We saw how to model problems in logic, i.e., how to solve a problem by turning the instance of the problem into an instance of the satisfiability problem.

After this tutorial you should be able to:

1. put formulas into normal forms,
2. model problems in propositional logic and solve them by reducing them to the satisfiability problem.

Problem 1. Another important normal form is *Disjunctive Normal Form* (DNF). These are formulas that are disjunctions of conjunctions of literals, i.e., of the form

$$\bigvee_i \left(\bigwedge_j l_{i,j} \right)$$

where each $l_{i,j}$ is a literal (i.e., an atom or the negation of an atom).

Which of the following are in DNF?

1. $x_1 \wedge x_2 \wedge x_3$
2. $\neg x_1 \vee \neg x_2 \vee \neg x_3$
3. $(x_1 \wedge x_2) \vee (x_1 \wedge x_3) \vee (x_2 \wedge x_3)$
4. $(\neg x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_3)$

Problem 2. Let's consider propositional logic formulas that only mention the three variables x_1, x_2, x_3.

What constraint does each the following CNF formulas express?

1. $(\neg x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_3)$
2. $(x_1 \vee x_2) \wedge (x_1 \vee x_3) \wedge (x_2 \vee x_3)$

Hint. Try the *duality trick*: take the negation of the constraint, write a DNF formula for it, and then negate that formula. The point is that some constraints are easy to write in DNF, and that it is easy to turn a negated DNF formula into CNF (since you only need to use DM and DN, no distributivity).

Hint. Alternatively, which assignment is each clause eliminating, and what do they all have in common?

Problem 3. Let N be a natural number, and let's consider propositional logic formulas over the N variables $x_1, x_2, \cdots, x_N$.

Express the following in CNF with as few clauses as possible.

1. Not all the variables have the same value.
2. All the variables have the same value.

Problem 4.

A *clique* of an undirected graph (V, E) is a set $C \subseteq V$ of vertices such that every pair of edges in C are adjacent. The size of C is the number of vertices in it (aka its cardinality). The *CLIQUE* problem is to decide, given (V, E) and K, if it has a clique of size K, and if so, to return one.

Your task is to reduce the CLIQUE problem to satisfiability for CNF formulas, i.e., find an encoding of V, E, K into a CNF formula $\Phi_{V,E,K}$ such that

- there is a solution to the instance V, E, K iff the formula $\Phi_{V,E,K}$ is satisfiable,
- you can convert every satisfying assignment of $\Phi_{V,E,K}$ into a solution for the instance V, E, K,
- and the size of $\Phi_{V,E,K}$ is bounded by a polynomial in N, K, where N is the number of vertices (i.e., $N = |V|$).

For concreteness you should assume that the vertex set V is of the form $\{1, 2, \cdots, N\}$ for some $N \geq 1$.

Hint: encode that the vertices in the clique can be ordered from 1 to K.

Explain what your encoding is and why it is correct. To do this, you should:

1. State all the variables you introduce, and say in plain language what they are supposed to represent. State all the clauses you introduce, and justify each with a short sentence in plain language saying what it captures.

   The size of your formulas $\Phi_{V,E,K}$ should be bounded above by a polynomial in N and K. For example, $O(K^3N^2)$ is a polynomial upper bound, but $O(K2^N)$ is not (it is exponential in N).

2. Find an asymptotic upper bound on the size of your encoding formula $\Phi_{V,E,K}$ in terms of N and K, and write it in big-Oh notation. You can assume that each variable can be stored in constant space.

There is a standard format for CNF formulas called the DIMACS format. You can test your encoding by running a SAT solver on it such as minisat.