for (binary) intensional constraints, such as equality, disequality or inequality.
- Further variable heuristics, such as Brelaz.
- Value heuristics.

Marking

This practical will be marked following the standard mark descriptors as given in the Student Handbook.

It is possible to obtain a mark of 17 by completing the Basic Specification (including report) to a high standard. To obtain a mark greater than 17 some extension activities must be attempted. A mark of 7 might be obtained by submitting a flawed (but serious attempt at an) implementation of a Forward Checking constraint solver along with a report documenting the design and problems encountered.

Pointers

Your attention is drawn to the following:

Marking

See the standard mark descriptors in the School Student Handbook:

http://info.cs.st-andrews.ac.uk/student-handbook/learning-teaching/feedback.html#Mark_Descriptors

Lateness

The standard penalty for late submission applies (Scheme B: 1 mark per 8 hour period, or part thereof):

http://info.cs.st-andrews.ac.uk/student-handbook/learning-teaching/assessment.html#lateness-penalties

Good Academic Practice

The University policy on Good Academic Practice applies:

https://www.st-andrews.ac.uk/students/rules/academicpractice/