Exercise 1.1 (10 points)
Consider the following CNF formulas (given in set notation) over the variable set $\mathcal{V} = \{a, b, c\}$. For each formula, mark all properties that hold. Further, for each satisfiable formula, provide a complete truth assignment over $\mathcal{V}$ that satisfies it. For each unsatisfiable formula, give a complete truth assignment over $\mathcal{V}$ that falsifies it. (Hint: See slides 17-24 of the first set of slides.)

(a) $F_2 = \{\{a, \neg b, \neg c\}, \{a, b, c\}, \{\neg a\}, \{\neg b\}, \{\neg c\}\}$ [2]
◯ unsatisfiable ◯ refutable ◯ satisfiable ◯ valid
Assignment:

(b) $F_3 = \{\{a, b, \neg b\}, \{\neg a, c, \neg c\}\}$ [2]
◯ unsatisfiable ◯ refutable ◯ satisfiable ◯ valid
Assignment:

(c) $F_1 = \{\{a, b, \neg c\}, \{\neg a, \neg c\}, \{b, c\}\}$ [2]
◯ unsatisfiable ◯ refutable ◯ satisfiable ◯ valid
Assignment:

(d) $F_4 = \emptyset$ [2]
◯ unsatisfiable ◯ refutable ◯ satisfiable ◯ valid
Assignment:

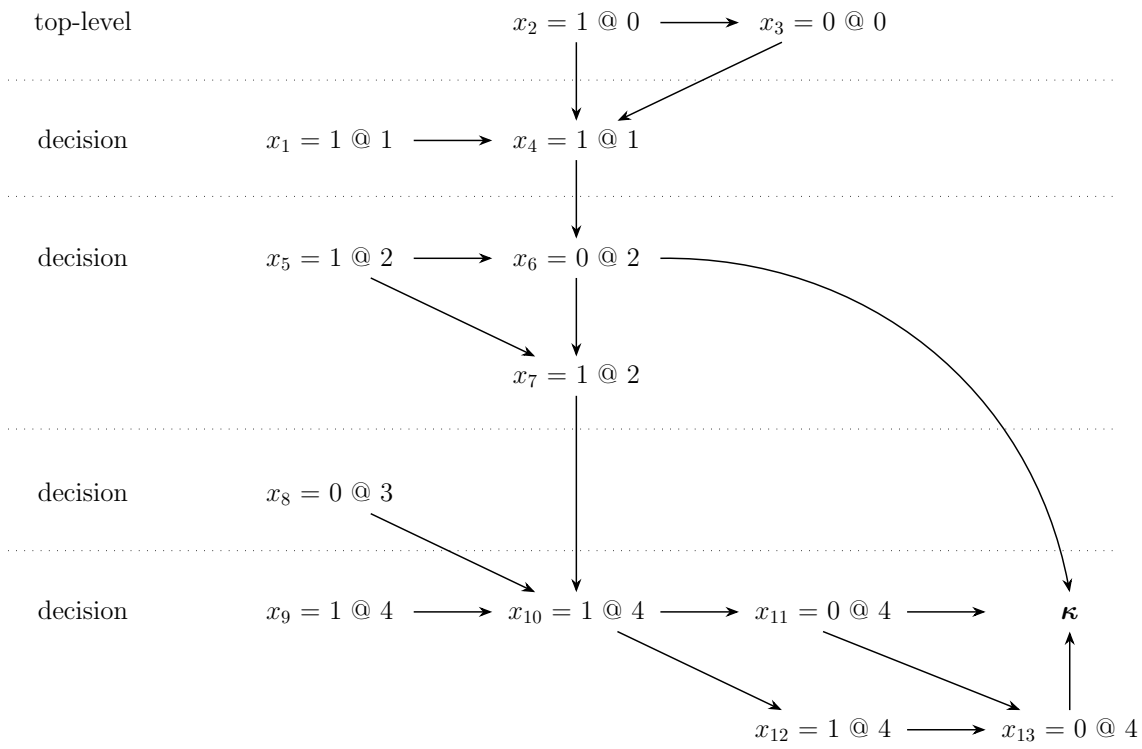
(e) $F_5 = \{\emptyset\}$ [2]
◯ unsatisfiable ◯ refutable ◯ satisfiable ◯ valid
Assignment:

Exercise 1.2 (20 points)
Consider the following CNF formula F_6:

$$(x_1) \wedge (x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2) \wedge (x_2 \vee \neg x_3 \vee x_4) \wedge (\neg x_1 \vee \neg x_3 \vee \neg x_4) \wedge (\neg x_1 \vee x_3 \vee x_4) \wedge (x_2 \vee x_3 \vee \neg x_4)$$

Execute the DPLL procedure from the lecture on F_6 (without using the pure literal rule). Write down what you do, in particular give the resulting formula whenever Boolean Constraint Propagation is finished. Assume that the algorithm always picks x_i with the smallest i not yet assigned and that it always tries the positive phase of the picked variable first. (Hint: See slides 21-23 of the second set of slides.)

Exercise 1.3 (20 points)
Look at the following implication graph, which was constructed while solving a formula F_7 using the CDCL algorithm. Try to reconstruct parts of the formula from the graph and then answer all the questions below. (Hint: See slides 12-40 of the third set of slides.)



(a) Based on the implication graph, list explicitly as many clauses of F_7 as you can. [3]

(b) What assignment is on the trail of the solver? [3]

(c) Which clause of F_7 is falsified? [2]

(d) What is the 1st UIP clause here? How did you find it? [3]

(e) Give a resolution derivation of the 1st UIP clause. (Hint: use the reverse order of the variable indices if in doubt about which step to do first.) [3]

(f) Considering the 1st UIP clause, to which decision level would the solver jump back? [3]

(g) What would be the last UIP clause in this graph? [3]

Bonus Exercise 1 (15 points (bonus))
This practical exercise is completely optional, it gives bonus points but can be ignored without any consequences.

Solve the N-Queens problem with SAT.

Implement a method QUEENS (N) that, given an argument N (where $N > 3$), generates a DIMACS file that encodes the *N-Queens problem*. The problem asks whether it is possible to place N chess queens on an $N \times N$ chessboard such that no two queens can attack each other (i.e., there are no two queens in the same row, column or diagonal). Try to solve your encoding of the problem with a SAT solver of your choice (for example CaDiCaL, Kissat, or MiniSat, etc.). Provide access to your source code (e.g. send it by mail or give a github link) and write a few sentences about your experiments. Preferably use a standard programming language (e.g. C, C++ or Python) to implement the DIMACS generator. Alternatively, you may consider using pySAT.