Tyler Pranger

Professor Arias

Language Study Lab 1

7 February 2020

1.

P Q	P NAND Q	P NOR Q	P ≡ Q
0 0	1	1	1
0 1	1	0	0
1 0	1	0	0
1 1	0	0	1

2.

a.

P Q	P -> Q	NOT P OR Q	≡
0 0	1	1	1
0 1	1	1	1
1 0	0	0	1
1 1	1	1	1

b.

P Q R	A. R OR NOT P	B. Q -> A	P -> B
0 0 0	1	1	1
0 0 1	1	1	1
0 1 0	1	1	1
0 1 1	1	1	1
1 0 0	0	1	1
1 0 1	1	1	1
1 1 0	0	0	0
1 1 1	1	1	1

c.

P Q	P OR Q	P AND Q	(P OR Q) -> (P AND Q)
0 0	0	0	1
0 1	1	0	0
1 0	1	0	0
1 1	1	1	1

3. P AND NOT Q ≡ P`Q
4.

P	Q	->	P NAND Q	P NOR Q
0	0	1	1	1
0	1	1	1	0
1	0	0	1	0
1	1	1	0	0

5.There are 16 boolean functions for 2 arguments.If a function does not depend on first argument, then it is equivalent to a function with one argument, and the total number of boolean functions with one argument is equal to 2^2 = 4. Also, the number of functions that do not depend on second argument is equal to 4. A function that does not depend on both arguments will be either a true or false function. Therefore, the total number of boolean functions that do not depend on both arguments is equal to **<u>2.</u>**

6.

F	AND	NOT ->	P	`Q -> P	Q	XOR	OR	NOR	≡	`Q	Q -> P	`P	P -> Q	NAND	T
0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

7.

P	Q	XOR
0	0	0
0	1	1
1	0	1
1	1	0

The function XOR is commutative, but not associative

8. a = (`P * `Q * R) + (P * `Q * `R) + (P * Q * `R) + (P * Q * R)
b = (`P * `Q * `R) + (`P * `Q * R) + (`P * Q * `R)

9.

A = (P + Q + R)(P + `Q + R)(P + `Q + `R)
B = (p + `q + `r)(`p + q + r)(`p + q + `r)(`p + `q + r)(`p + `q + `r)
C = (x + y + c)(x + `y + `c)(`x + y + `c)(`x + `y + c)

10.

a)

	00	01	11	10
00	0	1	1	1
01	1	1	1	1
11	1	1	0	1
10	1	1	1	1

b)

	00	01	11	10
00	1	1	1	1
01	1	1	0	1
11	1	0	0	0
10	1	1	0	0

c)

	00	01	11	10
00	0	1	0	1
01	1	0	1	1
11	0	1	1	1
10	1	0	1	0

d) PQR -> S

	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	1	1	0	0
10	1	1	1	1

e)

	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	0	0	0	0
10	1	1	0	0

11.
a) Not all of them are needed.
b)5. Not all of them are needed
c)1. All of them are used.
d)3. All of them are used.
e)1. All of them are used.
12.
Q→P:

Q.-> P	0	1
0	**1**	0
1	***1***	***1***

This K map shows two implicants for the function q→p . The larger implicant corresponds to q. This implicant covers the bottom two points of the map, which both have 1s. The smaller implicant, qp, covers the point p = 0 and q = 0. Since these two implicants together cover all the points that have value 1, their sum, q + qp, is an equivalent expression for q → p, then therefore (q → p) ≡ (q + qp).

13.
a) 0
b) 1
c) 6
d)1
e)2

14. a) (`p + `q + `r + `s)(p + q + r + s)
b) (p + q + `r + s)(`p + q + r + s)(p + q + r + s)(p + `q + r + s)(p + q + r + `s)(p + `q + r + `s)
c) (`p + `q + `r + `s)(p + q + `r + `s)(`p + q + `r + s)(p + `q + `r + s)(`p + `q + r + s)
(p + `q + r + `s)
d) (p + q + r + s)(p + q + r + `s)
e) (p + q + `r + `s)(p + q + `r + s)(p + q + r + s)(p + `q + r + s)(p + q + r + `s)(p + `q + r + `s)

15. a) 24 b) 8 c) 8 d) 2
Prime implicants: f(`P`Q, PQ, `R`S, RS)

16.A, B and C are all tautologies
See attachment under questions for work

17.
A tautology is when a function ends up outputting all values as true. Let's say we have two expressions, a and b, which are both equivalent. If x <-> y is a tautology. This statement is equivalent to:

(x <-> y) = (x -> y) AND (y <- x)

An algorithm that can be used to solve the tautology problem for this expression can be used to check whether the two statements are equivalent or not, based on checking if they are tautologies. If (x <-> y) is a tautology, then the two statements are therefore equivalent, since they have the same output. If (x <-> y) is not a tautology, then the statements are not equivalent.

b) In order for a statement to be satisfiable, there must be at least one case where it is true. By definition, a tautology only exists when every output is true, so every tautology is satisfiable. Therefore, the algorithm only has to check for whether or not (x <-> y) is a tautology or not. This will therefore answer if the expression is satisfiable.

18. See attachment
19.
1. Reflexivity of equivalence : (x + y) ≡ (x + y).
2. Commutative law for equivalence : ((x + y) ≡ yz) ≡ (yz ≡ (x + y)).
3. Transitive law for equivalence : (((x+y) ≡ yz) AND (yz ≡ (x))) → ((x+y) ≡ (x)).
4. Equivalence of the negations : ((x+y) ≡ yz) ≡ ((x+y) ≡ yz).
5. The commutative law for AND : (x+y)yz ≡ yz(x+y).
6. The associative law for AND : (x+y)(yz(x)) ≡ ((x+y)yz)(x).
7. The commutative law for OR : ((x+y) + yz) ≡ (yz + (x+y)).
8. The associative law for OR : ((x+y) + (yz + (x))) ≡ (((x+y) + yz) + (x)).

9. The distributive law of AND over OR : (x+y)(yz + (x)) ≡ ((x+y)yz + (x+y)(x)).
10. 1(TRUE) is the identity for AND : ((x+y) AND 1) ≡ (x+y).
11. 0(FALSE) is the identity for OR : (x+y) OR 0 ≡ (x+y).
12. 0 is the annihilator for AND : ((x+y) AND 0) ≡ 0.
13. Elimination of double negations : (NOT NOT (x+y)) ≡ (x+y).
14. The distributive law for OR over AND : ((x+y) + yz(x)) ≡ (((x+y) + yz)((x+y) + (x))).
15. 1 is the annihilator for OR : (1 OR (x+y)) ≡ 1.
16. Idempotence of AND : (x+y)(x+y) ≡ (x+y).
17. Idempotence of OR : (x+y) + (x+y) ≡ (x+y).
18. Subsumption.
    (a) ((x+y) + (x+y)yz) ≡ (x+y).
    (b) (x+y)((x+y) + yz) ≡ (x+y).
19. Elimination of certain negations.
    (a) (x+y)((x+y) + yz) ≡ (x+y)yz.
    (b) (x+y) + (x+y)yz ≡ (x+y) + yz.
20. DeMorgan0 s laws.
    (a) NOT ((x+y)yz) ≡ (x+y) + yz.
    (b) NOT ((x+y) + yz) ≡ (x+y)yz.
    (c) (NOT ((x+y)1(x+y)2...(x+y)k)) ≡ ((x+y)1 + (x+y)2 + ... + (x+y)k).
    (d) (NOT ((x+y)1 + (x+y)2 + ... + (x+y)k)) ≡ ((x+y)1(x+y)2...(x+y)k).
21. (((x+y) → yz) AND (yz → (x+y))) ≡ ((x+y) ≡ yz).
22. ((x+y) ≡ yz) → ((x+y) → yz).
23. Transitivity of implication : (((x+y) → yz) AND (yz → (x))) → ((x+y) → (x)).
24. Implication with AND and OR:
    (a) ((x+y) → yz) ≡ ((x+y) + yz).
    (b) ((x+y)1(x+y)2...(x+y)n → yz) ≡ ((x+y)1 + (x+y)2 + ... + (x+y)n + yz).

20.

a) pq + rs into (p + r)(p + s)(q + r)(q + s)
   pq + rs - Distributive law
   (pq + r)(pq + s) - Distributive Law again
   (pq + r) = (p + r)(q + r)
   (pq + s) = (p + s)(q + s)
   Substitution law
   <u>(p + r)(p + s)(q + r)(q + s)</u>

b) pq + p`qr
   p(q + `qr) Law of Boolean Algebra
   p(q + r)

21. p(p + q) into p   Distribute
pp + pq   Indempotence Law
p + pq   Distributive law
p(q + 1)   Annihilator 1
p(1)   Simplify
p

22. `(pq + `(pr)) = `(pq) * `(pr)
(`p + `q) * pr
pr`p + pr`q
0 + pr`q
Pr`q

`(`p + q`(r + `s)) = `(`p) * `(q`(r + `s))
p * (`q + `(`(r + `s)))
p * (`q  + r + `s)
p`q + pr + p`s

23.

a) w`x + w`xy + `(zx)w = w`x(1 + y) + (`z`x)w
w`x(1) + w`x`z (1 + y is equivalent to 1)
w`x(1 + `z)
w`x(1)
w`x

b) See picture attached

a)

			A	B	
r	p	q	$p \wedge q \wedge r$	p OR q	$A \to B$
0	0	0	0	0	1
0	0	1	0	1	1
1	0	0	0	0	1
1	0	1	0	1	1
0	1	0	0	1	1
0	1	1	0	1	1
1	1	0	0	1	1
1	1	1	1	1	1

d)

			A	B	C	D	
p	q	r	q OR r	$p \equiv A$	p AND r	$q \to C$	$B \to D$
0	0	0	0	1	0	1	1
0	0	1	1	0	0	1	1
0	1	0	1	0	0	0	1
0	1	1	1	0	0	0	1
1	0	0	0	0	0	1	1
1	0	1	1	1	1	1	1
1	1	0	1	1	0	0	0
1	1	1	1	1	1	1	1

c)

p	q	$p \to q$	$(p \to q) \to p$
0	0	1	1
0	1	1	1
1	0	0	1
1	1	1	1

b)

			A	B	C	D	
p	q	r	$p \to q$	$q \to r$	A AND B	$p \to r$	$C \to D$
0	0	0	1	1	1	1	1
0	0	1	1	1	1	1	1
0	1	0	1	0	0	1	1
0	1	1	1	1	1	1	1
1	0	0	0	1	0	0	1
1	0	1	0	1	0	1	1
1	1	0	1	0	0	0	1
1	1	1	1	1	1	1	1

1.

p	q	$p \equiv p$
0	0	1
0	1	1
1	0	1
1	1	1

4.

		A	B	
p	q	$p \equiv q$	$\bar{p} \equiv \bar{q}$	$A \equiv B$
0	0	1	1	1
0	1	0	0	1
1	0	0	0	1
1	1	1	1	1

2.

		A	B	
p	q	$p \equiv q$	$q \equiv p$	$B \equiv A$
0	0	1	1	1
0	1	0	0	1
1	0	0	0	1
1	1	1	1	1

5.

		A	B	
p	q	pq	qp	$A \equiv B$
0	0	0	0	1
0	1	0	0	1
1	0	0	0	1
1	1	1	1	1

~~3.~~

~~p~~	~~q~~	~~$q \equiv q$~~
~~0~~	~~0~~	~~1~~
~~0~~	~~1~~	~~0~~
~~1~~	~~0~~	~~0~~
~~1~~	~~1~~	~~1~~

3.

			A	B	C	D	
p	q	r	$p \equiv q$	$q \equiv r$	A AND B	$p \equiv r$	$C \to D$
0	0	0	1	1	1	1	1
0	0	1	1	0	0	0	1
0	1	0	0	0	0	1	1
0	1	1	0	1	0	0	1
1	0	0	0	1	0	0	1
1	0	1	0	0	0	1	1
1	1	0	1	0	0	0	1
1	1	1	1	1	1	1	1

6.

	A	B	C	D	
p q r	q r	p(A)	p q	r(C)	B ≡ D
0 0 0	0	0	0	0	1
0 0 1	0	0	0	0	1
0 1 0	0	0	0	0	1
0 1 1	1	0	0	0	1
1 0 0	0	0	0	0	1
1 0 1	0	0	0	0	1
1 1 0	0	0	1	0	1
1 1 1	1	1	1	1	1

9.

	A	B	C	D		
p q r	q + r	p(A)	p q	p r	C or D	B ≡ D
0 0 0	0	0	0	0	0	1
0 0 1	1	0	0	0	0	1
0 1 0	1	0	0	0	0	1
0 1 1	1	0	0	0	0	1
1 0 0	0	0	0	0	0	1
1 0 1	1	1	0	1	1	1
1 1 0	1	1	1	0	1	1
1 1 1	1	1	1	1	1	1

7.

	A	B	
p q	p OR q	q OR p	A ≡ B
0 0	0	0	1
0 1	1	1	1
1 0	1	1	1
1 1	1	1	1

10.

	A	B	C	
p q	p	1	A and B	C ≡ A
0 0	0	1	0	1
0 1	0	1	0	1
1 0	1	1	1	1
1 1	1	1	1	1

8.

	A	B	C	D	
p q r	q + r	p + A	p + q	C + r	B ≡ D
0 0 0	0	0	0	0	1
0 0 1	1	1	0	1	1
0 1 0	1	1	1	1	1
0 1 1	1	1	1	1	1
1 0 0	0	1	1	1	1
1 0 1	1	1	1	1	1
1 1 0	1	1	1	1	1
1 1 1	1	1	1	1	1

11.

	A	B	C	
p q	p	0	A or B	C ≡ A
0 0	0	0	0	1
0 1	0	0	0	1
1 0	1	0	1	1
1 1	1	0	1	1

12.

	A	B	C	
p q	p	0	p AND 0	B ≡ C
0 0	0	0	0	1
0 1	0	0	0	1
1 0	1	0	0	1
1 1	1	0	0	1

13.

	A	B	
p q	'p	'A	B ≡ p
0 0	1	0	1
0 1	1	0	1
1 0	0	1	1
1 1	0	1	1

14.

			A	B	C	D	E	
p	q	r	qr	p+A	p+q	p+r	C and D	B ≡ E
0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	1	0	1
0	1	0	0	0	1	0	0	1
0	1	1	1	1	1	1	1	1
1	0	0	0	1	1	1	1	1
1	0	1	0	1	1	1	1	1
1	1	0	0	1	1	1	1	1
1	1	1	1	1	1	1	1	1

18.

		A	B	
p	q	pq	p + A	B ≡ p
0	0	0	0	1
0	1	0	0	1
1	0	0	1	1
1	1	1	1	1

15.

		A	
p	q	1	A or p
0	0	1	1
0	1	1	1
1	0	1	1
1	1	1	1

19.

		A	B	C	
p	q	\`p+q	p(A)	pq	B ≡ C
0	0	1	0	0	1
0	1	1	0	0	1
1	0	0	0	0	1
1	1	1	1	1	1

20.

p	q	\`(pq)	\`p + \`q	A ≡ B
0	0	1	1	1
0	1	1	1	1
1	0	1	1	1
1	1	0	0	1

16.

		A	
p	q	pp	A ≡ B
0	0	0	1
0	1	0	1
1	0	1	1
1	1	1	1

21.

		A	B	C	D	
p	q	p → q	q → p	A and B	p ≡ q	C ≡ D
0	0	1	1	1	1	1
0	1	1	0	0	0	1
1	0	0	1	0	0	1
1	1	1	1	1	1	1

17.

	A	B	
p	p	p+p	A ≡ B
0	0	0	1
0	0	0	1
1	1	1	1
1	1	1	1

22.

		A	B	
p	q	p ≡ q	p → q	A → B
0	0	1	1	1
0	1	0	1	1
1	0	0	0	1
1	1	1	1	1

23.

p	q	r	A $p \to q$	B $q \to r$	C A and B	D $p \to r$	$C \to D$
0	0	0	1	1	1	1	1
0	0	1	1	1	1	1	1
0	1	0	1	0	0	1	1
0	1	1	1	1	1	1	1
1	0	0	0	1	0	0	1
1	0	1	0	1	0	1	1
1	1	0	1	0	0	0	1
1	1	1	1	1	1	1	1

24.

p	q	A $p \to q$	B $\sim p$	C $B + q$	$A \equiv C$
0	0	1	1	1	1
0	1	1	1	1	1
1	0	0	0	0	1
1	1	1	0	1	1

b)

$(w+{}'x)(w+y+{}'z)({}'w+{}'x+{}'y)({}'x)$

$\equiv (w{}'x+{}'x{}'x)(w+y+{}'z)({}'w+{}'x+{}'y)$

$\equiv [{}'x+w{}'x({}'w+{}'y)](w+y+{}'z)$ Distributive

$\equiv [{}'x+w{}'x{}'w+w{}'x{}'y](w+y+{}'z)$

$\equiv [{}'x+w{}'w{}'x+w{}'x{}'y](w+y+{}'z)$ ∴ Commutative Law

$\equiv [{}'x+0+w{}'x{}'y][w+y+{}'z]$ $A{}'A=0$

$\equiv [({}'x+w)({}'x+{}'x)({}'x+{}'y)][w+y+{}'z]$ Distributive

$\equiv [(w+{}'x)({}'x)({}'x+{}'y)][w+y+{}'z]$ ${}'A+{}'A={}'A$

$\equiv {}'x({}'x+{}'y)(w+{}'x)(w+y+{}'z)$

$\equiv ({}'x+{}'x{}'y)[w+{}'x(y+{}'z)]$ Distributive Law

$\equiv ({}'x+{}'x{}'y)[w+{}'xy+{}'x{}'z]$