

COMP3620/6320: Artificial Intelligence

Assignment 2: KRR

Main details:

Maximum marks: 20 (COMP3620), 25 (COMP6320)
Submission deadline: April 28, 2009
Programming language: C or C++ (that compiles with g++), or Java
Files used for Q2: See “Labs and Assignments” section of course web page

Marking scheme:

- *Written:* Full marks given for a formulation that correctly answers the question. Marks deducted for incorrect formalizations or overly-complicated answers.
- *Code:* Full marks given for working, readable, commented code with correct algorithm implementation. Marks deducted for errors, unreadable or un-commented code, or incorrect algorithm implementation.

Electronic submission (only):

All written questions should be in a file `ANSWERS.pdf`. MS Word or other document formats are not accepted. \LaTeX formatting is preferred.

Please submit `ANSWERS.pdf` and source code zipped into a single file `assign2_yourname.zip` with a `README.txt` stating what each file is for and how to compile and run your code.

To submit your file:

- Log on to `partch.anu.edu.au`
- In the home directory type

```
submit <course_code> <lab_name> <file_name>
```

for example:

```
submit comp3620 lab2 assign2_yourname.zip
```

Q1 [5 pts]. Propositional (binary CSP) encoding of a variant of the Minesweeper problem.

There is a 3x3 grid. Each cell in the grid may have a mine or not. For each cell, either it is observed or not. If a cell is observed, an indicator for that cell is true iff *that* cell or one of its adjacent cells (up, down, left, right) has a mine in it (note that the “or” here is a standard “disjunctive or”, not an “exclusive or”).

- (a) Define all binary variables required for a propositional encoding of this Minesweeper problem. Give your propositional variables interpretable names like `m_1_2` for the variable indicating whether a mine is in cell (1,2). As a function of n , how many variables are needed to encode Minesweeper for a grid of dimension $n \times n$?
- (b) Define the constraints that encode that the four corners have been observed and indicate that a mine is not present.
- (c) Define the remaining constraints for this problem required to enforce consistency between all of the variables. Note that corner, edge, and middle portions of the grid must be treated separately when determining adjacency, e.g., a corner only has two adjacent cells whereas the middle has four adjacent cells. Your constraint encoding should use \Rightarrow and \Leftrightarrow and thus should *not* be in CNF.
- (d) Encode the constraints (b) and (c) in the DIMACS CNF format described below. In the DIMACS comments, show the mapping between the variable names and the DIMACS variable IDs (e.g., `m_1_2` \rightarrow 5). Provide a listing of your DIMACS file. To avoid errors, it is strongly suggested that you write a short program or script to generate the DIMACS file automatically.

DIMACS Format: An input file starts with comments (each comment line starts with `c`). The number of variables and the number of clauses is defined by the line

```
p cnf variables clauses
```

Each of the next lines specifies a clause: a positive literal is denoted by the corresponding number, and a negative literal is denoted by the corresponding negative number. The last number in a line should be zero. For example,

```
c A sample .cnf file.
p cnf 3 2
1 -3 0
2 3 -1 0
```

is a file containing two CNF clauses $v_1 \vee \neg v_3$ and $v_2 \vee v_3 \vee \neg v_1$.

Q2 [10 pts]. Implementing and evaluating a DPLL solver:

- (a) Write a DPLL solver that reads a DIMACS CNF file and attempts to show that it is unsatisfiable. You should implement unit clause propagation, but pure symbol elimination is not required. Use a variable ordering that orders variables with lower IDs (in the DIMACS format) before variables with higher IDs. Provide the source code to your program — **aside from helper code for reading DIMACS files and making queries, you must write all DPLL code yourself. Submitting DPLL code that is not your own will result in disciplinary action.**
- (b) Use your DPLL solver to perform the following six queries:
- A mine is located in corner cell (1, 1).
 - A mine is *not* located in corner cell (1, 1).
 - A mine is located in non-corner edge cell (1, 2).
 - A mine is *not* located in non-corner edge cell (1, 2).
 - A mine is located in the middle cell (2, 2).
 - A mine is *not* located in the middle cell (2, 2).

What additional CNF clause(s) did you augment your DIMACS file from Q1(d) with in order to make each of these queries? Report the results (satisfiable or unsatisfiable) for each query along with the running time.

- (c) Run your DPLL solver on `mine_22_SAT.dimacs` and `mine_22_UNSAT.dimacs`, which are 22×22 Minesweeper problems containing more than 1400 variables and 2800 clauses (see “Labs and Assignments” section of course web page for a zip archive containing these files). Provide a heuristic modification to your basic DPLL code in order to speed up its inference by at least 10% on at least one of these two DIMACS files. What heuristic modification did you choose and why? What are the pre- and post-modification inference times for both DIMACS files? Provide the code for your modification. *Hint: a speedup of at least 20% should be possible with an improved variable ordering heuristic.*
- (d) If the inference system cannot prove that a cell does not have a mine, does this mean that the cell must have a mine? Defend your answer.

Q3 [5 pts]. Encoding first-order logic axioms:

Write the following sentences in First-order Logic (predicate names you may use are provided in parentheses). (a) has been done for you as an example.

- (a) Humans are animals. (*human*(.), *animal*(.))
Answer: $\forall x \text{ human}(x) \Rightarrow \text{animal}(x)$
- (b) Scott is human. (*human*(.))
- (c) The children of humans are also human. (*human*(.), *has-child*(.,.))
- (d) Animals have exactly one head. (*animal*(.), *has-head*(.,.), $. = .$)
- (e) John has a mother who cleans some room of his. (*has-room*(.,.), *has-mother*(.,.), *cleans-room*(.,.))
- (f) Every human has someone else who loves them. (*human*(.), *loves*(.,.), $. = .$)

Q4 [5 pts, COMP6320 (Required), COMP3620 (Extra Credit)]. More complex encodings of first-order logic axioms:

Write the following sentences in First-order Logic (predicate names you may use are provided in parentheses).

- (a) A tall person is defined as a human who is over 2m in height. (*tall-person(.)*, *human(.)*, $. > .$ is an inline inequality predicate, and *height(.)* is a function providing the height of a person in meters)
- (b) The Pythagorean theorem. ($. = .$, *right-triangle(.,.,.)* where the first two arguments are the respective lengths of the two sides and the third argument is the length of the hypotenuse; use the function symbols $+(.,.)$ and $*(.,.)$ with their normal arithmetic interpretation)
- (c) Write a first-order version of Minesweeper from Q1:
- For a 3x3 problem, assume an adjacency predicate has been properly defined for this problem, i.e., using the predicate *adjacent(x, y, w, z)* to indicate that cell (x, y) is adjacent to cell (w, z) , we already have a knowledge base of assertions like *adjacent(1, 1, 1, 2)*, *adjacent(1, 1, 2, 1)*, etc... For this question, you should define an additional *domain closure* constraint indicating that the only legal values for terms (i.e., variables, constants, functions) are 1, 2, or 3.
 - Write constraints requiring that if a cell (x_1, y_1) is observed then *indicator(x₁, y₁)* is true iff there exists a mine in (x_1, y_1) or one of the cells adjacent to it. (*adjacent(x₁, y₁, x₂, y₂)*, *mine(x₁, y₁)*, *observed(x₁, y₁)*, *indicator(x₁, y₁)*); note that any variable names may be used, x_1, y_1, x_2, y_2 have just been provided for clarity of predicate meanings).
 - Write constraints indicating that each of the four corners in the 3x3 problem is observed and that they do not indicate a mine is present.