# CS51 - Assignment 5
# Proofs

Due: October 9th, at 11:59pm




https://xkcd.com/1724/

For this assignment, we will:

- gain more practice with Propositional Logic,
- learn about functional completeness,
- write more proofs using loop invariants, and
- gain more practice writing Math and code in LaTeX.

You are encouraged to work on pen and paper or a board first before you proceed with typing your response in the provided `assignment5.tex`. As always, please keep track of the time you have spent on the assignment and submit your feedback.

## 1 Propositional Logic

Figure 1 shows the truth tables for 16 different logical connectives. We have seen many of these already in class:$\neg, \wedge, \vee, \Rightarrow, \iff, \oplus$ . The (slightly) new ones are the Sheffer stroke ($|$), which corresponds to the logic gate NAND (the negation of AND), and Peirce's arrow ($\downarrow$), which corresponds to the logic gate NOR (the negation of OR).

| $p$ | $q$ | 1<br>True | 2<br>$p \vee q$ | 3<br>$p \Leftarrow q$ | 4<br>$p$ | 5<br>$p \Rightarrow q$ | 6<br>$q$ | 7<br>$p \Leftrightarrow q$ | 8<br>$p \wedge q$ | 9<br>$p \mid q$ | 10<br>$p \oplus q$ | 11<br>$\neg q$ | 12 | 13<br>$\neg p$ | 14 | 15<br>$p \downarrow q$ | 16<br>False |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $T$ | $T$ | $T$ | $T$ | $T$ | $T$ | $T$ | $T$ | $T$ | $T$ | $F$ | $F$ | $F$ | $F$ | $F$ | $F$ | $F$ | $F$ |
| $T$ | $F$ | $T$ | $T$ | $T$ | $T$ | $F$ | $F$ | $F$ | $F$ | $T$ | $T$ | $T$ | $T$ | $F$ | $F$ | $F$ | $F$ |
| $F$ | $T$ | $T$ | $T$ | $F$ | $F$ | $T$ | $T$ | $F$ | $F$ | $T$ | $T$ | $F$ | $F$ | $T$ | $T$ | $F$ | $F$ |
| $F$ | $F$ | $T$ | $F$ | $T$ | $F$ | $T$ | $F$ | $T$ | $F$ | $T$ | $F$ | $T$ | $F$ | $T$ | $F$ | $T$ | $F$ |

Figure 1: The truth tables for 16 different logical connectives.

In the next five problems, you will prove that certain sets of logical connectives are *functionally complete* (also known as universal or adequately expressive). A set of logical operators is functionally complete if every logical connective can be expressed using some combination of the connectives in the set. Please note that each problem depends on the result of the previous one.

### 1.1 Problem 1

Prove that the set $\{\vee, \wedge, \Rightarrow, \neg\}$ is functionally complete. To do so, take each logical connective from Figure 1 and try to find a logically equivalent (results in the same truth/false values for the same input) proposition that only uses $\{\vee, \wedge, \Rightarrow, \neg\}$.

*Hint*: You might notice an interesting pattern between the columns 1–8 and 9–16 that you can take advantage of.

### 1.2 Problem 2

Prove that the set $\{\vee, \wedge, \neg\}$ is functionally complete.

*Hint:* You already have proven this partially through Problem 1. Essentially, all you have to do is show that you can express $\Rightarrow$ using $\{\vee, \wedge, \neg\}$ only. Then, translate any logical equivalent proposition that you created in Problem 1 and that contains $\Rightarrow$ to now only contain $\{\vee, \wedge, \neg\}$.

### 1.3 Problem 3

Prove that the sets $\{\vee, \neg\}$ and $\{\wedge, \neg\}$ are both functionally complete.

*Hint:* Look into Problem 2.

### 1.4 Problem 4

Prove that the set $\{|\}$ (the set that contains only the Sheffer stroke or NAND gate, as we saw in the checkpoint) is functionally universal.

*Hint:* Look into Problem 3. You only need to show that you can express $\wedge$ and $\neg$ using the Sheffer stroke.

### 1.5 Problem 5

Prove that the set $\{\downarrow\}$ (the set that contains only Peirce's arrow/NOR) is functionally universal.

*Hint:* Look into Problem 4. You only need to show that you can express $\vee$ and $\neg$ using the Peirce arrow.

## 2 Loop invariants

### 2.1 Problem 6

You are given the following loop:

```
while m>=0 and m<=100:
    m+=1
    n-=1
```

and a potential loop invariant: `m+n=100`.

Show that indeed it is a loop invariant, that is, if the condition is true before entering the loop, then it is also true after exiting the loop.

### 2.2 Problem 7

Let's assume that someone has magically handed you a function that takes a non-negative (decimal) number and returns the number of trailing zeros in its binary form, or in other words, how many times the number can be divided by 2.

```
def count_trailing_zeros_in_binary(n):
    count = 0
    while n % 2 == 0 and n > 0:
        n //= 2
        count += 1
    return count
```

Here are a few examples of calling this function:

```
>>> count_trailing_zeros_in_binary(23)
0
```

```
>>> count_trailing_zeros_in_binary(24)
3
>>> count_trailing_zeros_in_binary(32)
5
```

1. State the pre- and post-conditions for this program
2. State a loop invariant
3. Write a loop invariant proof by filling out all the steps for *Initialization*, *Maintenance*, *Post-condition*, and *Termination*.

## 3 Feedback

Create a file named `feedback5.txt` that answers the questions:

- How long did you spend on this assignment?
- Any comments or feedback? What things did you find interesting, challenging, or boring?

## 4 When you're done

Submit your `assignment5.tex`, resulting `assignment5.pdf`, and `feedback5.txt` files online using Gradescope; be sure you're uploading them to the right assignment. Note that you can resubmit the same assignment as many times as you would like up until the deadline.

## 5 Grading

| Problem | points |
|---|---|
| Problem 1 | 1 |
| Problem 2 | 1 |
| Problem 3 | 2 |
| Problem 4 | 1 |
| Problem 5 | 1 |
| Problem 6 | 1 |
| Problem 7 | 3 |
| Total | 10 |