Workshop Specification and Testing, Week 3

The topic of today is propositional logic. You have seen this in Chapter 2 of “The Haskell Road”.

The language of propositional logic is given by the following grammar:

$$\varphi ::= p \mid (\neg\varphi) \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid (\varphi \to \varphi) \mid (\varphi \leftrightarrow \varphi)$$

Here p ranges over a set of proposition letters P.

Draw parse trees for the following formulas (convention: we leave out the outermost parentheses):

1. $\neg(\neg(\neg p))$
2. $\neg(p \vee (\neg q))$
3. $\neg((\neg p) \wedge (\neg q))$
4. $(p \to q) \leftrightarrow ((\neg q) \to (\neg p))$

List the subformulas for all formulas in the first exercise.

Give a definition of the set of *subformulas* of a formula φ.

Give an *algorithm* for computing the number of subformulas of a formula φ.

Give truth tables for the formulas in the first exercise.

A **literal** is an atom p or the negation of an atom $\neg p$.

A **clause** is a disjunction of literals.

A formula C is in **conjunctive normal form (or: CNF)** if it is a conjunction of **clauses**.

Here is a grammar for literals, clauses, and formulas in CNF.

$$\begin{aligned} L &::= & p \mid \neg p \\ D &::= & L \mid L \vee D \\ C &::= & D \mid D \wedge C \end{aligned}$$

Give CNF equivalents of the formulas in the first exercise.

Can you define **disjunctive normal form (or: DNF)** and give a grammar for it?

Which of the following formulas are tautologies. Which are contradictions? Which are satisfiable?

1. $p \vee (\neg q)$.
2. $p \wedge (\neg p)$.
3. $p \vee (\neg p)$.
4. $p \to (p \vee q)$.
5. $(p \vee q) \to p$.

Simplify the following Ruby statement by simplifying the condition:

```
if not (guess != secret1 && guess != secret2)
  print "You win."
else
  print "You lose."
end
```

Simplify the following Ruby statement:

```
if guess != secret1
```

```
  if guess != secret2 print "You lose." else print "You win."
else
  print "You win."
end
```

Consider the following Ruby statement:

```
if guess1 == secret1
  print "one"
elsif guess2 == secret2
  print "two"
elsif guess3 == secret3
  print "three"
else
  print "four"
end
```

Suppose the statement is executed and *"three"* gets printed. What does this tell you about the state?

Suppose the behaviour of φ is specified by means of a truth table. Here is an example:

p	q	r	φ
T	T	T	F
F	T	T	T
T	F	T	T
F	F	T	F
T	T	F	F
F	T	F	T
T	F	F	T
F	f	F	T

To give an equivalent for φ in DNF, all you have to do is list the disjunction of the rows in the truth table where the formula turns out true. Give an equivalent of φ in DNF.

Consider the truth table of the previous example again. It is also easy to give an equivalent for φ in CNF on the basis of the truth table. How? (Hint: the negation of a row where the truth table gives false can be expressed as a disjunction. Take the conjunction of all these disjunctions.) Give an equivalent of φ in CNF.
