CSE 311: Foundations of Computing I Winter 2025

Problem Set 3

Due: Wednesday, January 29th by 11:00pm

Instructions

Solutions submission. You must submit your solution via Gradescope. In particular:

- Submit a *single* PDF file containing your solutions to tasks 2, 4, 6 (and optionally 7). Follow the prompt on Gradescope to link tasks to your pages.
- The instructions for submitting tasks 1, 3, and 5 appear below those individual problems.

Task 1 – Make the First Prove [20 pts]

For each of the following, complete a **formal proof** that the claim holds.

a) Given $S \wedge R$, U, and $(R \wedge U) \rightarrow (Q \wedge P)$, it follows that $S \wedge P$ holds.

Your proof is only allowed to use the rules Modus Ponens, Intro $\wedge$, Elim $\wedge$.

b) Given $P \wedge Q$, $\neg Q \vee R$, and $(R \vee \neg Q) \rightarrow S$. it follows that $P \wedge S$ holds.

Your proof is only allowed to use Modus Ponens, Intro $\wedge$, Elim $\wedge$, Intro $\vee$, and **Equivalent**. (*Hint*: One of the known equivalences will be useful!)

c) Given Q, $(P \vee Q) \rightarrow (S \vee R)$, $R \rightarrow (U \wedge V)$, and $S \rightarrow (U \wedge V)$. it follows that $V \wedge U$ holds.

Your proof is only allowed to use the rules Modus Ponens, Intro $\wedge$, Elim $\wedge$, Intro $\vee$, and **Cases**.

d) Given $P \vee Q$, $\neg P \wedge R$, and $Q \rightarrow S$. it follows that $R \wedge S$ holds.

Your proof is only allowed to use Modus Ponens, Intro $\wedge$, Elim $\wedge$, Intro $\vee$, Cases, and Equivalent.

e) Given $(P \rightarrow R) \wedge (Q \vee P)$, $\neg Q \wedge \neg R$, and $R \rightarrow (S \vee T)$. it follows that $R \rightarrow (S \wedge T)$ holds.

Your proof is only allowed to use Modus Ponens, Intro $\wedge$, Elim $\wedge$, Intro $\vee$, Elim $\vee$, Cases, **Principium Contradictionis**, and **Ex Falso**. (*Hint*: Focus on using those last two!)

Submit and check your **formal proofs** here:

http://cozy.cs.washington.edu

You can make as many attempts as needed to find a correct answer.

Task 2 – Cat Goes "Meow", Dog Goes "Proof" [18 pts]

For each of the following, complete a **formal proof** that the claim holds.

a) Given $A \wedge (B \wedge D)$ and $B \rightarrow C$, it follows that $(C \vee D) \wedge (C \vee E)$ holds.

Your proof is only allowed to use the rules Modus Ponens, Elim $\wedge$, Intro $\wedge$, Elim $\vee$, and Intro $\vee$.

b) Given $\neg(B \wedge D) \rightarrow C$, $B \rightarrow (\neg D \vee A)$, and $\neg C$. it follows that $A \vee E$ holds.

Your proof is only allowed to use Modus Ponens, Elim $\wedge$, Intro $\wedge$, Elim $\vee$, Intro $\vee$, and **Equivalent**. (*Hint*: Multiple known equivalences will be useful!)

c) Given $A \rightarrow (B \vee D)$, $\neg(B \vee D) \rightarrow \neg C$, and $(A \vee C) \wedge \neg D$. it follows that B holds.

Your proof is only allowed to use the rules Modus Ponens, Elim $\wedge$, Intro $\wedge$, Elim $\vee$, Intro $\vee$, Equivalent, and **Cases**.

Task 3 – Provin' Right Along [20 pts]

For each of the following, complete a **formal proof** that the claim holds.

a) Given $Q \wedge \neg R$, $P \rightarrow (R \vee S)$, and $(Q \wedge S) \rightarrow T$, it follows that $P \rightarrow T$.

Your proof is only allowed to use the rules the Modus Ponens, **Direct Proof**, Intro $\wedge$, Elim $\wedge$, Intro $\vee$, and Elim $\vee$. (*Hint*: Direct Proof will be needed!)

b) Given $Q \wedge (S \rightarrow R)$, $Q \leftrightarrow S$, and $(P \wedge R) \rightarrow T$, it follows that $P \rightarrow T$.

Your proof is only allowed to use the rules the Modus Ponens, Direct Proof, Intro $\wedge$, Elim $\wedge$, Intro $\vee$, Elim $\vee$, and Equivalent. Note that there are **no rules** for "$\leftrightarrow$"! To use the second fact, you will need to rewrite it as an equivalent statement with only "$\rightarrow$"s.

c) Given $R \wedge S$, $\neg S \vee U$, and $(R \wedge U) \rightarrow V$, it follows that V holds.

Your proof is only allowed to use the rules the Modus Ponens, Direct Proof, Intro $\wedge$, Elim $\wedge$, Intro $\vee$, Elim $\vee$, and the Latin Rules. Note, in particular, that Equivalent is **not** allowed.

Note that $\neg S \vee U$ is equivalent to $S \rightarrow U$. Since we know S (from $R \wedge S$), we could then get that U holds by Modus Ponens. However, in this problem, we don't have Equivalent, so that plan won't work as stated. Alternatively, since $\neg S \vee U$ is given as an "$\vee$", we can instead try Elim $\vee$ (which is secretly just Modus Ponens wearing a fake mustache).

Hint: Prove that $\neg\neg S$ holds using Reductio Ad Absurdum. Then, you can apply Elim $\vee$.

d) Given $P \wedge U$ and $(Q \vee S) \rightarrow R$, it follows that $(P \rightarrow Q) \rightarrow (R \wedge U)$.

Your proof is only allowed to use the rules the Modus Ponens, Direct Proof, Intro $\wedge$, Elim $\wedge$, Intro $\vee$, and Elim $\vee$. Equivalent is not allowed.

e) Given $P \wedge U$ and $(Q \vee S) \rightarrow R$, it follows that $P \rightarrow (Q \rightarrow (R \wedge U))$.

Your proof is only allowed to use the rules the Modus Ponens, Direct Proof, Intro $\wedge$, Elim $\wedge$, Intro $\vee$, and Elim $\vee$. Equivalent is not allowed.

Note that the **only** difference from part (d) is that we have moved the parentheses. We went from $(P \rightarrow Q) \rightarrow (R \vee U)$ to $P \rightarrow (Q \rightarrow (R \wedge U))$, but these are two very different statements! The former is an implication with another implication in its *premise*, while this is an implication with another implication in its *conclusion*. In part (d), using Direct Proof gives us an assumption that is an implication. Here, Direct Proof will give us P as an assumption, which is simpler, but since the conclusion is another implication, we need to use another Direct Proof, nested within the first one!

Submit and check your **formal proofs** here:

http://cozy.cs.washington.edu

You can make as many attempts as needed to find a correct answer.

Task 4 – One False Prove [12 pts]

For each of the claims below, (1) translate the English proof into a formal proof and (2) say which of the following categories describes the formal proof:

Proof The proof is correct.

Goof The claim is true but the proof is wrong.

Spoof The claim is false.

Finally, (3) if it is a goof, point out the errors in the proof and explain how to correct them, and if it is a spoof, point out the *first* error in the proof and then show that the claim is false by giving a counterexample. (If it is a correct proof, then skip part (3).)

Be careful! We want you to translate the English proof to a formal proof as closely as possible, including translating the mistake(s), if any! Also, an incorrect proof does not necessarily mean the claim is false, i.e., a goof is not a spoof!

Note that English proofs often skip steps that would be required in formal proofs. (That is fine as long as it is easy for the reader to see what needs to be filled in.) Skipped steps do not mean that the proof is incorrect. The proof is incorrect when it asserts a fact that is not necessarily true or does not follow by the reason given.

Hint: To give a counterexample to a claim in propositional logic, describe what truth values each atomic variable should have so that all the givens are true but the result is false.

a) **Claim**: Given $\neg a \wedge \neg b$ and $a \wedge b \to c$, it follows that $\neg c$ holds.

Proof or Spoof: Since $\neg a$ holds from the first given, we have $\neg a \vee \neg b$ is true. Observe that this is equivalent to $\neg(a \wedge b)$. The contrapositive of the second given is $\neg(a \wedge b) \to \neg c$. Therefore, $\neg c$ must follow.

b) **Claim**: Given $p \vee r$, $p \to \neg q$ and $q \leftrightarrow r$, it follows that $p \oplus r$ holds.

Proof or Spoof: First, we show $\neg(p \wedge r)$ holds. For contradiction, assume $p \wedge r$ is true. Since p is true, $\neg q$ follows from the second given. Since we have $\neg q$, it must be that $\neg r$ holds from the third given. Since both r and $\neg r$ holds, we have a contradiction! Thus, $\neg(p \wedge r)$ holds. Combined with the first given, this is equivalent to $p \oplus r$.

Hint: At least one of the Latin rules might be useful!

Task 5 – Get a Prove On [20 pts]

For each of the following, write a **formal proof** that the claim holds.

Your proof is allowed to use the basic six rules of Propositional Logic (Modus Ponens, Direct Proof, Intro $\wedge$, Elim $\wedge$, Intro $\vee$, and Elim $\vee$ or Cases), Equivalent, and the four rules for Predicate Logic (Intro $\forall$, Elim $\forall$, Intro $\exists$, Elim $\exists$).

Let $P(x)$, Q(x), and $R(x, y)$ be predicates defined in some fixed domain of discourse, and let c be some well-known constants in that domain.

a) Given $\exists\, x, P(x)$, $\forall\, x, (R(x, c) \leftrightarrow R(c, x))$, and $\forall\, x, (P(x) \rightarrow R(c, x))$, it follows that we must have $\exists\, d, (R(d, c) \wedge R(c, d))$.

b) Given $\forall\, x, (P(x) \wedge R(x, c))$ and $\forall\, x, \forall\, y, (R(x, y) \rightarrow R(y, x))$, it follows that $\forall\, x, \exists\, y, R(y, x)$.

c) Given $\forall\, x, \exists\, y, (P(x) \wedge R(x, y))$, it follows that $\forall\, x, (P(x) \wedge (\exists\, y, R(x, y)))$.

The fact that we can *move* the $\exists$ outside of the $\wedge$ was noted (but not proven) in lecture. In this problem, you will prove that you can sometimes move an $\exists$ inside of a $\wedge$.

d) Given $\forall\, x, (P(x) \rightarrow Q(x))$, it follows that $(\forall\, x, P(x)) \rightarrow (\forall\, x, Q(x))$.

In Homework 2 Task 6, you were asked to *explain* why the latter fact follows from the former one. In this problem, you are asked to prove it using our rules!

Hints: The claim to be proven is an "$\rightarrow$", so your last step should be Direct Proof. The conclusion of that implication is a "$\forall$"', so your second to last step should be Intro $\forall$. This means that your proof will have a subproof within a subproof!

e) Given $\forall\, x, (P(x) \rightarrow R(c, x))$ and $\forall\, x, \forall\, y, ((R(x, y) \wedge R(y, x)) \rightarrow Q(y))$, it must be the case that $(\forall\, x, (P(x) \wedge R(x, c))) \rightarrow (\forall\, x, Q(x))$.

Hints: As in part (d), the claim to be proven is an "$\rightarrow$" with a "$\forall$" in its conclusion, so your last two steps should be Intro $\forall$ and Direct Proof. Once again, you will have subproofs nested two deep.

Submit and check your **formal proofs** here:

http://cozy.cs.washington.edu

You can make as many attempts as needed to find a correct answer.

Important: Cozy uses low precedence quantifiers "$\forall x,$" and "$\exists x,$", rather than the high precedence version shown in the lecture slides. The extra "," is an indicator of the difference.

Task 6 – Bust a Prove [12 pts]

For each of the claims below, (1) translate the English proof into a formal proof and (2) say which of the following categories describes the formal proof:

Proof The proof is correct.

Goof The claim is true but the proof is wrong.

Spoof The claim is false.

Finally, (3) if it is a goof, point out the errors in the proof and explain how to correct them, and if it is a spoof, point out the *first* error in the proof and then show that the claim is false by giving a counterexample. (If it is a correct proof, then skip part (3).)

Be careful! We want you to translate the English proof to a formal proof as closely as possible, including translating the mistake(s), if any! Also, an incorrect proof does not necessarily mean the claim is false, i.e., a spoof is not a goof!

Note that English proofs often skip steps that would be required in formal proofs. (That is fine as long as it is easy for the reader to see what needs to be filled in.) Skipped steps do not mean that the proof is incorrect. The proof is incorrect when it asserts a fact that is not necessarily true or does not follow by the reason given.

Hint: To give a counterexample to a claim in predicate logic, describe a domain of discourse and definitions for all predicates such that all the givens are true but the result is false.

a) For this part, let the domain of discourse be the integers.

Claim: Given $\exists x\, \exists y\, P(x, y)$ and $\forall x\, \forall y\, (P(x, y) \to Q(x, y))$, it must follow that $\exists x\, \forall y\, Q(x, y)$

Proof or Spoof: We know $P(a, b)$ from the first given for some integers a and b. Then, $Q(a, b)$ follows from the second given. Since the second given says $P(x, y) \to Q(x, y)$ for all integers x and y, we have $Q(a, y)$ for all integers y. Finally, we conclude that there's some integer x such that for all integers y, $Q(x, y)$ holds.

b) **Claim**: Given $\forall y\, (Q(y) \to \exists x\, R(x, y))$ and $\forall x\, (P(x) \to \forall y\, S(x, y))$, it must follow that $\forall x\, \big((P(x) \wedge Q(x)) \to \exists z\, (R(x, z) \wedge S(z, x))\big)$.

Proof or Spoof: Let a be an arbitrary P that is also a Q. By the second given, since a is a P, there exists b such that $S(a, b)$. By the first given, since a is a Q, we also have $R(b, a)$. Since a was arbitrary, the claimed result follows.

Task 7 – Extra Credit: Put That In Your Type and Smoke It [0 pts]

In this problem, we will extend the machinery we used in Homework 1's extra credit problem in two ways. First, we will add some new instructions. Second, and more importantly, we will add *type information* to each instruction.

Rather than having a machine with single bit registers, we will imagine that each register can store more complex values such as

Primitives These include values of types int, long, float, boolean, char, and String.

Pairs of values The type of a pair is denoted by writing "$\times$" between the types of the two parts. For example, the pair $(1, \text{true})$ has type "int $\times$ boolean" since the first part is an int and the second part is a boolean.

Functions The type of a function is denoted by writing a "$\rightarrow$" between the input and output types. For example, a function that takes an int and returns a String is written "int $\rightarrow$ String".

We add type information, describing what is stored in each each register, in an additional column next to the instructions. For example, if R_1 contains a value of type int and R_2 contains a value of type int $\rightarrow$ (String $\times$ int), i.e., a function that takes an int as input and returns a pair containing a String and an int, then we could write the instruction

$$R_3 := \texttt{CALL}(R_1, R_2) \qquad\qquad \text{String} \times \text{int}$$

which calls the function stored in R_2, passing in the value from R_1 as input, and stores the result in R_3, and write a type of "String $\times$ int" in the right column since that is the type that is now stored in R_3.

In addition to `CALL`, we add new instructions for working with pairs. If R_1 stores a pair of type String $\times$ int, then $\texttt{LEFT}(R_1)$ returns the String part and $\texttt{RIGHT}(R_1)$ returns the int part. If R_2 contains a char and R_3 contains a boolean, then $\texttt{PAIR}(R_2, R_3)$ returns a pair of containing a char and a boolean, i.e., a value of type char $\times$ boolean.

a) Complete the following set of instructions so that they compute a value of type int $\times$ float in the last register assigned (R_N for some N):

R_1	int $\times$ boolean
R_2	char
R_3	(boolean $\times$ char) $\rightarrow$ (String $\times$ float)
$R_4 := \ldots$	$\ldots$

The first three lines show the types **already stored** in registers R_1, R_2, and R_3 at the start, before your instructions are executed. You are free to use the values in those registers in later instructions.

Store into a *new register* on each line. **Do not reassign** any registers.

b) Compare the types listed next to these instructions to the propositions listed on the lines of your proof in Task 1(a). Give a collection of text substitutions, such as replacing all instances of "P" by "float" (these can include substitutions for atomic propositions and for operators), that will make the sequence of propositions in Task 1(a) *exactly match* the sequence of types in part (a).

Note: You may need to change your solution to part (a) slightly to make this work!

c) Now, let's add another way to form new types. If A and B are types, then $A + B$ will be the type representing values that can be of either type A or type B. For example, String + int would be a type of values that can be strings or integers.

To work with this new type, we need some new instructions. First, if R_1 has type A, then the instruction $\texttt{LCASE}(R_1)$ returns the same value but now having type $A + B$ and $\texttt{RCASE}(R_1)$ returns the same value but now having type $B + A$ (Note that we can pick any type B that we want here.)

Second, if R_2 stores a value of type $A + B$, R_3 stores a function of type $A \to C$ (a function taking an A as input and returning a value of type C), and R_4 stores a function of type $B \to C$, then the instruction $\texttt{SWITCH}(R_2, R_3, R_4)$ returns a value of type C: it looks at the value in R_2, and, if it is of type A, it calls the function in R_3 and returns the result, whereas, if it is of type B, it calls the function in R_4 and returns the result. In either case, the result is something of type C.

Complete the following set of instructions so that they compute a value of type long × char in the last register assigned:

R_1	String
R_2	(float + String) → (int + boolean)
R_3	boolean → (char × long)
R_4	int → (char × long)
$R_5 := \ldots$	$\ldots$

The first three lines again show the types of values already stored in registers R_1, R_2, and R_3. As before, do not reassign any registers. Use a new register for each instruction's result.

d) Compare the types listed next to these instructions to the propositions listed on the lines of your proof in Task 1(c). Give a collection of text substitutions, such as replacing all instances of "P" by "float" (these can include substitutions for atomic propositions and for operators), that will make the sequence of propositions in Task 1(c) *exactly match* the sequence of types in part (c). (You may need to change your solution to part (c) slightly to make this work!)

e) Now that we see how to match up the propositions in our earlier proofs with types in the code above, let's look at the other two columns. Describe how to translate each of the rules of inference used in the proofs from both Task 1(a) and (c) so that they turn into the instructions in parts (a) and (c).

f) One of the important rules **not** used in Task 1(a) or (c) was Direct Proof. What new concept would we need to introduce to our assembly language so that the similarities noted above apply could also to proofs that use Direct Proof?