Problem Set 1

Engineering 3422, 2005

Tuesday Sept 20st

Q0. **(a)** Consider the following law

$$\text{Definition of Implication: } P \to Q \Leftrightarrow \neg P \vee Q$$

Make the following variable substitutions: Replace P by $P \vee Q$ and Q by R. I.e. calculate

$$(P \to Q \Leftrightarrow \neg P \vee Q)\,[P, Q := P \vee Q, R]$$

Is the resulting instance a law? Why?

(b) In the expression $\underline{\neg(P \vee Q)} \vee R$ substitute the underlined part of expression according to De Morgan's law. Are the two expressions equivalent.

Q1 Give an algebraic proof of the following equivalences. For each step, remember to use the name of the law applied as a hint. Underline the part of each expression you are focussing on.

(a) $P \vee Q \to R \Leftrightarrow (P \to R) \wedge (Q \to R)$

(b) $\neg(P \vee (\neg P \wedge Q)) \Leftrightarrow (\neg P \wedge \neg Q)$

Q2 Show the following statements are tautologies. Use algebraic proof. For each step, remember to use the name of the law applied as a hint. Underline the part of each expression you are focussing on.

(a) $P \wedge Q \leftrightarrow P \vee Q \leftrightarrow P \leftrightarrow Q$ (Note that $\leftrightarrow$ is both commutative and associative)

(b) $P \to P$

(c) $\neg P \to (P \to Q)$

(d) $(P \to Q) \to (P \vee R \to Q \vee R)$

Q3 **(a)** I had the following statement in a C++ program[1]

[1] "assert" is a predefined subroutine (well, technically, it's a macro) defined in <assert.h> which halts the program with an error message if its argument evaluates to false. "assert" calls are very useful for helping you to develop reliable software.

```
assert( (!is_member_call || recipient_is_this)
        && (recipient==NULL || recipient_is_this)
        && (!recipient_is_this || recipient!=NULL || is_member_call) ) ;
```

I thought this was pretty close to unreadable, so I decided to simplify it as much as possible.

Let's assign propositional variables to the primitive statements as follows

P	:	is_member_call
Q	:	recipient_is_this
R	:	recipient != NULL

We get $(\neg P \vee Q) \wedge (\neg R \vee Q) \wedge (\neg Q \vee R \vee P)$.

Algebraically simplify this statement as much as you can. For each step, remember to use the name of the law applied as a hint. Then translate the result back into C++ notation. You may find the law you proved in Q1 (a) helpful. Hint: it should be possible to simplify to an expression using only two propositional operations.

(b) As you know, $\wedge$ and $\vee$ are commutative. Does it follow that in C++ the expression

```
0 <= i && i <N && A[i]==0
```

can always be replaced by the expression

```
A[i]==0 && 0 <= i && i <N      ?
```

Q4. Problems 13 and 14 of section 2.4.3.

Q5 A logician has two caskets, one gold and one silver. Into one she placed many coins and into the other she placed pile of rocks. On the gold casket she wrote the inscription: the gold is not here. On the silver casket she wrote: exactly one of these inscriptions is true. She honestly tells you that each inscription is either true or false. Which casket should you choose? Explain clearly.