Computational Thinking 2024/25
Logic Coursework

Barnaby Martin

You should submit two files, each through a different Gradescope portal. One should be a PDF document containing your answers to all the theoretical/mathematical questions, and the other a single Python file with your code. Please name the Python file according to your username (e.g. mpll19.py).

The coding part of the coursework will be to write a SAT-solver in Python. Note that you will be restricted in some of your choices for data structures and function names. The data structure for a literal will be an integer, where a negative integer indicates the negation of the variable denoted by the corresponding positive integer. The data structure for partial assignment should be a list of literals. The data structure for a clause set should be a list of lists of literals.

1. Answer the following questions about complete sets of logical connectives, in each case justifying your answer. **[12 marks]**

   (i). Show $\{\neg, \rightarrow\}$ is a complete set of connectives. [3 marks.]

   (ii). Show $\{\rightarrow, 0\}$ is a complete set of connectives (where 0 is the constant false). [3 marks.]

   (ii). Is $\{\text{NAND}, \wedge\}$ a complete set of connectives? [3 marks.]

   (iv). Is $\{\wedge, \vee\}$ a complete set of connectives? [3 marks.]

2. Convert $(((p \rightarrow q) \rightarrow r) \rightarrow (s \rightarrow t))$ to

   (i). Conjunctive Normal Form (CNF) [4 marks.]

   (ii). Disjunctive Normal Form (DNF) [4 marks.]

   [8 marks]

3. What is the purpose of Tseitin's Algorithm? Apply Tseitin's Algorithm to turn the propositional formula $(((x_1 \wedge x_2 \wedge x_3) \rightarrow (y_1 \wedge y_2 \wedge y_3)) \vee z)$ to CNF. **[8 marks]**

4. State with justification if each of the following sentences of predicate logic is logically valid. **[8 marks]**

   (i.) $(\forall x \exists y \forall z\ E(x,y) \wedge E(y,z)) \rightarrow (\forall x \forall z \exists y\ E(x,y) \wedge E(y,z))$ [2 marks].

   (ii.) $(\forall x \exists y \exists u \forall v\ E(x,y) \wedge E(u,v)) \rightarrow (\exists u \forall v \forall x \exists y\ E(x,y) \wedge E(u,v))$ [2 marks].

   (iii.) $(\forall x \exists y \forall z\ R(x,y,z)) \rightarrow (\exists x \forall y \exists z\ R(x,y,z))$ [2 marks].

   (iv.) $((\forall x \forall y \exists z (E(x,y) \wedge E(y,z))) \rightarrow (\forall x \forall y \forall z (E(x,y) \vee E(y,z))))$ [2 marks].

5. Evaluate the given sentence on the respective relation E over domain $\{0,1,2\}$ **[8 marks]**

   (i.) $\forall x \forall y \forall z \exists w (E(x,w) \wedge E(y,w) \wedge E(z,w))$ [1 mark].

   (ii.) $\exists x \forall y \forall z \exists w (E(x,w) \wedge E(y,w) \wedge E(z,w))$ [1 mark].

   (iii.) $\forall y \exists x \forall z \exists w (E(x,w) \wedge E(y,w) \wedge E(z,w))$ [1 mark].

   (iv.) $\exists x \exists y \exists z \forall w (E(x,w) \wedge E(y,w) \wedge E(z,w))$ [1 mark].

(v.) $\forall x_1 \exists x_2 \forall y_1 \exists y_2 \forall z_1 \exists z_2 \forall z \exists y$
$E(x_1, x_2) \wedge E(x_2, w) \wedge E(y_1, y_2) \wedge E(y_2, w) \wedge E(z_1, z_2) \wedge E(z_2, w) \wedge E(z, w)$
[2 marks].

(vi.) $\forall x_1 \exists x_2 \forall y_1 \exists y_2 \forall z_1 \forall z \exists z_2 \exists y$
$E(x_1, x_2) \wedge E(x_2, w) \wedge E(y_1, y_2) \wedge E(y_2, w) \wedge E(z_1, z_2) \wedge E(z_2, w) \wedge E(z, w)$
[2 marks].

6. Write a Python function `load_dimacs`, that takes a file in DIMACS format as input and returns the clause set as a list of lists. For example the file containing the lines: "p cnf 3 2", "1 -2 0" and "-1 3 0" would become $[[1, -2], [-1, 3]]$. **[6 marks]**

7. Write a Python function `simple_sat_solve`, that takes a single argument `clause_set` as a list of lists and solves the satisfiability of the clause set by running through all truth assignments. In case the clause set is satisfiable it should output a full satisfying assignment as a list of literals; in the case the clause set is unstatisfiable the function should output `False`. For example on the input $[[1, -2], [-1, 3]]$ the function might return $[1, 3]$. **[10 marks]**

8. Write a recursive Python function `branching_sat_solve` that takes two arguments `clause_set` and `partial_assignment` as input and returns either a full satisfying assignment, if the `clause_set` is satisfiable under the partial assignment, or `False` if it is not. You should assume `clause_set` is a list of lists representing the clause set and `partial_assignment` is a list of literals. The function should solve the satisfiability of the clause set by branching on the two truth assignments for a given variable. When the function is run with an empty partial assignment it should act as a SAT-solver. **[10 marks]**

9. Write a Python function `unit_propagate` that takes a single argument `clause_set` as input and which outputs a new clause set after iteratively applying unit propagation until it cannot be applied further. **[10 marks]**

10. Write a recursive Python function `dpll_sat_solve` that takes two arguments `clause_set` and `partial_assignment` as input and solves the satisfiability of the clause set under the partial assignment by applying unit propagation before branching on the two truth assignments for a given variable (this is the famous DPLL algorithm but without pure literal elimination). In case the clause set is satisfiable under the partial assignment it should output a full satisfying assignment; if it is not satisfiable the function should return `False`. When the function is run with an empty partial assignment it should act as a SAT-solver. **[10 marks]**

11. The final 10 marks of the coursework will be allocated according to the speed of your functions `unit_propagate` and `dpll_sat_solve` running on some benchmark instances. If your code is faster than mine, you receive 10 marks; within a factor of 2, 8 marks; within a factor of 3, 6 marks; within a factor of 4, 4 marks; within a factor of 5, 2 marks. **[10 marks]**

Total marks: 100