Constraint Satisfaction Problems

B. Nebel, C. Becker-Asano, S. Wölfl
Wintersemester 2014/15

University of Freiburg
Department of Computer Science

Project: Part 3

Due: 9.1.2015
(9 + 11 points)

Task 1: Variable selection heuristics (3 + 3 + 3 points) In order to improve the efficiency of your CSP implementation, implement more informed strategies for choosing the next variable that is to be assigned some value. We consider static variable orderings only, namely the max cardinality and the min width ordering. Since both are *static orderings*, they should be computed before the backtracking algorithm is started.

Implement command line options that allow for choosing the variable ordering used during backtracking search:

```
python3 solver.py --heuristic=none instance.xml
python3 solver.py --heuristic=minwidth instance.xml
python3 solver.py --heuristic=maxcard instance.xml
```

In the first case, a lexicographic ordering on the names of the variables should be used.

Evaluate and discuss the influence of the variable ordering on the backtracking search: consider also the option to enforce generalized arc consistency before search. Use the instances from project 2. Moreover, we will provide some further instances, which will also involve some intensional constraints and the all-different constraint.

The results of your evaluation are expected in the file: *project03_ordering.txt*. The table summarizing the results should have the following format:

```
-----------------------------------------------------------
instance01.xml          SAT
none/none           TIMEOUT
none/maxcard         90.54s
none/minwidth        70.22s
gac3/none           TIMEOUT
gac3/maxcard         10.13s
gac3/minwidth         9.23s

instance02.xml          SAT
...
-----------------------------------------------------------
```

Task 2: Lookahead strategies (4+4+3 points) The next step of our project is to implement lookahead strategies. Notice that different to standard backtracking search, lookahead requires to keep track of the changes made at every step (specifically those made in the propagation step).

Implement the forward-checking and the real-full look ahead algorithms. Both are used without a preprocessing step. Evaluate and discuss the runtime of both look ahead schemes. If you are interested in a more fine-grained analysis it could be of interest to compare the

number of states visited during search as well as the average time spent in each state for performing the filtering of future domains.

Implement command line options that allow for selecting the look ahead scheme as follows:

```
python3 solver.py --lookahead=none instance.xml
python3 solver.py --lookahead=forwardchecking instance.xml
python3 solver.py --lookahead=realfull instance.xml
```

As in the previous project the command without any lookahead algorithm specified should still work:

```
python3 solver.py instance.xml
```

should perform the standard backtracking search.

Evaluate and discuss the influence of the lookahead scheme used during search: consider also that you can combine this with different variable orderings. Use the same instances as in the previous task.

The results of your evaluation are expected in the file: *project03_lookahead.txt*. The table should have the following format:

```
------------------------------------------------------------
instance01.xml         SAT
none/none          TIMEOUT
none/maxcard        90.54s
none/minwidth       70.22s
fcla/none          TIMEOUT
fcla/maxcard        11.23s
fcla/minwidth        9.23s
rfla/none          TIMEOUT
rfla/maxcard        11.23s
rfla/minwidth        9.23s

instance02.xml         SAT
...
------------------------------------------------------------
```