Workshop Specification and Testing, Week 3: Answers

Question 1 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. $(p \to q) \leftrightarrow ((\neg q) \to (\neg p))$

Answer: $\neg\neg\neg p$

$$\begin{tikzcd}
\neg \arrow[d, dash] \\
\neg \arrow[d, dash] \\
\neg \arrow[d, dash] \\
p
\end{tikzcd}$$

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

$$\begin{tikzcd}
 & \neg \arrow[d, dash] & \\
 & \vee & \\
p & & \neg \arrow[d, dash] \\
 & & q
\end{tikzcd}$$

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

$$\begin{tikzcd}
 & \neg \arrow[d, dash] & \\
 & \wedge & \\
\neg \arrow[d, dash] & & \neg \arrow[d, dash] \\
p & & q
\end{tikzcd}$$

5.20.4: $((p \to q) \leftrightarrow (\neg q \to \neg p))$

$$\begin{tikzcd}
 & & \leftrightarrow \arrow[dl, dash] \arrow[dr, dash] & & \\
 & \to & & \to & \\
p & q & & \neg \arrow[d, dash] & \neg \arrow[d, dash] \\
 & & & q & p
\end{tikzcd}$$

Question 2 List the subformulas for all formulas in the first exercise.

Answer: $\neg\neg\neg p$:

$$\{\neg\neg\neg p, \neg\neg p, \neg p, p\}$$

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

$$\{\neg(p \vee \neg q), p \vee \neg q, p, \neg q, q\}$$

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

$$\{\neg(\neg p \wedge \neg q), \neg p \wedge \neg q, \neg p, p, \neg q, q\}$$

$((p \to q) \leftrightarrow (\neg q \to \neg p))$

$$\{((p \to q) \leftrightarrow (\neg q \to \neg p)), (p \to q), (\neg q \to \neg p), \neg p, \neg q, p, q\}$$

Question 3 Give a definition of the set of subformulas of a formula φ.

Answer: Definition by recursion on the structure of the formula.

$$\begin{array}{rcl}
\mathrm{SUB}(p) & := & \{p\} \\
\mathrm{SUB}(\neg\varphi) & := & \{\neg\varphi\} \cup \mathrm{SUB}(\varphi) \\
\mathrm{SUB}(\varphi_1 \wedge \varphi_2) & := & \{\varphi_1 \wedge \varphi_2\} \cup \mathrm{SUB}(\varphi_1) \cup \mathrm{SUB}(\varphi_2) \\
\mathrm{SUB}(\varphi_1 \vee \varphi_2) & := & \{\varphi_1 \vee \varphi_2\} \cup \mathrm{SUB}(\varphi_1) \cup \mathrm{SUB}(\varphi_2) \\
\mathrm{SUB}(\varphi_1 \to \varphi_2) & := & \{\varphi_1 \to \varphi_2\} \cup \mathrm{SUB}(\varphi_1) \cup \mathrm{SUB}(\varphi_2) \\
\mathrm{SUB}(\varphi_1 \leftrightarrow \varphi_2) & := & \{\varphi_1 \leftrightarrow \varphi_2\} \cup \mathrm{SUB}(\varphi_1) \cup \mathrm{SUB}(\varphi_2)
\end{array}$$

Question 4 Give an algorithm for computing the number of subformulas of a formula φ.

Answer: A recursive definition in the spirit of the definition of SUB computes an upper bound to the number of subformulas, for the same subformula may occur more than once. E.g., the subformula p occurs twice in $p \vee \neg p$.

A better approach is therefore to first compute SUB(φ), and then count its elements.

Question 5 Give truth tables for the formulas in the first exercise.

Answer: $\neg\neg\neg p$:

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

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

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

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

p	q	$\neg p$	$\neg q$	$\neg p \wedge \neg q$	$\neg(\neg p \wedge \neg q)$
T	T	F	F	F	T
F	T	T	F	F	T
T	F	F	T	F	T
F	F	T	T	T	F

$((p \rightarrow q) \leftrightarrow (\neg q \rightarrow \neg p))$

p	q	$\neg p$	$\neg q$	$p \rightarrow q$	$\neg q \rightarrow \neg p$	$(p \rightarrow q) \leftrightarrow (\neg q \rightarrow \neg p)$
T	T	F	F	T	T	T
F	T	T	F	T	T	T
T	F	F	T	F	F	T
F	F	T	T	T	T	T

Question 6 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.

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

Give CNF equivalents of the formulas in the first exercise.

Answer: Instead of following an algorithm, we might just as well look at the truth tables, and construct a CNF formula with the same truth table.

$\neg\neg\neg p$. CNF equivalent: $\neg p$

$\neg(p \vee \neg q)$. CNF equivalent: $\neg p \wedge q$.

$\neg(\neg p \wedge \neg q)$. CNF equivalent: $p \vee q$.

$((p \rightarrow q) \leftrightarrow (\neg q \rightarrow \neg p))$. CNF equivalent: $p \vee \neg p$.

Question 7 Can you define **disjunctive normal form (or: DNF)** and give a grammar for it?

Answer: A formula in disjunctive normal form is a disjunction of conjunctions of literals. Here is a grammar for DNF:

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

Question 8 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. $p \rightarrow (p \vee q)$.
5. $(p \vee q) \rightarrow p$.

Answer:

1. $p \vee \neg q$: not a tautology, for $p = F, q = T$ falsifies it. Satisfiable, for $p = T$ satisfies it. Hence not a contradition.
2. $p \wedge \neg p$: a contradiction, for neither $p = T$ nor $p = F$ satisfies it. Hence not a tautology and not satisfiable.
3. $p \vee \neg p$: a tautology, for both $p = T$ and $p = F$ satisfy it. Hence satisfiable and not a contradiction.
4. $p \rightarrow (p \vee q)$: a tautology, for all valuations for p, q satisfy it. Hence satisfiable and not a contradiction.
5. $(p \vee q) \rightarrow p$: Not a tautology, for $p = F, q = T$ falsifies it. Not a contradiction, for $p = T, q = T$ satisfies it. Hence satisfiable.

Question 9 Simplify the following Ruby statement by simplifying the condition:

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

Answer: Use the equivalence between $\neg(\neg p \wedge \neg q)$ and $p \vee q$. This gives:

```
if guess == secret1 || guess == secret2
  print "You win."
else
  print "You lose."
end
```

Question 10 Simplify the following Ruby statement:

```
if guess != secret1
  if guess != secret2 print "You lose." else print "You win."
else
  print "You win."
end
```

Answer:

```
if guess == secret1 || guess == secret2
  print "You win."
else
  print "You lose."
end
```

Question 11 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?

Answer: The output makes clear that `guess1 != secret1`, `guess2 != secret2`, and `guess3 == secret3`.

Question 12 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.

Answer:

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

Question 13 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.

Answer:

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