CSE 311: Foundations of Computing I

Section 1: Logic Solutions

0. Exclusive Or

For each of the following, decide whether inclusive-or or exclusive-or is intended:

(a) Experience with C or Java is required.

Solution:

Inclusive Or.

(b) Lunch includes soup or salad.

Solution:

Exclusive Or.

(c) Publish or perish

Solution:

Exclusive Or.

(d) To enter the country you need a passport or voter registration card.

Solution:

Inclusive Or.

1. Translations

For each of the following, define propositional variables and translate the sentences into logical notation.

(a) I will remember to send you the address only if you send me an e-mail message.

Solution:

p : I will remember to send you the address

q : You send me an e-mail message

$$\boxed{p \rightarrow q}$$

(b) If berries are ripe along the trail, hiking is safe if and only if grizzly bears have not been seen in the area.

Solution:

p : Berries are ripe along the trail

q : Hiking is safe

r : Grizzly bears have been seen in the area

$$\boxed{p \rightarrow (q \leftrightarrow \neg r)}$$

(c) Unless I am trying to type something, my cat is either eating or sleeping.

Solution:

$$p : \text{My cat is eating}$$
$$q : \text{My cat is sleeping}$$
$$r : \text{I'm trying to type}$$

$$\boxed{\neg r \rightarrow (p \oplus q)}$$

2. Teatime

Consider the following sentence:

If I am drinking tea then I am eating a cookie, or, if I am eating a cookie then I am drinking tea.

(a) Define propositional variables and translate the sentence into an expression in logical notation.

Solution:

$$p : \text{I am drinking tea}$$
$$q : \text{I am eating a cookie}$$

$$\boxed{(p \rightarrow q) \lor (q \rightarrow p)}$$

(b) Fill out a truth table for your expression.

Solution:

p	q	$(p \rightarrow q)$	$(q \rightarrow p)$	$(p \rightarrow q) \lor (q \rightarrow p)$
T	T	T	T	T
T	F	F	T	T
F	T	T	F	T
F	F	T	T	T

(c) Based on your truth table, classify the original sentence as a contingency, tautology, or contradiction.

Solution:

Tautology

3. Truth Tables

Write a truth table for each of the following:

(a) $(p \oplus q) \vee (p \oplus \neg q)$

Solution:

p	q	$p \oplus q$	$p \oplus \neg q$	$(p \oplus q) \vee (p \oplus \neg q)$
T	T	F	T	T
T	F	T	F	T
F	T	T	F	T
F	F	F	T	T

(b) $(p \vee q) \rightarrow (p \oplus q)$

Solution:

p	q	$p \vee q$	$p \oplus q$	$(p \vee q) \rightarrow (p \oplus q)$
T	T	T	F	F
T	F	T	T	T
F	T	T	T	T
F	F	F	F	T

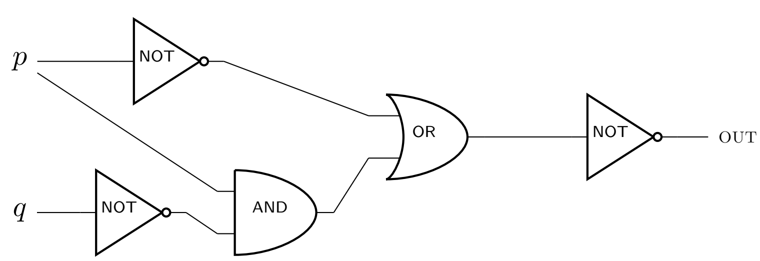
(c) $p \leftrightarrow \neg p$

Solution:

p	$\neg p$	$p \leftrightarrow \neg p$
T	F	F
F	T	F

4. Circuitous

Translate the following circuit into a logical expression.



Solution:

$\neg(\neg p \vee (p \wedge \neg q))$

5. The Curious Case of The Lying TAs

A new UW student wandered around the Paul Allen Center on their first day at UW. They found (as many do) that there is a secret room in its basements. On the door of this secret room is a sign that says:

```
All ye who enter, beware!  Every inhabitant of this room is either a TA who
always lies or a student who always tells the truth!
```

The UW student somehow magically divines that this sign is telling the truth and enters the room. Now, consider the following scenarios:

(a) After entering the room, two inhabitants suddenly walk up to the UW student. One of them one of them says: "*At least one of us is a TA*".

Model this scenario in the following method. Hint: your method should consist of a series of calls to the `assume(...)` method.

Solution:

```
public static void modelPartA(BoolExpr xIsTa, BoolExpr xIsStudent,
                              BoolExpr yIsTa, BoolExpr yIsStudent) {
    // Step 1: assert each inhabitant is a TA xor a student
    assume(xor(xIsTa, xIsStudent));
    assume(xor(yIsTa, yIsStudent));

    // Step 2: encode the claim "At least one of us is a TA"
    BoolExpr claim = or(xIsTa, yIsTa);

    // Step 3: if x is a student, then the claim is true
    assume(implies(xIsStudent, claim));

    // Step 4: if x is a TA, then the claim is false
    assume(implies(xIsTA, not(claim)));
}
```

(b) Now, consider the same scenario as part (a), only this time three inhabitants walk up to the student. Model this new scenario:

Solution:

```
public static void modelPartB(BoolExpr xIsTa, BoolExpr xIsStudent,
                              BoolExpr yIsTa, BoolExpr yIsStudent,
                              BoolExpr zIsTa, BoolExpr zIsStudent) {
    assume(xor(xIsTa, xIsStudent));
    assume(xor(yIsTa, yIsStudent));
    assume(xor(zIsTa, zIsStudent));

    BoolExpr claim = or(or(xIsTa, yIsTa), zIsTA);

    assume(implies(xIsStudent, claim));
    assume(implies(xIsTA, not(claim)));
}
```

(c) What if n inhabitants walk up to the student? Model this situation.

Solution:

```
public static void modelPartC(int n, BoolExpr[] isTa, BoolExpr[] isStudent) {
    for (int i = 0; i < n; i++) {
        assume(xor(isTa[i], isStudent[i]);
    }

    BoolExpr claim = isTa[0];
    for (int i = 1; i < n; i++) {
        claim = or(claim, isTa[i]);
    }

    assume(implies(isStudent[0], claim));
    assume(implies(isTa[0], not(claim)));
}
```

(d) Let's consider a new scenario. Suppose three inhabitants walk up and surround the UW student. One of them says: "*Every TA in this circle has a TA to her immediate right*". Model this situation:

Solution:

```
public static void modelPartD(BoolExpr xIsTa, BoolExpr xIsStudent,
                              BoolExpr yIsTa, BoolExpr yIsStudent,
                              BoolExpr zIsTa, BoolExpr zIsStudent) {
    assume(xor(xIsTa, xIsStudent));
    assume(xor(yIsTa, yIsStudent));
    assume(xor(zIsTa, zIsStudent));

    BoolExpr claim = and(
        implies(xIsTa, yIsTa),
        and(
            implies(yIsTa, zIsTa),
            implies(zIsTA, xIsTa)));

    assume(implies(xIsStudent, claim));
    assume(implies(xIsTA, not(claim)));
}
```