After this tutorial you should be able to:

1. prove formulas are equivalent using known equivalences,
2. model and reason about facts in propositional logic,
3. put formulas into normal forms,
4. model problems in propositional logic and solve them by reducing them to the satisfiability problem.

Problem 1. What is the difference/relationship between $E \rightarrow F$ and $E \models F$?

Solution 1.

- $E \rightarrow F$ is a logical formula (in the same way that $E \wedge F$ is a logical formula)
- $E \models F$ is a mathematical statement about logical formulas.
- $E \models F$ means exactly the same thing as "$(E \rightarrow F)$ is valid".

Problem 2. Show that

$$(\neg p \wedge \neg q) \vee (\neg q \wedge \neg r) \equiv \neg(q \vee (p \wedge r))$$

using the Equivalences from the table of equivalences in the lecture slides.

Solution 2.

$$\begin{aligned}
&(\neg p \wedge \neg q) \vee (\neg q \wedge \neg r) && \\
&= (\neg q \wedge \neg p) \vee (\neg q \wedge \neg r) && \textit{Comm} \\
&= \neg q \wedge (\neg p \vee \neg r) && \textit{Distr} \\
&= \neg q \wedge \neg(p \wedge r) && \textit{DeMorgan} \\
&= \neg(q \vee (p \wedge r)) && \textit{DeMorgan}
\end{aligned}$$

Problem 3. Show that

$$(p \rightarrow q) \equiv (\neg q \rightarrow \neg p)$$

This is called the *law of the contrapositive*.

Solution 3.

$$\begin{aligned}
&p \rightarrow q && \\
&= \neg p \vee q && \textit{Cond} \\
&= q \vee \neg p && \textit{Comm} \\
&= \neg\neg q \vee \neg p && \textit{DN} \\
&= \neg q \rightarrow \neg p && \textit{Cond}
\end{aligned}$$

Problem 4. Argue why

$$\bot \models F$$

for every propositional formula F.

Solution 4. We give two similar arguments (either one is fine).

1. The statement $\bot \models F$ says for every assignment α, (*): if α makes $\bot$ true then it also makes F true. So take an arbitrary assignment α. Since α makes $\bot$ false, the implication (*) is true.
2. The statement $\bot \models F$ says for every assignment α, if α makes $\bot$ true then it also makes F true. This is exactly the same as saying (**): for every assignment α, either (a) α doesn't make $\bot$ true or (b) it makes F true (inclusive or). We will show (**) holds. To do this, take an arbitrary assignment α. But a) holds since no assignment makes $\bot$ true.

Problem 5. 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. $(\neg x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_3)$

Solution 5.

1. Yes. We can take $l_{1,j} = x_j$ for $j = 1,2,3$.
2. Yes. We can take $l_{i,1} = \neg x_i$ for $i = 1,2,3$.
3. Yes. We can take $l_{1,1} = x_1, l_{1,2} = x_2, l_{2,1} = x_1, l_{2,2} = x_3, l_{3,1} = x_2, l_{3,2} = x_3$.
4. No.

Problem 6. 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 constraint 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).

Solution 6.

1. "At most one of the variables is true".

   To see this you can reason as follows. The negation of the formula is

$$\begin{aligned} &\neg((\neg x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_3)) \\ &\equiv (\neg\neg x_1 \wedge \neg\neg x_2) \vee (\neg\neg x_1 \wedge \neg\neg x_3) \vee (\neg\neg x_2 \wedge \neg\neg x_3) && \text{DM} \\ &\equiv (x_1 \wedge x_2) \vee (x_1 \wedge x_3) \vee (x_2 \wedge x_3) && \text{DN} \end{aligned}$$

   which clearly expresses that at least two of the variables are true, the negation of which expresses that "at most one of the variables is true". We call this the duality trick.

2. "At least two of the variables are true".

   To see this you can reason as follows: it is the same formula as the previous one except that each negated variable is replaced by its variable. So, under this replacement, "at most one of the variables is true" becomes "at most one of the variables is false", which is the same as "at least two variables are true".

   Another way to see this is to look at the negation of the formula:

$$\begin{aligned} &\neg((x_1 \vee x_2) \wedge (x_1 \vee x_3) \wedge (x_2 \vee x_3)) \\ &\equiv \neg(x_1 \vee x_2) \vee \neg(x_1 \vee x_3) \vee \neg(x_2 \vee x_3) && \text{DM} \\ &\equiv (\neg x_1 \wedge \neg x_2) \vee (\neg x_1 \wedge \neg x_3) \vee (\neg x_2 \wedge \neg x_3) && \text{DM} \end{aligned}$$

   which says that "at least two variables are false". The negation of which says "at most one variable is false", which is the same as "at least two variables are true".

Problem 7. 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.

Solution 7.

1. This is the same as saying that at least one variable is true and at least one variable is false. We can write this in CNF with just two clauses as:
$$\left(\bigvee_{i\leq N} x_i\right) \wedge \left(\bigvee_{i\leq N} \neg x_i\right).$$

2. There are a number of ways to do this.
   (a) One way to do this is in DNF:
   $$\left(\bigwedge_{i\leq N} x_i\right) \vee \left(\bigwedge_{i\leq N} \neg x_i\right).$$
   But this seems annoying to translate into CNF.

   (b) Another way is to notice that you can use $\leftrightarrow$ and express "all the variables have the same value" as:
   $$\bigwedge_{1\leq i<j\leq N} (x_i \leftrightarrow x_j)$$
   which is the same as
   $$\bigwedge_{1\leq i<j\leq N} (x_i \vee \neg x_j)(\neg x_i \vee x_j)$$
   which is in CNF. This has $N(N-1)$ clauses.[1]

   (c) There is another idea which uses just N clauses. Set up two "chain reactions":
   $$(x_1 \to x_2) \wedge (x_2 \to x_3) \wedge \cdots \wedge (x_{N-1} \to x_N)$$
   and
   $$(\neg x_1 \to \neg x_2) \wedge (\neg x_2 \to \neg x_3) \wedge \cdots \wedge (\neg x_{N-1} \to \neg x_N)$$
   The first expresses that if x_1 is true, then so is x_2, and then so is x_3, etc, up to x_N. The second expresses that if x_1 is false, then so is x_2, etc., up to x_N. We can write this more precisely and compactly:
   $$\left(\bigwedge_{1\leq i<N} (\neg x_i \vee x_{i+1})\right) \wedge \left(\bigwedge_{1\leq i<N} (x_i \vee \neg x_{i+1})\right)$$
   This is in CNF and uses $2(N-1)$ clauses.

[1] Why? Well, how many pairs (i,j) are there with $1 \leq i < j \leq N$? $1+2+3+\cdots+N-1 = N(N-1)/2$. And there are two clauses for each such pair. So this gives $N(N-1)$ clauses in total.

(d) Actually it can be done with just N clauses! The formula

$$(x_1 \to x_2) \wedge (x_2 \to x_3) \wedge \cdots \wedge (x_{N-1} \to x_N) \wedge (x_N \to x_1)$$

which in CNF is:

$$\left(\bigwedge_{1 \le i < N} (\neg x_i \vee x_{i+1}) \right) \wedge (\neg x_N \vee x_1).$$

and has N clauses.

Problem 8.

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 $N \ge 1$.

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.

Solution 8.

1. Introduce variables x_n^k for $n \leq N, k \leq K$.

   The idea is that x_n^k says that vertex n is the kth vertex in the clique.

   Here are the constraints.

   (a) For every k with $1 \leq k \leq K$ we add the clause

   $$\bigvee_{n \leq N} x_n^k$$

   These clauses express that every index in $\{1, 2, \cdots, K\}$ has at least one associated vertex.

   (b) For every k, n, m with $1 \leq k \leq K$ and $1 \leq n \leq m \leq N$ we add the clause

   $$\neg x_n^k \vee \neg x_m^k$$

   These clauses express that every index in $\{1, 2, \cdots, K\}$ has at most one associated vertex.

   Thus: every index has exactly one associated vertex. But this doesn't stop different indices being associated with the same vertex...

   (c) For every k, l, n with $1 \leq k < l \leq K$ and $1 \leq n \leq N$ add the clauses

   $$\neg x_n^k \vee \neg x_n^l$$

   These clauses express that different indices can't be associated with the same vertiex.

   (d) For every k, l, i, j with $1 \leq k < l \leq K$ and $(i, j) \notin E$ we add the clause

   $$\neg x_i^k \vee \neg x_j^l$$

   These clauses express that every pair of true variables corresponds to adjacent vertices.

2. Here are the sizes of the clauses, for each type of constraint.

   (a) Each clause has size $O(N)$, there are $O(K)$ of them, so their total size is $O(NK)$.

   (b) Each clause has size $O(1)$, there are $O(N^2K)$ of them, so their total size is $O(N^2K)$.

   (c) Each clause has size $O(1)$, there are $O(K^2N)$ of them, so their total size $O(K^2N)$.

   (d) Each clause has size $O(1)$, there are $O(K^2N^2$ of them, so their total size is $O(K^2N^2)$

   Adding their total sizes together gives $O(K^2N^2)$, i.e., polynomial in N and K.

In order to test your encoding, you should translate a satisfying assignment into a solution. If α is an assignment that satisfies all the clauses, then the set

$$C := \{n \in V : \alpha(x_n^k) = 1 \text{ for some } k \leq K\}$$

is a clique of size K.

To see this, note that clauses in (a) and (b) ensure that for every $k \leq K$ exactly one variable of the form x_n^k is true in α; let $f(k) \in C$ be the vertex such that $x_{f(k)}^k$ is true. Clause (c) says that different indices are associated with different vertices in C, i.e., that $f : \{1, 2, \cdots, K\} \to C$ is injective. This means that the image of f, i.e., C, has size exactly K. Clause (d) says that different vertices in the image of f are adjacent, i.e., C forms a clique. (Note that every clause in (c) is already in (d) since we are only dealing with irreflexive graphs, i.e., $(n, n) \notin E$ for every n; this means that we only needed to write clauses (a), (b), and (d)).