CSE 311: Foundations of Computing I — Spring 2024

# Problem Set 3

Due: Wednesday, April 17th 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, 3, 6 (and optionally 7). Follow the prompt on Gradescope to link tasks to your pages.
- The instructions for submitting tasks 1, 4, and 5 appear below those individual problems.

## Task 1 – Cat on a Hot Tin Proof [20 pts]

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

**a)** Given $P \wedge Q$, $(Q \wedge R) \to V$, and $P \to (R \wedge S)$, it follows that $V$ holds.

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

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

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

**c)** Given $P \to (R \wedge \neg S)$, $\neg U \to S$, and $(R \wedge U) \to V$, it follows that $P \to V$.

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

**d)** Given $P \to (Q \to U)$ and $Q \leftrightarrow V$, it follows that $(P \wedge Q) \to (U \wedge V)$.

Your proof is only allowed to use the rules the Modus Ponens, Direct Proof, Intro $\wedge$, Elim $\wedge$, Intro $\vee$, Elim $\vee$, and Equivalent. (*Hint*: A different equivalence will be useful this time.)

**e)** Given $P \to \neg R$, $(Q \vee R) \wedge (R \vee S)$, and $Q \to ((R \vee S) \to V)$, it follows that $P \to V$.

Your proof is only allowed to use the rules the Modus Ponens, Direct Proof, Intro $\wedge$, Elim $\wedge$, Intro $\vee$, Elim $\vee$. Note that Equivalent is not allowed this time. (*Hint*: Elim $\vee$ will be useful!)

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 – Just Another Pretty Case [14 pts]

In lecture, we saw the following rules, which were described as alternatives to "Elim $\lor$":

**Cases**

$$\begin{array}{lll} A \lor B & A \to C & B \to C \\ \hline \therefore & C & \end{array}$$

**Simple Cases**

$$\begin{array}{ll} A \to C & \neg A \to C \\ \hline \therefore & C \end{array}$$

To gain some familiarity with the first rule, use it to write a **formal proof** of the following:

**a)** Given $P \land (R \lor S)$, $R \to (P \to Q)$, and $S \to (P \to Q)$, it follows that $R \lor Q$ holds.

Your proof is only allowed to use the rules the Modus Ponens, Direct Proof, Intro $\land$, Elim $\land$, Intro $\lor$, and **Cases** (not Simple Cases or Elim $\lor$).

In the rest of this problem, we will show that Cases, Simple Cases, and Elim $\lor$ are equally powerful! That is, any one of them allows you to prove the same things as the other two.

**b)** Using **Cases**, give a formal proof that the Simple Cases rule is valid. Specifically, prove that, given $A \to C$ and $\neg A \to C$, it follows that $C$ holds.

Your proof is only allowed to use the rules the Modus Ponens, Direct Proof, Intro $\land$, Elim $\land$, Intro $\lor$, and Cases (not Simple Cases or Elim $\lor$), but you can also use *Excluded Middle*.[1]

*Hint*: Your proof can be very short. Mine has only 4 lines!

**c)** Using **Simple Cases**, give a formal proof that the Elim $\lor$ rule is valid. Specifically, prove that, given $A \lor B$ and $\neg A$, it follows that $B$ holds.

Your proof is only allowed to use the rules Direct Proof, Intro $\land$, Elim $\land$, Intro $\lor$, and Simple Cases (not Cases or Elim $\lor$), but you can also use the *Law of Implication* and *Double Negation* equivalences. (Note that Modus Ponens is **not** allowed this time!)

**d)** Using **Elim** $\lor$, give a formal proof that the Cases rule is valid. Specifically, prove that, given $A \lor B$, $A \to C$, and $B \to C$, it follows that $C$ holds.

Your proof is only allowed to use the rules the Modus Ponens, Direct Proof, Intro $\land$, Elim $\land$, Intro $\lor$, and Elim $\lor$ (not Cases or Simple Cases), but you can also use "simple cases" *equivalence* you proved in Homework 2, Task 2 (d).

You can use Cozy to write and check your proof on this page (being careful to restrict the allowed rules). However, note that you must still **submit your proof in Gradescope**

[1] Recall that this allows you to infer $A \lor \neg A$ for any proposition $A$.

## Task 3 – One False Prove [18 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 $\neg q \vee \neg s \vee \neg r$ and $p \vee s$, it follows that $(r \wedge q) \rightarrow p$ holds.

*Proof or Spoof*: Assume $r$ and $q$. The first given is equivalent to $\neg(r \wedge q) \vee \neg s$. Since we know that $r$ and $q$ are true, this tells us that $\neg s$ must be true. The second given then tells us that $p$ must be true.

**b)** **Claim**: Given $p \rightarrow (\neg q \rightarrow r)$, $s \vee \neg q$, $q \rightarrow \neg s$, and $\neg p \rightarrow q$, it follows that $r$ holds.

*Proof or Spoof*: The second and third givens combined are equivalent to $\neg q$. The contrapositive of the last given is $p \rightarrow \neg q$, which combined with $\neg q$ gives us $p$. From these two, the first given results in $r$.

**c)** **Claim**: Given $s \rightarrow (p \wedge q)$, $\neg s \rightarrow r$, and $(r \wedge p) \rightarrow q$, it follows that $q$ holds.

*Proof or Spoof*: First, note that $s \rightarrow q$ holds, since, assuming $s$, from given $s \rightarrow (p \wedge q)$ we can get $p$ and also $q$. Second, note that $\neg s \rightarrow q$ holds, since, assuming $\neg s$, from given $\neg s \rightarrow r$ we can get $r$, and we already saw that $p$ holds, so together we have $r \wedge p$, from which $q$ follows by the last given. Finally, either $s$ or $\neg s$ is true, and $q$ follows either from our first claim or second claim above, respectively.

## Task 4 – Latin Down the Hatches [20 pts]

In this problem, we will try working without the Tautology or Equivalence rules (which are "expensive").

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

Your proof is only allowed to use the most basic six rules of Propositional Logic (Modus Ponens, Direct Proof, Intro $\wedge$, Elim $\wedge$, Intro $\vee$, and Elim $\vee$ or Cases) and the four Latin rules (Principium Contradictionis, Reductio Ad Absurdum, Ex Falso Quodlibet, and Ad Litteram Verum).

**a)** Given $A$, it follows that $\neg\neg A$ holds. This is one direction of **Double Negation**.

**b)** Given $A \to B$, it follows that $\neg B \to \neg A$. This is one direction of **Contrapositive**.

**c)** Given $\neg A \vee B$, it follows that $A \to B$. This is one direction of **Law of Implication**.

**d)** Given $\neg A \wedge \neg B$, it follows that $\neg(A \vee B)$ holds. This is one direction of **De Morgan's Law**.

**e)** Given $\neg A \vee \neg B$, it follows that $\neg(A \wedge B)$ holds. This is one direction of **De Morgan's Law**.

Note that each of the inferences proven above is actually an equivalence. However, the other directions cannot be proven without Tautology! Hence, a proof system without the "expensive" Tautology rule is not able to prove all facts that are true in Propositional Logic.

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 5 – Provin' Right Along [20 pts]

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

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

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

**a)** Given $\exists x\, P(x)$, $\forall x\, Q(x, c)$ and $\forall x\, (P(x) \to Q(c, x))$, it follows that $\exists a\, \exists b\, (Q(a, b) \wedge Q(b, a))$.

**b)** Given $\forall x\, Q(c, x)$ and $\forall x\, \forall y\, (Q(x, y) \to Q(y, x))$, it follows that $\forall a\, \exists b\, (Q(a, b) \wedge Q(b, a))$.

**c)** Given $\forall a\, (P(a) \wedge \exists b\, Q(b, a))$, it follows that $\forall a\, \exists b\, (P(a) \wedge Q(b, a))$.

The fact that we can *move* the $\exists$ outside of the $\wedge$ was noted but not proven in lecture. In this problem, you will show that you can prove it with our rules!

**d)** Given $\forall a\, ((\exists b\, Q(b, a)) \to P(a))$, it follows that $\forall a\, \forall b\, (Q(b, a) \to P(a))$.

Note that, this time, when we *move* the $\exists$ outside of the "$\to$", it turns into a $\forall$! This shows that you cannot blindly move $\exists$ (or $\forall$) around other operators and assume the meaning is unchanged.

*Hints*: The statement to be proved includes two $\forall$s and a "$\to$". To prove those facts using introduction rules will require subproofs nested three levels deep! (Note that you can Intro $\forall$ two variables at once in Cozy, though, which would require only subproofs that are nested two levels.)

**e)** Given $\exists a\, (P(a) \to \forall b\, Q(b, a))$, it follows that $\forall b\, \exists a\, (P(a) \to Q(b, a))$.

Note that the conclusion has the $\exists$ and $\forall$ in the opposite order! As we discussed in lecture, changing quantifier order does not usually preserve the meaning. Here, however, we are not claiming that the given and conclusion are equivalent but rather that the given merely *implies* the conclusion.

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 a slightly different notation with quantifiers ($\forall$ and $\exists$). Rather than writing "$\forall x\, P(x) \wedge Q(x)$" like the slides, it writes "$\forall\, x, P(x) \wedge Q(x)$" with an extra *comma*. And, while the slides notation means "$(\forall x\, P(x)) \wedge Q(x)$" (high precedence), Cozy's notation means "$\forall\, x, (P(x) \wedge Q(x))$" (low precedence). We will attempt to consistently use the comma to distinguish between notations.

(While it seems strange now, Cozy's notation is actually the usual version in computer science. . . )

## Task 6 – Bust a Prove [18 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)** **Claim**: Given $\forall y\,(Q(y) \rightarrow \forall x\,R(x,y))$ and $\forall x\,(P(x) \rightarrow \exists y\,R(x,y))$, it must follow that $\forall x\,\big((P(x) \land Q(x)) \rightarrow \exists z\,(R(z,x) \land R(x,z))\big)$.

*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 $R(a,b)$. By the first given, since $a$ is a $Q$, we also have $R(b,a)$. Since $a$ was arbitrary, the claimed result follows.

**b)** For this part, assume the domain of discourse is the integers.

**Claim**: Given $\forall x\,\exists y\,R(x,y)$ and $\forall x\,\forall y\,(R(x,y) \rightarrow R(y,x))$, it follows that $\exists x\,R(x,x)$.

*Proof or Spoof*: By the first given plugging in 7 for $x$, we get that there exists $b$ such that $R(7,b)$. By the second given, since $R(7,b)$, we can plug in 7 for both $x$ and $y$, and we get $R(7,7)$. The result follows.

**c)** **Claim**: Given $\forall x\,\forall y\,(R(x,y) \rightarrow R(y,x))$ and $\forall x\,\forall y\,\forall z\,((R(x,y) \land R(y,z)) \rightarrow R(x,z))$, it follows that $\forall x\,\forall y\,(R(x,y) \rightarrow R(x,x))$.

*Proof or Spoof*: Let $a$ and $b$ be arbitrary and assume $R(a,b)$. By the first given, we have $R(b,a)$. By the second given, plugging in $b$ for $x$, $a$ for $y$, and $b$ for $z$, we get $R(b,b)$. Since $a$ and $b$ were arbitrary, the 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, 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 char in the last register assigned ($R_N$ for some $N$):

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

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 Problem 1(a). Give a collection of text substitutions, such as replacing all instances of "$P$" by "String" (these can include substitutions for atomic propositions and for operators), that will make the sequence of propositions in Problem 1(a) *exactly match* the sequence of types in part (a). (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 `CASE`$(R_1)$ returns the same value but now having type $A + B$. (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 `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 float + boolean in the last register assigned:

| | |
|---|---|
| $R_1$ | int × (float + String) |
| $R_2$ | float → (int → boolean) |
| $R_3$ | String → (int → boolean) |
| $R_4 := \ldots$ | $\ldots$ |

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 Problem 2(a). Give a collection of text substitutions, such as replacing all instances of "$P$" by "String" (these can include substitutions for atomic propositions and for operators), that will make the sequence of propositions in Problem 2(a) *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 Problem 1(a) and 2(a) so that they turn into the instructions in parts (a) and (c).

**f)** One of the important rules **not** used in Problems 1(a) or 2(a) 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?