Technical University of Munich **Prof. J. Esparza**
Logic (IN2049), SS2026 **V. Fischer**

Exercise sheet: Polynomial time formula classes

Exercise 1: 2-CNF-SAT

A 2-CNF Formula is a Formula in CNF, where every clause contains exactly 2 literals. Show that SAT restricted to 2-CNF Formulae is decidable in Polynomial time:

Let F be a 2-CNF formula. Consider the directed graph $G = (V, E)$ where

$$V = \{p, \neg p \mid p \in \text{Vars}(F)\}$$

$$E = \left\{(\overline{L}_1, L_2), (\overline{L}_2, L_1) \mid \{L_1, L_2\}\text{is a clause of}F\right\}$$

Here $\overline{p} \coloneqq \neg p$ and $\overline{\neg p} \coloneqq p$.

Show that F is satisfiable if and only if there is no Literal L such that G has a cycle containing both L and $\overline{L}$.

Exercise 2: Horn Formulae

1. Show that for any CNF formula F one can compute in polynomial time an equisatifiable formula $G_1 \wedge G_2$, with G_1 a Horn formula and G_2 a 2-CNF formula. Justify your answer.

   (**Hint:** Consider first the case that F consists of a single clause.)

2. A *renamable Horn formula* is a CNF formula that can be turned into a Horn formula by negating (all occurrences of) some of its variables. For example,

   $$(p_1 \vee \neg p_2 \vee \neg p_3) \wedge (p_2 \vee p_3) \wedge (\neg p_1)$$

   can be turned into a Horn formula by negating p_1 and p_2.

   Given a CNF-formula F, show how to derive a 2-CNF formula G such that G is satisfiable if and only if F is a renamable Horn formula. Show moreover that one can derive a renaming that turns F into a Horn formula from a satisfying assignment for G.

3. Show that a Horn-renamable CNF formula in which no unit clauses occur has a satisfying assignment which makes in every clause all literals true except at most one.

Exercise 3: XOR-CNF-SAT

We define the xor operator as $x \oplus y \equiv \neg(x \leftrightarrow y)$. A Xor-clause is a formula of the form $L_1 \oplus L_2 \oplus \cdots \oplus L_k$, where L_i are literals. A Xor-CNF formula is a conjunction of Xor-clauses.

Prove that the SAT problem restricted to XOR-CNF formulae can be solved in Polynomial time.

Hint: Try to find a way to eliminate a variable from the formula, such that the result is equisatisfiable.

Exercise 4: Solving Sudoku with a SAT-Solver

Write a program which solves sudokus using a SAT-Solver. Start by constructing a CNF Formula F which is satisfiable if and only if the sudoku has a solution. Pass the formula to a SAT Solver (for example picosat) and extract a solution for the sudoku from a satisfying assignment. Your program should read in sudokus from stdin in the following format: Each line is a string of $81 = 9 \cdot 9$ digits representing the concatenation of all rows of the sudoku. Blank squares are represented by the digit 0. It should then print out the solution in the same format and wait for the next input. Try to make it as fast as possible. The three students with the fastest implementations will get a bar of chocalate as a reward. You may use any language you want, but need to provide building instructions and a list of dependencies. Upload your code to moodle by **Tuesday, May 5th, 23:59 CEST**.