CS 181 HW 1

Exercise 1.2 - Quantifiers

a) An expression $\varphi(n,k)$ such that for every natural numbers n,k, $\varphi(n,k)$ is true if and only if k divides n.

$$\varphi(n,k) = \{\varphi(n,k) \mid \forall n,k \in \mathbb{N}, \exists m \in \mathbb{N}, (n = mk)\}$$

b) An expression $\varphi(n)$ such that for every natural number n, $\varphi(n)$ is true if and only if n is a power of 3.

$$\varphi(n) = \{\varphi(n) \mid \forall n \in \mathbb{N}, (n=1) \vee \exists m \in \mathbb{N} (n = 3m \wedge \varphi(m))\}$$

Exercise 1.12 - O-notation

a) $F(n) = n$, $G(n) = 100n$

$f(n) \leq c \cdot g(n)$

$\boxed{F = O(G)}$

$\boxed{F = \Omega(G)}$

b) $F(n) = n$, $G(n) = \sqrt{n}$ → $n^{1/2}$

grow faster

$\boxed{F = \Omega(G)}$

c) $F(n) = n \log n$, $G(n) = 2^{(\log(n))^2}$

logarithmic

exponential

$G(n) = 2^{2(\log n)}$

$\log < \exp$

$\boxed{F = o(G)}$

d) $F(n) = \sqrt{n}$, $G(n) = 2^{\sqrt{\log n}}$

$F(n) = n^{1/2}$, $G(n) = 2^{(\log n)^{1/2}} = 2^{\frac{1}{2}(\log n)}$

$$\boxed{F = \omega(G)}$$

e) $F(n) = \binom{n}{\lceil 0.2n \rceil}$, $G(n) = 2^{0.1n}$ (exponential)

$\binom{n}{k} = \frac{n!}{k!(n-k)!}$, $n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$ stirling approximation

$$F(n) = \frac{n!}{(\lceil 0.2n \rceil)!\,(n - \lceil 0.2n \rceil)!}$$

$$\approx \frac{\sqrt{2\pi n}\left(\frac{n}{e}\right)^n}{\sqrt{2\pi \lceil 0.2n \rceil}\left(\frac{\lceil 0.2n \rceil}{e}\right)^{\lceil 0.2n \rceil} \sqrt{2\pi (n - \lceil 0.2n \rceil)}\left(\frac{n - \lceil 0.2n \rceil}{e}\right)^{n - \lceil 0.2n \rceil}}$$

$$\boxed{F = \omega(G)} \qquad F(n) > G(n)$$

Problem 3

If S, T are 2 finite sets, prove that there is a one to one mapping from S to T iff $|S| \leq |T|$

(domain, codomain)

Show 2 ways: 2) $|S| \leq |T| \Rightarrow f: S \to T$ is 1-1

(next page)

1) $f: S \to T$ is 1-1 $\Rightarrow |S| \leq |T|$

Proof:

1st way: We want to prove if $\exists$ a function F such that $f: S \to T$ is one to one $\Rightarrow |S| \leq |T|$

So, if f is 1-1, then $a, b \in S$, $a \neq b$.

Then $f(a) \neq f(b)$. There are only two possible cardinality cases. ① If f is 1-1, then in the case of $|S| = |T|$, there exists a function f such that $S_1, S_2, \ldots, S_n \in S$ and $t_1, t_2, \ldots, t_n \in T$;

$S_1 \neq S_2 \neq \ldots \neq S_n$, all the elements from set S are distinct from each other;

Then a possible mapping could be

$f(S_1) = t_1, f(S_2) = t_2, \ldots, f(S_n) = t_n$.

Since f is given as 1-1, then

$f(S_1) \neq f(S_2) \neq \ldots \neq f(S_n) \Rightarrow$

$t_1 \neq t_2 \neq \ldots \neq t_n$ where $|S| = |T|$

2) If f is 1-1, then in the case of $|S| < |T|$, there exists a function f such that $s_1, s_2, \ldots, s_m \in S$ and $t_1, t_2, \ldots, t_m, \ldots, t_p \in T$ where $m < p$; $s_1 \neq s_2 \neq \ldots \neq s_m$, all the elements from set S are distinct from each other;

Then a possible mapping could be

$f(s_1) = t_1, f(s_2) = t_2, \ldots, f(s_m) = t_m$ and values $t_{m+1}, \ldots, t_p$ are not mapped.

Since f is given as 1-1, then

$f(s_1) \neq f(s_2) \neq \ldots \neq f(s_m) \Rightarrow$

$t_1 \neq t_2 \neq \ldots \neq t_m$ where $|S| < |T|$

It is impossible to have $|S| > |T|$, since it violate the 1-1 mapping property, because it is possible that there are unmapped elements from set S.

As a result, if f is 1-1, then there exist a mapping such that $s_1, s_2, \ldots, s_a \in S$; $t_1, t_2, \ldots, t_b \in T$, where $a < b$ or $a = b$, $s_1 \neq s_2 \neq \ldots \neq s_a$, $t_1 \neq t_2 \neq \ldots \neq t_b$. Then $f(s_1) \neq f(s_2) \neq \ldots \neq f(s_a)$. s_a, $|S| \leq |T|$ follows.

2nd way : We want to prove if
$|S| \leq |T| \Rightarrow f : S \to T$ is 1-1

① If $|S| = |T|$, then $s_1, s_2, \ldots, s_c \in S$ and $t_1, t_2, \ldots, t_d \in T$ and $c = d$. There exists a mapping from set S to set T such that $f: S \to T$; $s_1 \neq s_2 \neq \ldots \neq s_c$ and $t_1 \neq t_2 \neq \ldots \neq t_d$, then $f(s_1) = t_1, f(s_2) = t_2, \ldots, f(s_c) = t_d$.
According to definition of 1-1, if f is 1-1, then $a, b \in S$, $a \neq b$, then $f(a) \neq f(b)$. In this case, it follows that $f: S \to T$ is 1-1 because $f(s_1) \neq f(s_2) \neq \ldots f(s_c)$

② If $|S| < |T|$, then $s_1, s_2, \ldots, s_e \in S$ and $t_1, t_2, \ldots, t_f \in T$ and $e < f$. There exists a mapping from set S to set T such that $f: S \to T$; $s_1 \neq s_2 \neq \ldots \neq s_e$ and $t_1 \neq t_2 \neq \ldots \neq t_e \neq t_{e+1} \neq \ldots \neq t_f$, then $f(s_1) = t_1, f(s_2) = t_2, \ldots, f(s_e) = t_e$
According to definition of 1-1, it follows that $f: S \to T$ is 1-1 because $f(s_1) \neq f(s_2) \neq \ldots \neq f(s_e)$

■

Exercise 3.1 Compare 4 bit numbers

In a 4 bit comparator the condition of $A > B$ which is number represented by $a_0 a_1 a_2 a_3 > b_0 b_1 b_2 b_3$ can be possible in four cases :

① If $a_3 = 1$ and $b_3 = 0$

② If $a_3 = b_3$, $a_2 = 1$ and $b_2 = 0$

③ If $a_3 = b_3$, $a_2 = b_2$, $a_1 = 1$ and $b_1 = 0$

④ If $a_3 = b_3$, $a_2 = b_2$, $a_1 = b_1$, $a_0 = 1$ and $b_0 = 0$

Pre-Formula

$$G = a_3 b_3' + (a_3 \text{ XNOR } b_3) a_2 b_2' + (a_3 \text{ XNOR } b_3)(a_2 \text{ XNOR } b_2) a_1 b_1' + (a_3 \text{ XNOR } b_3)(a_2 \text{ XNOR } b_2)(a_1 \text{ XNOR } b_1) a_0 b_0'$$

Replacing XNOR with AND, OR, NOT gate

$$a_3 \text{ XNOR } b_3 = (a_3 \vee \overline{b_3}) \wedge (\overline{a_3} \vee b_3)$$

$$a_2 \text{ XNOR } b_2 = (a_2 \vee \overline{b_2}) \wedge (\overline{a_2} \vee b_2)$$

$$a_1 \text{ XNOR } b_1 = (a_1 \vee \overline{b_1}) \wedge (\overline{a_1} \vee b_1)$$

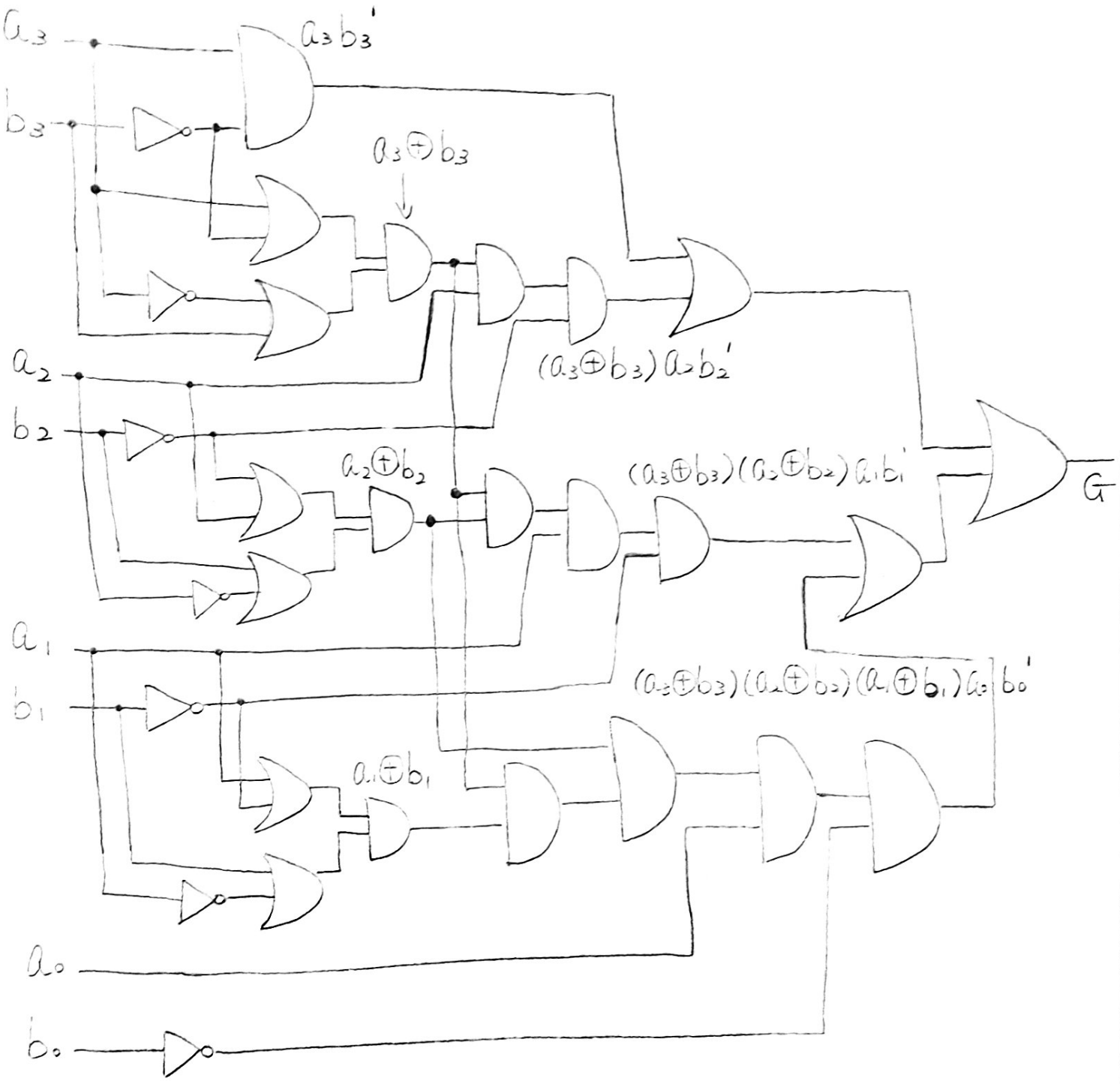
$$a_0 \text{ XNOR } b_0 = (a_0 \vee \overline{b_0}) \wedge (\overline{a_0} \vee b_0)$$

Final Formula

$$G = (a_3 \wedge \bar{b}_3) \vee \Big[\Big[\big[(a_3 \vee \bar{b}_3) \wedge (\bar{a}_3 \vee b_3)\big] \wedge a_2\Big] \wedge \bar{b}_2\Big] \vee$$

$$\Big[\big[(a_3 \vee \bar{b}_3) \wedge (\bar{a}_3 \vee b_3)\big] \wedge \big[(a_2 \vee \bar{b}_2) \wedge (\bar{a}_2 \vee b_2)\big] \wedge a_1 \wedge \bar{b}_1\Big] \vee$$

$$\big[\big[(a_3 \vee \bar{b}_3) \wedge (\bar{a}_3 \vee b_3)\big] \wedge \big[(a_2 \vee \bar{b}_2) \wedge (\bar{a}_2 \vee b_2)\big] \wedge \big[(a_1 \vee \bar{b}_1) \wedge (\bar{a}_1 \vee b_1)\big] \wedge a_0 \wedge \bar{b}_0\big]$$


a_3
b_3
$a_3 b_3'$
$a_3 \oplus b_3$
$(a_3 \oplus b_3) a_2 b_2'$
a_2
b_2
$a_2 \oplus b_2$
$(a_3 \oplus b_3)(a_2 \oplus b_2) a_1 b_1'$
G
a_1
b_1
$(a_3 \oplus b_3)(a_2 \oplus b_2)(a_1 \oplus b_1) a_0 b_0'$
$a_1 \oplus b_1$
a_0
b_0

Exercise 3.4 – AND, OR is not universal

According to the definition of monotone circuit, if $x, x' \in \{0,1\}^n$, $x_i \leq x_i'$ for every $i \in [n]$, then $C(x) \leq C(x')$, which has the property that increasing the value of any input can only increase the value of the output.

We want to show AND & OR gate are monotone through induction.

① AND Gate, when input $n = 2$ bits, the truth table are shown as follows:

x	y	AND(x, y)
0	0	0
0	1	0
1	0	0
1	1	1

So, as the number represented by x, y increase, the current output of AND(x, y) is either same or greater than the previous output AND(x', y'). The inequality shown as follows:

$$AND(0,0) \leq AND(0,1) \leq AND(1,0) \leq AND(1,1)$$

② Assume it is true when $n = k$,

b_0	b_1	b_2	....	b_K	AND($b_0, b_1, .., b_K$)
0	0	0	.....	0	0
0	0	0	....	1	0
⋮					⋮
1	1	1	.....	1	1

} all of them are zeros as long as one of the input bit is zero

$$AND(0,0,\ldots,0) \leq AND(0,0,\ldots,1) \leq \ldots \leq AND(1,1,\ldots,1)$$

all zeros — kth index — kth index — all ones — kth index

(3) When $n = k+1$ case which are $k+1$ input bits. Havever, in order for AND gate to output 1, all of the input bits need to be one. In this case, for $k+1$ input bits, the only case when AND gate output 1 is when $b_0 = 1, b_1 = 1, \ldots, b_{k+1} = 1$; all bits are 1. Therefore, AND gate only output 0 as long as there is a zero in one of the input bits which are the rest of the possible bit combinations. So, increasing the value of input, the value of output either stay the same or greater. The truth table is shown below:

b_0	b_1	....	b_k	b_{k+1}	AND($b_0, b_1, \ldots, b_k, b_{k+1}$)
0	0	...	0	0	0
0	0	...	0	1	0
					⋮ } all outputs are zero.
1	1	...	1	1	1

Since we know if all k bits are 1 will output 1, then it is only possible $k+1$ th index bit is 1 in order to output 1 for AND gate. So, the inequality holds:

$$\text{AND}(0,0,\ldots,0) \leq \text{AND}(0,0,\ldots,1) \leq \ldots \leq \text{AND}(1,1,\ldots,1)$$

($k+1$ th index) ($k+1$ th index)

all zeros — all ones — $k+1$ th index

OR gate induction

① OR Gate, when input bits $n=2$:

x	y	OR(x,y)
0	0	0
0	1	1
1	0	1
1	1	1

As the number represented by x, y increase, the current output of OR(x,y) is either same or greater than the previous output OR(x',y') The inequality shown as follows:

$$OR(0,0) \leq OR(0,1) \leq OR(1,0) \leq OR(1,1)$$

② Assume it is true when $n=k$

b_0	b_1	b_2	...	b_K	$OR(b_0, b_1, \ldots, b_K)$
0	0	0	...	0	0
0	0	0	...	1	1
		⋮			⋮
1	1	1	...	1	1

} all of them are ones as long as one of the input bit is 1

$$OR(0,0,\ldots,0) \leq OR(0,0,\ldots,1) \leq \ldots \leq OR(1,1,\ldots,1)$$

Kth index

all zeros

all ones

(3) when $n=k+1$ case which are $k+1$ input bits. However, the only case for OR gate to output 0 is when all of the input bits are zero. In this case, for $k+1$ input bits, the only case when OR gate output 0 is when $b_0=0, b_1=0, \ldots, b_{k+1}=0$; all bits are 0. Therefore, OR gate only output 1 as long as there is a 1 in one of the input bits which are the rest of the possible bit combinations. So, increasing the value of input, then the value of output either stay same or become greater.

b_0 b_1 . . . . b_k b_{k+1}	$OR(b_0, b_1, \ldots, b_{k+1})$
0 0 . . . . 0 0	0
0 0 . . . . 0 1	1
⋮	⋮ } all outputs are one
1 1 . . . . 1 1	1

Since we know if all k bits are 0 will output 0, then it is only possible $k+1$ th index bit is 0 in order to output 0 for OR gate. So, inequality holds:

$OR(0,0,\ldots,0) \leq OR(0,0,\ldots,1) \leq \ldots \leq OR(1,1,\ldots,1)$

($k+1$ th index)

all zeros

all ones

As a result, AND, OR gates are monotonic.
Therefore, any n bit input circuit that built based just AND & OR gates are monotone. So, monotone circuits can only implement monotone functions.
Therefore, it cannot implement non-montonic functions.
Here is an example:

If a circuit contain a NAND gate such as output $= (\overline{A \wedge B}) \vee C$, it is impossible to implement with {AND, OR, 0, 1} because a NAND gate require both a AND gate and NOT gate to be implemented.

$$\overline{A \wedge B} = \neg(A \wedge B)$$

Therefore, {AND, OR, 0, 1} is not universal. ∎

Exercise 3.9 – Lookup is universal

Given $\{Lookup_1, 0, 1\}$ st.

$Lookup_1 = \{0,1\}^3 \to \{0,1\}$ st. $Lookup_1(a,b,c) = \begin{cases} a & \text{if } c = 0 \text{ (off)} \\ b & \text{if } c = 1 \text{ (on)} \end{cases}$

We want to prove $\{Lookup_1, 0, 1\}$ is universal

Proof: According to Theorem 3.12 from textbook, NAND is a universal operation such that for every boolean circuit C of s gates, there exists a NAND circuit C' of at most 3s gates that computes the same function as C.

We can show $\{Lookup_1, 0, 1\}$ is universal by showing it can implement AND, OR, NOT gate. Since we can replace NAND gate with AND, OR, NOT gates or vice versa.

NOT gate:

x	$Lookup_1(1,0,x) = Lookup_1(a,b,c)$
0	1
1	0

$$NOT(x) = Lookup(1,0,x)$$

AND gate

x	y	AND(x, y)
0	0	0
0	1	0
1	0	0
1	1	1

$$Lookup(a,b,c) = \begin{cases} a & \text{if } c = 0 \\ b & \text{if } c = 1 \end{cases}$$

$$AND(x,y) = Lookup(0, x, y)$$

OR gate

x	y	OR(x, y)
0	0	0
0	1	1
1	0	1
1	1	1

$$OR(x,y) = Lookup(x, 1, y)$$

Therefore, we can represent AND, OR, NOT gate with Lookup function. NAND is the NOT of AND which is NAND(a, b) = NOT(AND(a, b)). So, we compute AND, OR, NOT by composing only the NAND function. Then, we can also use NAND to compose AND, OR, NOT gate that are represented by the Lookup function. NAND gate is universal according to Theorem 3.12, since we can represent AND, OR, NOT gate as follows:

(1) NOT(a) = NAND(a, a)

(2) AND(a, b) = NAND(NAND(a, b), NAND(a, b))

(3) OR(a, b) = NAND(NAND(a, a), NAND(b, b))

As a result, NAND is universal, as well as {AND, OR, NOT}. Then {Lookup, 1, 0} is universal as follows