Introduction to Artificial Intelligence

Dr. Thomas Keller
C. Büchner, C. Grundke, A. Kauffmann

University of Zürich
Spring Semester 2022

Exercise Sheet 6

Due: April 6, 2022

Points total: 20(+4 bonus) marks

Exercise 6.1 – Satisfiability and CNF — 6 marks

Consider the propositional formula $\varphi = \neg(A \Rightarrow (B \wedge \neg C)) \wedge ((A \wedge C) \vee \neg B)$.

(a) Is φ satisfiable, unsatisfiable, falsifiable, valid? Briefly justify your answer. (2 marks)

(b) Transform φ into CNF using the equivalences from Figure 7.11 in the book "Artificial Intelligence: A Modern Approach", 4th edition (global). Provide all intermediate formulas that result from applying equivalence transformations and state the name of the rule for every transformation step. (4 marks)

Exercise 6.2 – Resolution — 6 marks

Let φ be the CNF formula corresponding the the following set of clauses:

$$\{\{A, B, C\}, \{\neg A, \neg B, D\}, \{A, \neg B, C\}, \{B, C, D\}, \{\neg D, F\}, \{E, \neg F\}, \{\neg D, \neg E\}\}.$$

(a) Show that $\varphi \models \psi$ where $\psi = C \wedge \neg D$. To do so, add $\neg\psi$ to the knowledge base (i.e., φ) and then repeatedly resolve two clauses and add the resolvent to the set of clauses until you can derive the empty clause.

Hint: You may choose any order for resolving clauses. (5 marks)

(b) What is the number of required resulution steps compared to the size (number of rows) of a truth table that verifies the statement from (a)? (1 mark)

Exercise 6.3 – Model Checking — 8 marks

Consider the formula

$$\varphi = (A \vee B) \wedge (\neg A \vee B) \wedge (\neg B \vee C) \wedge (\neg C \vee \neg D) \wedge (\neg C \vee D \vee E \vee \neg F) \wedge (\neg B \vee \neg E \vee \neg F).$$

(a) Perfom DPLL and provide the recursive structure in the form of a search tree. Provide for each node (1) the cause triggering the recursion (pure symbol, unit clause, or splitting), (2) the resulting variable assignment, and (3) the remaining unsatisfied clauses. If there are multiple variables that could be chosen by any rule, break ties in alphabetical order and always assign **True** first for splitting.

Hint: A variant of DPLL that simplifies clauses during executions (which is not done in the version presented in the book) is presented in the lecture on Monday, April 4. You may use this variant to simplify the notation in your solution. (5 marks)

(b) Find a variable order with which DPLL terminates with fewer steps (when always picking **True** first). While you do not need to execute DPLL again, you must explain why your variable order is more efficient. (2 marks)

(c) With the variable order of (a), would the algorithm terminate faster if it always first picked **False**? Briefly explain your answer. (1 mark)

Exercise 6.4 – Model Checking Bonus

4 bonus marks

Note: Due to a last-minute change when preparing exercise 6.3, it is possible to solve that exercise using only the pure symbol heuristic (which was not the intention). Consequently, the subtasks (b) and (c) don't make too much sense. Since we nevertheless would like to give you the opportunity to practice DPLL and think about some other important aspects of it, we have added this bonus exercise *at a later stage. It has the same questions as 6.3 but uses a different formula φ.*

Consider the formula

$$\varphi = (A \vee B) \wedge (\neg A \vee B) \wedge (\neg B \vee C) \wedge (\neg C \vee \neg D) \wedge (\neg C \vee D \vee E) \wedge (\neg B \vee \neg E).$$

(a) Perfom DPLL and provide the recursive structure in the form of a search tree. Provide for each node (1) the cause triggering the recursion (pure symbol, unit clause, or splitting), (2) the resulting variable assignment, and (3) the remaining unsatisfied clauses. If there are multiple variables that could be chosen by any rule, break ties in alphabetical order and always assign **True** first for splitting.

Hint: A variant of DPLL that simplifies clauses during executions (which is not done in the version presented in the book) is presented in the lecture on Monday, April 4. You may use this variant to simplify the notation in your solution. (2½ bonus marks)

(b) Find a variable order with which DPLL terminates with fewer steps (when always picking **True** first). While you do not need to execute DPLL again, you must explain why your variable order is more efficient. (1 bonus mark)

(c) With the variable order of (a), would the algorithm terminate faster if it always first picked **False**? Briefly explain your answer. (½ bonus mark)

Submission rules:

- Exercise sheets must be submitted in groups of three students. Please submit a single copy of the exercises per group (only one member of the group does the submission).
- Create a single PDF file (ending `.pdf`) for all non-programming exercises. Use a file name that does not contain any spaces or special characters other than the underscore "_". If you want to submit handwritten solutions, include their scans in the single PDF. Make sure it is in a reasonable resolution so that it is readable, but ensure at the same time that the PDF size is not astronomically large. Put the names of all group members on top of the first page. Make sure your PDF has size A4 (fits the page size if printed on A4). Submit your single PDF file to the corresponding exercise assignment in MOODLE.

- For programming exercises, only create those code text files required by the exercise. Put your names in a comment on top of each file. Make sure your code compiles and test it. Code that does not compile or which we cannot successfully execute will not be graded. Create a ZIP file (ending `.zip`, `.tar.gz`, or `.tgz`; *not* `.rar` or anything else) containing the code text file(s) (ending `.py`) and nothing else. Do not use directories within the ZIP, i.e., zip the files directly.
- Do not upload several versions to MOODLE, i.e., if you need to resubmit, use the same file name again so that the previous submission is overwritten.