ECE 208, Fall 2019
Assignment No. 1
Due 6:00 p.m., Thursday, September 26
15 points in total

Submission: Submissions are to be in pdf format. No late submissions accepted. Submit to the appropriate dropbox in Learn.

1. (1 point) Exercise 2.4, from page 45 of the textbook:
   Prove
   $$((A \oplus B) \oplus B) \equiv A \text{ and } ((A \iff B) \iff B) \equiv A \ .$$

   Solution:
   $$((A \oplus B) \oplus B) \equiv (A \oplus (B \oplus B)) \equiv (A \oplus false) \equiv A \ ,$$
   or, by truth table:

A	B	$A \oplus B$	$(A \oplus B) \oplus B$
F	F	F	F
F	T	T	F
T	F	T	T
T	T	F	T

   $$((A \iff B) \iff B) \equiv (A \iff (B \iff B)) \equiv (A \iff true) \equiv A \ .$$
   or, by truth table,

A	B	$(A \iff B)$	$(A \iff B) \iff B$
F	F	T	F
F	T	F	F
T	F	F	T
T	T	T	T

2. (1 point) Question 2.9, from page 46 of the textbook.

Solution:

A	B	C	$(A \wedge B)$	$(A \wedge B) \Rightarrow C$	$A \Rightarrow C$	$B \Rightarrow C$	$(A \Rightarrow C) \vee (B \Rightarrow C)$
F	F	F	F	T	T	T	T
F	F	T	F	T	T	T	T
F	T	F	F	T	T	F	T
F	T	T	F	T	T	T	T
T	F	F	F	T	F	T	T
T	F	T	F	T	T	T	T
T	T	F	T	F	F	F	F
T	T	T	T	T	T	T	T

3. (5 points) We're skipping section 2.4 of the textbook, but that's because students should already know that material. So fill in the details in the proof of

Theorem 2.37 Let $\circ$ be a binary operator from which negation and all other operators can be defined (without the use of any other operators). Then $\circ$ is either nand or nor.

Proof: Ben-Ari gives a proof outline:

Suppose that all of the operators can be defined in terms of $\circ$. Then negation must be defined by some equivalence of the form

$$\neg A \equiv A \circ A \circ \ \ldots \ \circ A \ , \tag{1}$$

where parentheses can be added to the right-hand side as necessary.

Moreover, any binary operator op must be defined by an equivalence

$$A_1 \, op \, A_2 \equiv B_1 \circ B_2 \circ \ \ldots \ \circ B_n \ . \tag{2}$$

where each B_i is either A_1 or A_2, and parentheses may be added to the right-hand side as needed.

Let $\mathscr{I}$ be any interpretation such that $v_{\mathscr{I}}(A) = T$. Then it follows from the above that we must have

$$F = v_{\mathscr{I}}(\neg A) = v_{\mathscr{I}}(A \circ A \circ \ \ldots \circ A) \ .$$

Show that $v_{\mathscr{I}}(A_1 \circ A_2) = F$ whenever $v_{\mathscr{I}}(A_1) = v_{\mathscr{I}}(A_2) = T$. Similarly, **show that** $v_{\mathscr{I}}(A_1 \circ A_2) = T$ whenever $v_{\mathscr{I}}(A_1) = v_{\mathscr{I}}(A_2) = F$. [Hint: I would use proof by contradiction (as well as induction).]

Suppose to the contrary that $v_{\mathscr{I}}(A_1 \circ A_2) = T$ whenever $v_{\mathscr{I}}(A_1) = v_{\mathscr{I}}(A_2) = T$. Then consider the case where A is an atomic proposition; if $v_{\mathscr{I}}(A) = T$, then by structural induction, the truth value of any subformula of $A \circ A \circ \ \ldots \circ A$ is T, regardless of the

number of occurrences of $\circ$, or any parentheses. But then no equivalence of the form of (1) holds.

Similarly, if $v_{\mathscr{I}}(A) = F$, then we must have

$$T = v_{\mathscr{I}}(\neg A) = v_{\mathscr{I}}(A \circ A \circ \ \ldots \ \circ A) \ .$$

But if $v_{\mathscr{I}}(A_1 \circ A_2) = F$ whenever $v_{\mathscr{I}}(A_1) = v_{\mathscr{I}}(A_2) = F$, then consider the case where A is an atomic proposition: whenever $v_{\mathscr{I}}(A) = F$, by structural induction, the truth value of any subformula of $A \circ A \circ \ \ldots \circ A$ is F, so, again, no equivalence of the form of (1) holds.

Thus the truth table for $\circ$ must therefore have the following form:

A_1	A_2	$A_1 \circ A_2$
F	F	T
F	T	T or F
T	F	T or F
T	T	F

If $\circ$ gives the same truth value T for both middle rows, then $\circ$ is nand; if it gives F for both rows then $\circ$ is nor.

The only remaining possibility is that $\circ$ is defined to give different truth values for the middle two rows. **Prove** by induction that, in that case, only 'projection' and 'negated projection' are definable, in the sense that

$$B_1 \circ B_2 \circ \ \ldots \ \circ B_n \equiv \neg \ \ldots \neg B_i \tag{3}$$

for some i and zero or more negations. [Note that, without loss of generality, the B_i in the above equation can be assumed to be atomic propositions.]

First consider the two possibilities for the truth table. If $\circ$ gives F for the second row of the truth table, and T for the third, then

$$A_1 \circ A_2 \equiv \neg A_2 \ .$$

If $\circ$ gives T for the second row, and F for the third, then

$$A_1 \circ A_2 \equiv \neg A_1 \ .$$

Without loss of generality, the B_i in (3) can be assumed to be atomic propositions. For arbitrary $n \in \mathbb{N}$ and atomic propositions B_i, $0 < i \leq n$, we shall prove by structural induction that an equivalence of the form of equation (3) holds.

The base case is trivial: each atomic proposition is of the form of the right-hand side of (3), with zero negations.

For the induction step, suppose that the left- and right-hand subtrees of a subformula of the left-hand side of (3) are each equivalent to a formula of the form of the right-hand side. Then, by the above truth-table argument, that subformula must be equivalent to negation of a formula of the form of the right-hand side. But then it too is of the form of the right-hand side. This completes the induction.

□

4. (6 points) Here's a little exercise in program verification: show that the procedure given in the notes for translating a formula's reverse Polish notation into the tree representation is correct.

   First, let's make the stack-based algorithm more precise:

   Algorithm: Parse RPN.

   Input: Reverse Polish notation for a well-formed formula A.

   Output: A (in tree form) at the top of the stack.

```
Initial stack contents:  arbitrary.
Start at the left-hand end of the input string.
While not past the right-hand end of the input string,
     If the current symbol is an atomic proposition symbol p,
          push the tree consisting of a leaf labelled by p.
     If the current symbol is the negation symbol,
          pop the tree representation of formula F;
          push the tree with a root labelled by the negation symbol,
               and with the subtree F rooted at its only child.
     If the current symbol is a binary operator,
          pop the tree representation of formula F1;
          pop the tree representation of formula F2;
          push the tree with a root labelled by the binary operator,
               and subtree F1 rooted at its left-hand child,
               and subtree F2 rooted at its right-hand child.
     Advance to the next symbol in the input string
end while
```

   Prove by structural induction that the procedure does indeed leave the formula A (in its tree form) at the top of the stack. (Note that this will imply that RPN is unambiguous, in spite of its lack of parentheses.) For the induction step to work, the inductive hypothesis will also have to assert that when the algorithm terminates the initial stack contents lie immediately below the top element of the stack.

Hint: Let the RPN representation of a formula A be denoted $RPN(A)$. Show that:

- if A is a leaf labelled by an atomic proposition p, then $RPN(A)$ is p;
- if A is the negation of a formula F, then $RPN(A)$ is $RPN(F)\neg$;
- if A is tree representation of $F_1 \, op \, F_2$ (in conventional, infix notation), then $RPN(A)$ is $RPN(F_2)RPN(F_1)op$ – that is, the concatenation of the strings $RPN(F_2)$, $RPN(F_1)$ and op.

Solution: To prove the hint, for any formula A, let $Preorder(A)$ be the Polish notation for A. The RPN representation is simply the reverse, so $RPN(A)$ is:

- p, if $Preorder(A)$ is of the form of an atomic proposition symbol p;
- $RPN(F)\neg$, if $Preorder(A)$ is of the form $\neg Preorder(F)$; and
- $RPN(F_2)RPN(F_1)op$, if $Preorder(A)$ is of the form $op\,Preorder(F_1)Preorder(F_2)$ (in infix notation).

The proof for the hint does not require an induction – only a case analysis.

We now show by induction on the structure of the formula A that when the algorithm terminates (as it must, because it continues to advance along the finite input string until it reaches the end), A – in tree form – is at the top of the stack, with the original stack contents below it.

Base case: If A is a leaf node labelled with an atomic proposition, then all that the algorithm does is to push A onto the stack. The initial contents of the stack therefore lie immediately below.

Induction step:

- Suppose that A is the negation of a formula F, and that the inductive hypothesis holds for F. By the hint, the input $RPN(A)$ is the concatenation $RPN(F)\neg$. Therefore, by inductive assumption, as soon as the algorithm has passed the end of $RPN(F)$, F is on the stack, with the original stack contents below it; in its final step, the algorithm pops F from the stack, forms the tree representing its negation, A, and pushes it onto the stack, leaving the original stack contents immediately below it.
- Suppose that A is of the form $F_1 \, op \, F_2$ (in infix notation), and that the inductive hypothesis holds for F_1 and F_2. By the hint, the input $RPN(A)$ is the concatenation of $RPN(F_2)$, $RPN(F_1)$, and op. Therefore, by inductive assumption, when the algorithm passes the end of $RPN(F_2)$, F_2 is at the top of the stack, with the initial stack contents immediately below it.

  Now, as the algorithm begins to process the input substring $RPN(F_1)$, it is exactly as if the algorithm were being started on the input $RPN(F_1)$, with the stack initially containing F_2 at the top, and the prior contents immediately below. By inductive assumption then, when the algorithm passes the end of the input

substring $RPN(F_1)$, F_1 lies at the top of the stack, with F_2 immediately below it, and the initial stack contents immediately below that. The algorithm then pops F_1 and F_2 in sequence, and pushes the tree A. The latter is therefore left at the top of the stack, with the initial stack contents immediately below it.

This completes the induction.

5. (1 point) Prove Theorems 2.44 to 2.47, inclusively, of the textbook.

Theorem 2.44: If U is satisfiable, then so is $U \setminus \{A_i\}$, for all i.

Proof: If U is satisfiable, then there exists an interpretation $\mathscr{I}$ for U that simultaneously satisfies all formulas in U. For any i, such an interpretation also satisfies $U \setminus \{A_i\}$.

Theorem 2.45: If U is satisfiable and B is valid, then $U \cup \{B\}$ is satisfiable.

Proof: If U is satisfiable, then there exists an interpretation $\mathscr{I}$ for U that satisfies all formulas in U. Extend it (if necessary) to an interpretation for $U \cup \{B\}$, by assigning truth values to the atoms of B. Because B is valid, it is satisfied by all interpretations, including the extended version of $\mathscr{I}$. Therefore $U \cup \{B\}$ is satisfiable.

Theorem 2.46: If U is unsatisfiable, then for any formula B, $U \cup \{B\}$ is unsatisfiable.

Proof: (by contrapositive). If $U \cup \{B\}$ is satisfiable, there would be an interpretation that satisfied all of its formulas; in particular, such an interpretation would satisfy all formulas in U.

Theorem 2.47: If U is unsatisfiable and, for some i, A_i is valid, then $U \setminus \{A_i\}$ is unsatisfiable.

Proof: Suppose that U is unsatisfiable. Let $\mathscr{I}$ be an arbitrary interpretation for U. Then $\mathscr{I}$ assigns F to some formula in U. But if $A_i \in U$ is valid, then $\mathscr{I}$ assigns T to A_i, so it must assign F to some formula other than A_i in U. Because $\mathscr{I}$ is an arbitrary interpretation for U, every such interpretation must assign F to some formula in U. Hence, U is unsatisfiable.

6. (1 point) Prove Theorems 2.53 and 2.54 of the textbook.

Theorem 2.53: If $U \vDash A$, then $U \cup \{B\} \vDash A$ for any formula B.

Proof: If $U \vDash A$, then every interpretation that satisfies all formulas in U also satisfies A. Now, any interpretation for $U \cup \{B\}$ that satisfies all formulas in that set must necessarily satisfy all formulas in U; hence, it also satisfies A.

Theorem 2.54: If $U \vDash A$ and B is valid, then $U \setminus \{B\} \vDash A$.

Proof: (by contrapositive). Suppose that A is not a logical consequence of $U \setminus \{B\}$. Then there exists an interpretation $\mathscr{I}$ for $U \setminus \{B\} \cup \{A\}$ that satisfies all formulas in $U \setminus \{B\}$ but does not satisfy A. Extend $\mathscr{I}$ (if necessary) to an interpretation for $U \cup \{A\}$ (in other words, assign truth values to atoms of B that do not appear in $U \cup \{A\}$). Because B is valid, the result is a model of U; but it does not satisfy A. Hence, A is not a logical consequence of U.