

# NP-completeness of 3-SAT

## COMS20010 2020, Video 9-4

John Lapinskas, University of Bristol

# An easier problem to reduce from: 3-SAT

A **literal** is either a variable  $x$  or its negation  $\neg x$ . An **OR clause** is an OR of literals. A **CNF** formula is an AND of OR clauses. The **SAT** problem asks: “Is the input CNF formula satisfiable?”

---

We first give a special case of SAT which is still NP-complete. This will make it much easier to prove other problems are NP-hard!

The **width** of an OR clause is the number of literals it contains. A CNF formula has **width**  $k$  if **all** its OR clauses have width  $k$ . For example,

$$(x \vee \neg y \vee z) \wedge (a \vee x \vee z) \text{ has width } 3.$$

**3-SAT** asks: is the input **width-3** CNF formula satisfiable?

**Theorem:** 3-SAT is NP-complete.

Certainly  $3\text{-SAT} \in \text{NP}$ , but our proof that SAT is NP-hard breaks for 3-SAT. So to prove NP-hardness, we will reduce SAT to 3-SAT; the result then follows since SAT is NP-hard.

The **width** of an OR clause is the number of literals it contains.  
A CNF formula has **width**  $k$  if **all** its OR clauses have width  $k$ .

**3-SAT** asks: is the input **width-3** CNF formula satisfiable?

**Theorem:** 3-SAT is NP-complete.      **Goal:** Prove  $\text{SAT} \leq_c \text{3-SAT}$ .

---

Let  $F$  be the input CNF formula to SAT. We will rewrite  $F$  as an equivalent width-3 formula  $F'$ , then apply our 3-SAT oracle to  $F'$ .

Write  $F = C_1 \wedge C_2 \wedge \dots \wedge C_\ell$ , where  $C_1, \dots, C_\ell$  are OR clauses. We want to simulate each clause  $C_i$  in  $F'$ . How we do this depends on its width.

**$C_i$  has width 2:** Say  $C_i = x \vee y$ . Then we would like to replace  $C_i$  with  $x \vee y \vee \text{False}$  in  $F'$ , since this is True if and only if  $x \vee y = \text{True}$ .

But False is not a literal... Can we add a new variable which is always False in any satisfying assignment? Yes! If we add this CNF to  $F$ :

$$F_z = (\neg z_1 \vee z_2 \vee z_3) \wedge (\neg z_1 \vee z_2 \vee \neg z_3) \wedge (\neg z_1 \vee \neg z_2 \vee z_3) \wedge (\neg z_1 \vee \neg z_2 \vee \neg z_3)$$

then  $z_1$  is forced to be False: No matter what value  $z_2$  and  $z_3$  take, their literals must both be False in one of the above OR clauses. ✓

The **width** of an OR clause is the number of literals it contains.

A CNF formula has **width**  $k$  if **all** its OR clauses have width  $k$ .

**3-SAT** asks: is the input **width-3** CNF formula satisfiable?

**Theorem:** 3-SAT is NP-complete.      **Goal:** Prove  $\text{SAT} \leq_c \text{3-SAT}$ .

Let  $F = C_1 \wedge \cdots \wedge C_\ell$  be the input CNF formula to SAT. We will rewrite  $F$  as an equivalent width-3 formula  $F'$ , simulating each  $C_i$  with an equivalent width-3 CNF, then apply our 3-SAT oracle to  $F'$ .

**Width 2 clauses:**      ✓

---

**If  $C_i$  has width 1:** Say  $C_i = \neg x$ . Then we would like to replace  $C_i$  with  $\neg x \vee \text{False} \vee \text{False} \dots$  which we already know how to do!

We just need to introduce an extra copy of our always-False variable  $z_1$  (since OR clauses can't contain two copies of the same literal).      ✓

**If  $C_i$  has width 3:** We can just leave it as it is.      ✓

The **width** of an OR clause is the number of literals it contains.

A CNF formula has **width**  $k$  if **all** its OR clauses have width  $k$ .

**3-SAT** asks: is the input **width-3** CNF formula satisfiable?

**Theorem:** 3-SAT is NP-complete.      **Goal:** Prove  $\text{SAT} \leq_c \text{3-SAT}$ .

Let  $F = C_1 \wedge \cdots \wedge C_\ell$  be the input CNF formula to SAT. We will rewrite  $F$  as an equivalent width-3 formula  $F'$ , simulating each  $C_i$  with an equivalent width-3 CNF, then apply our 3-SAT oracle to  $F'$ .

**Width 1 clauses:** ✓    **Width 2 clauses:** ✓    **Width 3 clauses:** ✓

---

**If  $C_i$  has width  $k \geq 4$ :** Say  $C_i = \ell_1 \vee \cdots \vee \ell_k$ . We would like to replace

$$C_i \rightarrow (e_1 = \ell_1 \vee \ell_2) \wedge (e_2 = e_1 \vee \ell_3) \wedge \cdots \wedge (e_{k-2} = e_{k-3} \vee \ell_{k-2}) \wedge (e_{k-2} \vee \ell_k),$$

as given the values of  $\ell_1, \dots, \ell_k$ , this is satisfiable if and only if  $\ell_1 \vee \cdots \vee \ell_k = \text{True}$ . How do we implement the  $e_i$ 's? We have

$$(a = b \vee c) \text{ if and only if } (a \vee \neg b) \wedge (a \vee \neg c) \wedge (\neg a \vee b \vee c);$$

the first two clauses on the right enforce  $a = \text{False} \Rightarrow b \vee c = \text{False}$ , and the last enforces  $b \vee c = \text{False} \Rightarrow a = \text{False}$ . □

## Example of $\text{SAT} \leq_c 3\text{-SAT}$ reduction

Suppose our original SAT instance is:

$$F = u \wedge (\neg u \vee \neg v) \wedge (v \vee \neg w \vee x \vee \neg y \vee \neg z) \wedge (y \vee z) \wedge (\neg v \vee w \vee z).$$

We transform this into a 3-SAT instance as follows:

$$F' = u \wedge (\neg u \vee \neg v) \wedge (v \vee \neg w \vee x \vee \neg y \vee \neg z) \wedge (y \vee z) \wedge (\neg v \vee w \vee z).$$

First we note what we'd like the clauses to become.

## Example of $\text{SAT} \leq_c 3\text{-SAT}$ reduction

Suppose our original SAT instance is:

$$F = u \wedge (\neg u \vee \neg v) \wedge (v \vee \neg w \vee x \vee \neg y \vee \neg z) \wedge (y \vee z) \wedge (\neg v \vee w \vee z).$$

We transform this into a 3-SAT instance as follows:

$$\begin{aligned} F' = & (u \vee \text{False} \vee \text{False}) \wedge (\neg u \vee \neg v \vee \text{False}) \\ & \wedge (e_1 = v \vee \neg w) \wedge (e_2 = e_1 \vee x) \wedge (e_3 = e_2 \vee \neg y) \\ & \wedge (e_3 \vee \neg z \vee \text{False}) \wedge (y \vee z \vee \text{False}) \wedge (\neg v \vee w \vee z). \end{aligned}$$

We simulate the first instance of False with  $f_1 \dots$

## Example of $\text{SAT} \leq_c 3\text{-SAT}$ reduction

Suppose our original SAT instance is:

$$F = u \wedge (\neg u \vee \neg v) \wedge (v \vee \neg w \vee x \vee \neg y \vee \neg z) \wedge (y \vee z) \wedge (\neg v \vee w \vee z).$$

We transform this into a 3-SAT instance as follows:

$$\begin{aligned} F' = & (u \vee f_1 \vee \text{False}) \wedge (\neg u \vee \neg v \vee f_1) \\ & \wedge (e_1 = v \vee \neg w) \wedge (e_2 = e_1 \vee x) \wedge (e_3 = e_2 \vee \neg y) \\ & \wedge (e_3 \vee \neg z \vee f_1) \wedge (y \vee z \vee f_1) \wedge (\neg v \vee w \vee z) \\ & \wedge (\neg f_1 \vee a_1 \vee a_2) \wedge (\neg f_1 \vee a_1 \vee \neg a_2) \wedge (\neg f_1 \vee \neg a_1 \vee a_2) \\ & \wedge (\neg f_1 \vee \neg a_1 \vee \neg a_2). \end{aligned}$$

We simulate the first instance of False with  $f_1$ ... and the second instance with  $f_2$ .



## Example of $\text{SAT} \leq_c 3\text{-SAT}$ reduction

Suppose our original SAT instance is:

$$F = u \wedge (\neg u \vee \neg v) \wedge (v \vee \neg w \vee x \vee \neg y \vee \neg z) \wedge (y \vee z) \wedge (\neg v \vee w \vee z).$$

We transform this into a 3-SAT instance as follows:

$$\begin{aligned} F' = & (u \vee f_1 \vee f_2) \wedge (\neg u \vee \neg v \vee f_1) \\ & \wedge (e_1 = v \vee \neg w) \wedge (e_2 = e_1 \vee x) \wedge (e_3 = e_2 \vee \neg y) \\ & \wedge (e_3 \vee \neg z \vee f_1) \wedge (y \vee z \vee f_1) \wedge (\neg v \vee w \vee z) \\ & \wedge (\neg f_1 \vee a_1 \vee a_2) \wedge (\neg f_1 \vee a_1 \vee \neg a_2) \wedge (\neg f_1 \vee \neg a_1 \vee a_2) \\ & \wedge (\neg f_1 \vee \neg a_1 \vee \neg a_2) \wedge (\neg f_2 \vee a_1 \vee a_2) \wedge (\neg f_2 \vee a_1 \vee \neg a_2) \\ & \wedge (\neg f_2 \vee \neg a_1 \vee a_2) \wedge (\neg f_2 \vee \neg a_1 \vee \neg a_2). \end{aligned}$$

Finally, we simulate the equality clauses.

## Example of $\text{SAT} \leq_c 3\text{-SAT}$ reduction

Suppose our original SAT instance is:

$$F = u \wedge (\neg u \vee \neg v) \wedge (v \vee \neg w \vee x \vee \neg y \vee \neg z) \wedge (y \vee z) \wedge (\neg v \vee w \vee z).$$

We transform this into a 3-SAT instance as follows:

$$\begin{aligned} F' = & (u \vee f_1 \vee f_2) \wedge (\neg u \vee \neg v \vee f_1) \\ & \wedge (e_1 \vee \neg v \vee f_1) \wedge (e_1 \vee w \vee f_1) \wedge (\neg e_1 \vee v \vee \neg w) \\ & \wedge (e_2 \vee \neg e_1 \vee f_1) \wedge (e_2 \vee \neg x \vee f_1) \wedge (\neg e_2 \vee e_1 \vee x) \\ & \wedge (e_3 \vee \neg e_2 \vee f_1) \wedge (e_3 \vee y \vee f_1) \vee (\neg e_3 \vee e_2 \vee \neg y) \\ & \wedge (e_3 \vee \neg z \vee f_1) \wedge (y \vee z \vee f_1) \wedge (\neg v \vee w \vee z) \\ & \wedge (\neg f_1 \vee a_1 \vee a_2) \wedge (\neg f_1 \vee a_1 \vee \neg a_2) \wedge (\neg f_1 \vee \neg a_1 \vee a_2) \\ & \wedge (\neg f_1 \vee \neg a_1 \vee \neg a_2) \wedge (\neg f_2 \vee a_1 \vee a_2) \wedge (\neg f_2 \vee a_1 \vee \neg a_2) \\ & \wedge (\neg f_2 \vee \neg a_1 \vee a_2) \wedge (\neg f_2 \vee \neg a_1 \vee \neg a_2). \end{aligned}$$

Phew! We could have done this directly, without the gadgets as intermediate steps, but they made it much easier to think about...