Problem Sheet 7

Henri Sota
h.sota@jacobs-university.de
Computer Science 2022

November 1, 2019

Problem 7.1

In order to prove that the two elementary boolean functions, $\rightarrow$ and $\neg$ are universal (sufficient to express all possible boolean functions), we need to show that they are functionally complete, which means that all formulas in propositional logic can be rewritten to an equivalent form that uses only the set $\{\rightarrow, \neg\}$. We need to show that we can produce $\wedge$, $\vee$ and $\neg$ with them.

- $\wedge$ can be rewritten in an equivalent form using $\rightarrow$ and $\neg$:

$$P \wedge Q = (P \rightarrow \neg Q) \rightarrow \neg(P \rightarrow \neg Q)$$

Because $(P \rightarrow \neg Q)$ is equivalent to NAND operator, which is universal (functionally complete) on its own. Proof that the formula above is equivalent:

P	**Q**	$\neg$**Q**	**P** $\rightarrow \neg$**Q**	$\neg$(**P** $\rightarrow \neg$**Q**)	(**P** $\rightarrow \neg$**Q**) $\rightarrow \neg$(**P** $\rightarrow \neg$**Q**)	**P** $\wedge$ **Q**
0	0	1	1	0	0	0
0	1	0	1	0	0	0
1	0	1	1	0	0	0
1	1	0	0	1	1	1

- $\vee$ can be rewritten in an equivalent form using $\rightarrow$ and $\neg$:

$$P \vee Q = (P \rightarrow \neg P) \rightarrow \neg(Q \rightarrow \neg Q)$$

Because $(P \rightarrow \neg Q)$ is equivalent to NAND operator, which is universal (functionally complete) on its own. Proof that the formula above is equivalent:

P	**Q**	$\neg$**P**	$\neg$**Q**	**P** $\rightarrow \neg$**P**	**Q** $\rightarrow \neg$**Q**	$\neg$(**Q** $\rightarrow \neg$**Q**)	(**P** $\rightarrow \neg$**P**) $\rightarrow \neg$(**Q** $\rightarrow \neg$**Q**)	**P** $\vee$ **Q**
0	0	1	1	1	1	0	0	0
0	1	1	0	1	0	1	1	1
1	0	0	1	0	1	0	1	1
1	1	0	0	0	0	1	1	1

- $\neg$ can be rewritten in an equivalent form using only $\neg$ (itself):

$$\neg P = \neg P$$

Therefore $\rightarrow$ and $\neg$ are universal.

P	**¬P**
0	1
1	0

Problem 7.2

$$\varphi(P, Q, R, S) = (\neg P \vee Q) \wedge (\neg Q \vee R) \wedge (\neg R \vee S) \wedge (\neg S \vee P)$$

a) There are only 2 interpretations of the variables P, Q, R, S that satisfy φ
First Interpretation: $P = 0, Q = 0, R = 0, S = 0$
Second Interpretation: $P = 1, Q = 1, R = 1, S = 1$

P	**Q**	**R**	**S**	**¬P ∨ Q**	**¬Q ∨ R**	**¬R ∨ S**	**¬S ∨ P**	**φ(P, Q, R, S)**
0	0	0	0	1	1	1	1	1
0	0	0	1	1	1	1	0	0
0	0	1	0	1	1	0	1	0
0	0	1	1	1	1	1	0	0
0	1	0	0	1	0	1	1	0
0	1	0	1	1	0	1	0	0
0	1	1	0	1	1	0	1	0
0	1	1	1	1	1	1	0	0
1	0	0	0	0	1	1	1	0
1	0	0	1	0	1	1	1	0
1	0	1	0	0	1	0	1	0
1	0	1	1	0	1	1	1	0
1	1	0	0	1	0	1	1	0
1	1	0	1	1	0	1	1	0
1	1	1	0	1	1	0	1	0
1	1	1	1	1	1	1	1	1

b) In order to obtain a DNF from our truth table, we have to look at the rows that where the result is 1. In this case, 1st row and 16th row have 1 as their result. For each interpretation, we negate the variables that are 0 on that interpretation and logically combine them with $\wedge$ operator. We then combine interpretations with $\vee$ operator.

$$\chi = (\neg P \wedge \neg Q \wedge \neg R \wedge \neg S) \vee (P \wedge Q \wedge R \wedge S)$$

c) Utilizing boolean expression equivalence laws to transform the CNF representation $(\neg P \vee Q) \wedge (\neg Q \vee R) \wedge (\neg R \vee S) \wedge (\neg S \vee P)$ into the DNF representation $(\neg P \wedge \neg Q \wedge \neg R \wedge \neg S) \vee (P \wedge Q \wedge R \wedge S)$.

$$
\begin{aligned}
\varphi &= (\neg P \vee Q) \wedge (\neg Q \vee R) \wedge (\neg R \vee S) \wedge (\neg S \vee P) \\
&= ((\neg P \vee Q) \wedge (\neg Q \vee R)) \wedge (\neg R \vee S) \wedge (\neg S \vee P)[1] \\
&= (((\neg P \vee Q) \wedge \neg Q) \vee ((\neg P \vee Q) \wedge R))) \wedge (\neg R \vee S) \wedge (\neg S \vee P)[2] \\
&= ((\neg P \wedge \neg Q) \vee (\neg Q \wedge \neg Q) \vee (\neg P \wedge R) \vee (Q \wedge R)) \wedge (\neg R \vee S) \wedge (\neg S \vee P)[2] \\
&= ((\neg P \wedge \neg Q) \vee (\neg P \wedge R) \vee (Q \wedge R)) \wedge (\neg R \vee S) \wedge (\neg S \vee P)[3] \\
&= (((\neg P \wedge \neg Q) \wedge (\neg R \vee S)) \vee ((\neg P \wedge R) \wedge (\neg R \vee S)) \vee ((Q \wedge R) \wedge (\neg R \vee S))) \wedge (\neg S \vee P)[1][2] \\
&= ((\neg P \wedge \neg Q \wedge \neg R) \vee (\neg P \wedge \neg Q \wedge S) \vee (\neg P \wedge R \wedge \neg R) \vee (\neg P \wedge R \wedge S) \\
&\vee (Q \wedge R \wedge \neg R) \vee (Q \wedge R \wedge S)) \wedge (\neg S \vee P)[2] \\
&= ((\neg P \wedge \neg Q \wedge \neg R) \vee (\neg P \wedge \neg Q \wedge S) \vee (\neg P \wedge R \wedge S) \vee (Q \wedge R \wedge S)) \wedge (\neg S \vee P)[3] \\
&= ((\neg P \wedge \neg Q \wedge \neg R) \wedge (\neg S \vee P)) \vee ((\neg P \wedge \neg Q \wedge S) \wedge (\neg S \vee P)) \\
&\vee ((\neg P \wedge R \wedge S) \wedge (\neg S \vee P)) \vee ((Q \wedge R \wedge S) \wedge (\neg S \vee P))[1][2]
\end{aligned}
$$

$$(\neg P \wedge \neg Q \wedge \neg R) \wedge (\neg S \vee P) = (\neg P \wedge \neg Q \wedge \neg R \wedge \neg S) \vee (\neg P \wedge \neg Q \wedge \neg R \wedge P) = (\neg P \wedge \neg Q \wedge \neg R \wedge \neg S) \vee 0[4]$$

$$(\neg P \wedge \neg Q \wedge S) \wedge (\neg S \vee P) = (\neg P \wedge \neg Q \wedge S \wedge \neg S) \vee (\neg P \wedge \neg Q \wedge S \wedge P) = 0 \vee 0 = 0[3]$$

$$(\neg P \wedge R \wedge S) \wedge (\neg S \vee P) = (\neg P \wedge R \wedge S \wedge \neg S) \vee (\neg P \wedge R \wedge S \wedge P) = 0 \vee 0 = 0[3]$$

$$(Q \wedge R \wedge S) \wedge (\neg S \vee P) = (Q \wedge R \wedge S \wedge \neg S) \vee (Q \wedge R \wedge S \wedge P) = 0 \vee (P \wedge Q \wedge R \wedge S)[4]$$

1 Associativity

2 Distributivity

3 $x \wedge \neg x = 0$

4 Identity

Substituting the 4 expressions above into the expansion above them:

$$
\begin{aligned}
\varphi(P, Q, R, S) &= (\neg P \wedge \neg Q \wedge \neg R \wedge \neg S) \vee 0 \vee 0 \vee (P \wedge Q \wedge R \wedge S) \\
&= (\neg P \wedge \neg Q \wedge \neg R \wedge \neg S) \vee (P \wedge Q \wedge R \wedge S)
\end{aligned}
$$

The last line is the same expression as the DNF representation we found on part b).

Problem 7.3

a)

```
-- a) Code variables
-- Function takes a BoolExpr and returns a list of Variables
-- Pattern match the BoolExpr for every case that BoolExpr is defined for
-- T, F and Var BoolExpr are the base cases
-- Not BoolExpr calls itself again with argument BoolExpr
-- And and Or return the sorted union of the recursive calls of variables
variables :: BoolExpr -> [Variable]
variables T = []
variables F = []
variables (Var v)           = [v]
variables (Not v)           = variables v
variables (And e1 e2)       = sort(union (variables e1) (variables e2))
variables (Or e1 e2)        = sort(union (variables e1) (variables e2))
```

b)
```
-- b) Code subsets
-- Function takes a list of variables and returns the powerset of the list
-- Base case is the empty list which returns a list with the empty list as the
-- only element
-- x:xs divides the list into its head and tail
-- It calls itself with the tail and appends the map of the subset with its
    head
-- function
subsets :: [Variable] -> [[Variable]]
subsets [] = [[]]
subsets (x:xs) = subsets xs ++ (map (x:) (subsets xs))

-- b) Code truthtable
-- Function takes a BoolExpr and returns the entire truth table of the BooExpr
-- using a list of tuples made of each interpretation and its evaluation
-- Function maps each element of the powerset of the variables of the BoolExpr
-- to produce a tuple made of the element and the evaluation of the BoolExpr
-- with the element as its interpretation
-- Mapping all of the interpretations produces a list of tuples that is the
-- whole truth table of the BoolExpr
truthtable :: BoolExpr -> [([Variable], Bool)]
truthtable expr = map (\x -> (x, evaluate expr x)) (subsets (variables expr))
```

In order to test these functions, I've used multiple calls to these with different inputs such as Variables, Boolean Expressions:

```
print(variables T)
print(variables (And (Var 'a') (Or (Var 'c') (Var 'b'))))
print(variables (And (Var 'a') (Or (Var 'a') (Var 'a'))))
print(subsets (variables (And (Var 'a') (Or (Var 'c') (Var 'b')))))
print(truthtable (And (Var 'a') (Or (Var 'c') (Var 'b'))))
```

which produced as expected:

""

""

"abc"

"a"

["","c","b","bc","a","ac","ab","abc"]

[("",False),("c",False),("b",False),("bc",False),("a",False),("ac",True),("ab",True),("abc",True)]