# Theory of Computation

## Homework 2

**Problem 1.** Given a Boolean expression

$$\phi = ((a \wedge b) \Rightarrow (c \vee (d \Rightarrow e))) \wedge (a \Rightarrow f)\,.$$

**(a)** Turn $\phi$ into a CNF.

**(b)** Illustrate a Boolean circuit for CNF.

**Ans:**

**(a)** By implication, $\phi_1 \Rightarrow \phi_2 = \neg\phi_1 \vee \phi_2$,

$$\begin{aligned} \phi &= (\neg\,(a \wedge b) \vee (c \vee (d \Rightarrow e))) \wedge (a \Rightarrow f) \\ &= (\neg\,(a \wedge b) \vee (c \vee (\neg d \vee e))) \wedge (a \Rightarrow f) \\ &= (\neg\,(a \wedge b) \vee (c \vee (\neg d \vee e))) \wedge (\neg a \vee f)\,. \end{aligned}$$

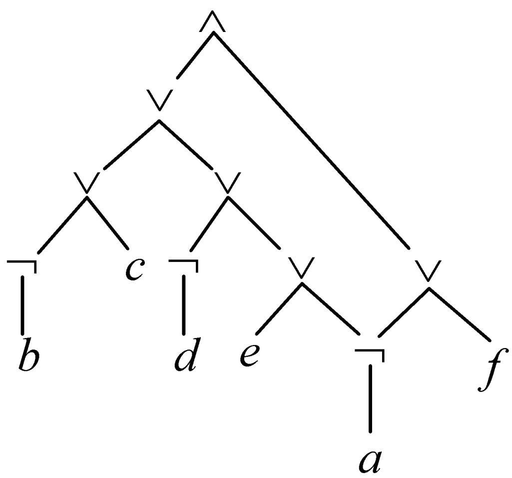
By De Morgan's laws, $\neg\,(\phi_1 \wedge \phi_2) \equiv (\neg\phi_1 \vee \neg\phi_2)$,

$$\begin{aligned} \phi &= (\neg\,(a \wedge b) \vee (c \vee (\neg d \vee e))) \wedge (\neg a \vee f) \\ &= (\neg a \vee \neg b \vee (c \vee (\neg d \vee e))) \wedge (\neg a \vee f)\,. \end{aligned}$$

Finally, the CNF of $\phi$ is

$$\phi = (\neg a \vee \neg b \vee c \vee \neg d \vee e) \wedge (\neg a \vee f)\,.$$

**(b)** A Boolean circuit is as follows:



**Problem 2.** If $f(n)$ and $g(n)$ are proper complexity functions, sketch proofs that show the following items are proper complexity functions:

**(a)** $f(g)$,

**(b)** $f + g$,

**(c)** $f \cdot g$,

**(d)** $2^g$.

*Proof.* Assume that $f$ and $g$ are computed by TMs $M_f$ and $M_g$, respectively.

**(a)** Simulate $M_g$, storing the "output" on a work tape, and then simulate $M_f$(using a different set of tapes), using that work tape as input. Note that $f(n) \geq n$ has to be satisfied.

**(b)** Simulate $M_f$, then simulate $M_g$. The outputs will be concatenated together, and so the output will be of length $f + g$.

**(c)** Simulate $M_f$, storing the "output" on a work tape. Then, repeat the following until that work tape is empty: delete the last character from the work tape, and simulate $M_g$.

**(d)** In addition to the tapes used by the simulation of $M_g$, we will use 2 extra tapes, $T_1$ and $T_2$. Begin by writing a single character to $T_1$. Then, simulate $M_g$, except, each time $M_g$ tries to output a character, instead, call the following subroutine: Copy $T_1$ over $T_2$, then append $T_2$ to $T_1$. The result is that the length of $T_1$ is doubled each time $M_g$ tries to output a character. Then, when the simulation of $M_g$ terminates, simply copy $T_1$ to the output.

It is clear that all of these run in the required time and space bounds. $\square$