Submission Phase

1. Do assignment ✓ (/logic2-002/human_grading/view/courses/974087/assessments/3/submissions)

Evaluation Phase

2. Evaluate peers ✓ (/logic2-002/human_grading/view/courses/974087/assessments/3/peerGradingSets)
3. Self-evaluate ✓ (/logic2-002/human_grading/view/courses/974087/assessments/3/selfGradingSets)

Results Phase

4. See results ✓ (/logic2-002/human_grading/view/courses/974087/assessments/3/results/mine)

This is an exercise in showing your understanding of how first order predicate logic works and how it may be applied. This question will take you beyond the skills we have tested in the quizzes for the subject, into a deeper level of understanding of logic. You will need to reflect on what you have learned, you might need to re-watch some lectures or explore ideas in the notes, and experiment with ideas until things become clear. Read each question part, think about them your answers, and after you have worked with the ideas for a while, write your answer as clearly as you can. Good luck!

(Use our one page grading rubric (https://d396qusza40orc.cloudfront.net/logic1%2Fassessment%2FLLI%20Peer%20Grading%20Rubric.pdf) to inform your own writing as you formulate your answers. These are the criteria you will use in assessing other students' answers, and they are the criteria the your readers will use to assess yours.)

This question is in three parts.

Part 1 (write approximately 500 words)

The formula $(\forall x)(\exists y)(\forall z)(Lxy \,\&\, Myz)$ is true in the model $\mathfrak{M}$ with domain $D = \{a, b, c\}$, where we interpret the predicates L and M like this:

$I(L)$	a	b	c
a	1	1	0
b	1	1	1
c	0	0	1

$I(M)$	a	b	c
a	1	1	1
b	1	0	0
c	1	1	1

We can say something similar to the formula $(\forall x)(\exists y)(\forall z)(Lxy \,\&\, Myz)$ without using an existential quantifier. We can find a function f, interpreted on the domain D, in such a way that the formula $(\forall x)(\forall z)(Lxf(x) \,\&\, Mf(x)z)$ is also true.

- Find such an interpretation for the function f that makes this formula true.

	$I(f)$
a	
b	
c	

(If you need a refresher on function symbols and how they work, check the first part of Lecture 4.5 (https://class.coursera.org/logic2-002/lecture/57) and Section 4.5 of the notes (https://d28rh4a8wq0iu5.cloudfront.net/logic2/notes/logic2notes.pdf).)

- Use the reasoning you went through in finding your function f to explain why any model at all in which $(\forall x)(\exists y)(\forall z)(Lxy \,\&\, Myz)$ is true can be converted into a model in which $(\forall x)(\forall z)(Lxf(x) \,\&\, Mf(x)z)$ is also true (by providing some interpretation for the function symbol f); and *vice versa*, that any model $(\forall x)(\forall z)(Lxf(x) \,\&\, Mf(x)z)$ is true, can be made into a model in which $(\forall x)(\exists y)(\forall z)(Lxy \,\&\, Myz)$ is true, too.
- Explain, then, why the function symbol f in this case depends on the one variable, x, but not on the remaining variable, z.
- Finally, explain why the two formulas, $(\forall x)(\exists y)(\forall z)(Lxy \,\&\, Myz)$ and $(\forall x)(\forall z)(Lxf(x) \,\&\, Mf(x)z)$ are *not* logically equivalent, even though a model in which one is true can be made into a model in which the other is true, too. (Remember: to show that two formulas are not equivalent, you just need to find one model in which one of the formulas is true and the other is not.)

Part 2 (write approximately 500 words)

This fact is *general*.

- Using the insight you've gained from looking at the previous example. explain in your own words how for any formula in the language of predicate logic of the form

  $(Q_1 x_1)(Q_2 x_1) \cdots (Q_n x_n) M(x_1, \ldots, x_n)$

  we can construct an a formula in which there are *no existential quantifiers*, which is satisfiable (true in some model) if and only if the original formula is. (Hint: we replace each existential quantifier Q_i in the string $(Q_1 x_1)(Q_2 x_2) \cdots (Q_n x_n)$ by a function f_i appearing in the $M(x_1, \ldots, x_n)$ part of the formula.)

  In your answer, be careful to explain which variables are replaced by functions (such as y being replaced by $f(x)$ in the example in Part 1) and which remaining variables are needed in the definition of each function. (*Consider*: why, in Part 1, does $f(x)$ depend on x but not on the variable z?)
- Once you have explained this, explain in your own words how you could convert *any* formula in the language of predicate logic into a logically equivalent formula $(Q_1 x_1)(Q_2 x_2) \cdots (Q_n x_n) M(x_1, \ldots, x_n)$ in which the quantifiers are all at the front.

(This process of eliminating existential quantifiers is called *Skolemiation*, after Thoralf Skolem (http://en.wikipedia.org/wiki/Thoralf_Skolem), the Norwegian logician and mathematician. The function f used to replace the existential quantifier is said to be a *Skolem Function*.)

Part 3 (write approximately 500 words)

Finally, consider one of the application areas you have examined in the subject (Computer Science, Linguistics, Digital Systems, Mathematics, Philosophy) and think of the role of existential quantifiers in formulas in the discipline you have chosen, and answer the following question for that discipline.

- In what ways could an existential quantifier be replaced by a Skolem function, and how could this replacement change the meaning of the expression that has been translated in this way?

  You might like to think about the following topics. These are some *examples* of how Skolem functions connect with each of the different disciplines we have studied.

  - *Computer Science*: Prolog databases do not contain existentially quantified formulas. (You can’t specify in a database, for example, that everyone has some mother.) How could you use a Skolem function to represent existentially quantified information? Does this representation change the meaning of the information you are attempting to encode?
  - *Linguistics*: Consider how we have represented quantifiers and pronouns in donkey sentences and other complex constructions in natural languages. Is there a different way to represent the logical structure of such sentences using Skolem functions? Does a representation like this do a better or a worse job of exposing the logical structure of natural language sentences?
  - *Digital Systems*: Consider the “*eventually*” operator $\Diamond$ from Linear Temporal Logic, where $\Diamond A$ states that at some time in the future A is true. What difference would it make to have a Skolem function specifying a time at which A becomes true? What could you state in a temporal logic with Skolem functions that you could not state in $\mathrm{LTL}^1_\ominus$?
  - *Mathematics*: The statement for the convergence of a sequence crucially uses an existential quantifier after a universal quantifier (it is a $\forall\exists\forall$ formula). What can you state about sequence convergence using Skolem functions?
  - *Philosophy*: If $(\forall x)(\exists y)(Fy \mathbin{\&} Lxy)$ is true, then with a Skolem function f we have $(\forall x)(Ff(x) \mathbin{\&} Lxf(x))$. Since f is a *function*, it follows that there for each x is a *unique* y such that $Fy \mathbin{\&} Lxy \mathbin{\&} y = f(x)$. For each true existentially quantified formula, there is a corresponding true definite description, picking out a distinguished witness of that formula. Is this a plausible idea? Is there, for every true existentially quantified formula, some way to pick out a single individual that satisfies that formula?

Be careful to write your answer in a way which could be understood by someone who did not focus on the same topic area as you. You can assume basic understanding of predicate logic, but do not assume detailed understanding of the particular application area.

PART I.

1. Interpretation of f(x) in the given particular domain only: with D = {a, b, c} and the provided truth tables L and M:
   - Forward direction: (∀x)(∃y)(∀z)(Lxy&Myz) is TRUE can be converted to (∀x)(∀z)(Lxf(x)&Mf(x)z) is TRUE. In the first formula: (∀x)(∃y)(∀z)(Lxy&Myz) if we can find the specific value y = y" so that (∀x)(∀z)(Lxy" & My"z) is TRUE, then, as the predicate L and M are always true with ALL the rest variables x and z ((∀x)(∀z)), without loosing generalization, we can substitute y" = f(x) (a function symbol) which takes the value in domain D. Then *if with the specific y", we can always make My"z true with ALL values z*, then the original formula can be converted to (∀x)(∀z)(Lxf(x) & Mf(x)z) which is also TRUE.
   - Reverse direction: (∀x)(∀z)(Lxf(x) & Mf(x)z) is TRUE, then because the formula is true with (∀x) and (∀z), then we can neglect the value z and assign f(x) = y which is a free value/variable. Assigning y with a particular value in domain D, *if it is the case when with a particular value f(x), we can make the predicate Lxf(x) true with ALL x*, then we can have (∀x)(∃y)(∀z)(Lxy&Myz) is also TRUE. In this given case, we can always find such x, f(x) and z as below.
   - Finding f: if x = a, we have y = a so that Lxy = TRUE then Myz is true with (∀z); if x = c, y = c then Lxy = TRUE then Myz = TRUE with (∀z); if x = b then y can be either a or c so that Lxy = TRUE and we can applied the 2 mentioned cases. Here we base on the fact that Myz is true with ALL values z if and only if y = f(x) a or c then we use this fact to consider the formula Lxy, Eventually, we can find the function f as below:

	I(f)
a	**a**
b	**a or c**
c	**c**

2. Reasoning why the function symbol f depends on only variable x but not z: taking a look into the formula: (∀x)(∃y)(∀z)(Lxy&Myz) converted into (∀x)(∀z)(Lxf(x) & Mf(x)z). This can be easily proven by reasoning from inside to outside for the formula (∀x)(∀z), we consider z first then x. As mentioned above, this formula is true with (∀z). Likewise, for any given value z taken from domain D, we can also find the particular value y" = f(x) so that My"z = 1. As a result, we only consider the value y" in domain D such that it makes the formula Lxy" is TRUE and we are done.

In other words, the variable y is outside the scope of variable z, and by processing inside to outside, we have y independent to z but no x, so y = f(x) but not f(x, z). Consequently, with the condition (∀z) in the inner scope, we can make our substitution simpler by just consider the only dependent variable is the x, neglecting z. The proof is thus done.

3. Explain why the two formulas, (∀x)(∃y)(∀z)(Lxy&Myz) and (∀x)(∀z)(Lxf(x)&Mf(x)z) are *not* logically equivalent. We can change the model so that f(x) is not the case in part 1. For example, we have f(x:=b) = b (previously taken value a or c), thus the first formula (∀x)(∃y)(∀z)(Lxy&Myz), which is not contingent on f(x), is always true. However, consider the second one: (∀x)(∀z)(Lxf(x)&Mf(x)z). We can easily detect that Mf(x)z := Mbz is not true with (∀z) (e.g. Mbb = Mbc = 0). Consequently, the given pair of formula are not logically equivalent.

PART II.
Consider the formula (Q1x1)(Q2x1)···(Qnxn)M(x1,…,xn) where Qi are the quantifiers.

1. Forward direction: We use same operation taken in part 1, where the existence quantifier (∃y) is removed with a replacement of f(x) which takes a particular value (or a constant). One important criterion to consider is the **scope of quantifier** and we only to replace only the existential quantifier with the function symbol only. By considering the scope in the correct order, we scan through the quantifier from Qn -> Qn-1 -> ... -> Q2 -> Q1, and as we process over a quantifier Qi (and its associated variable xi), the scope is narrowed down and the dependent variables when we replace any existence quantifier onward are also narrowed down accordingly.

For example, if Qn is a existential quantifier, then we can always, without loosing the generalization and accurateness of the formula M, we replace xn = f(x1, x2, ..., xn-1) as Qn is the most inner scope. However, at Qn-1, if it's also a existential quantifier, then we can also replace xn-1 = f(x1, x2, ..., xn-2). Here the variable xn is neglected as it's independent to xn-1 (or outside the scope of effectiveness). To generalize, if the respective quantifier Qi is existential, we can replace xi = f(x1, x2, ..., xi-1) with all i from 1 to n.

By achieving the elimination of all existence quantifiers among the Qn, Qn-1, ..., Q2, Q1, we can come up with the converted formula with only universal quantifiers (if existed among the Qi), with the correctness of the formula conserved.

2. Reverse direction: on the contrary, we can convert *any* formula in the language of predicate logic into a logically equivalent formula (Q1x1)(Q2x2)···(Qnxn)M(x1,…,xn) by exactly the inverse process. For example with the predicate logic Qab with a, b are the names, we can replace b with f(a) as b is dependent to a, and the formula is now become (∃a)Qaf(a). With many other variables, we also take into consideration the scope from smallest to largest when replacing the constant with the variable to make the dependent relationship is replicated.

PART III. Computer Science (PROLOG)

To recall, in PROLOG, we have some primitive rules for writing the programs with facts and rules.

- (1) To define the fact (a constant), we use the form attr(c) with the interpretation as constant c has the attribute a. With 2 predicate form, we use the same thing with one more constant, for example love(a, b) for "a loves b".
- (2) To define the variable, we use capital letter X and we define the rule such as: q(X,Y):- p(X,Y), which will be translated as "for all X and all Y, if q(X,Y) then p(X,Y).

Now given the case where we need to define the formula with existential quantifier in PROLOG, with the straightforward reasoning, we should use the case (1) where we replace with the fact with a constant. For example, we want to express: "there exists a tree with red leaves" with red_leaves(tree_a). Of course in this translation, the existential quantifier must be transformed to the particular constant defined by the programmer (here is tree_a, although tree_a has no meaning in the real case but it has to be defined) which is required by PROLOG.

The case becomes more complicated if the order of the predicate logic increases. For example, there exists a variable x having property **p** that for every variable Y, the fact red_leaves(x, Y) is always true. In this case, we should apply the similar process to turn the rule into the clause with only existential quantifier by employing the method of

Skolem function. For example, we define **p(a).** and then **red_leaves(a, Y).**

In the more complicated where there are more universal dependent quantifiers, we should replace one by one, from the smaller cope to the bigger scope as describe above, in part II.

In conclusion, it's NOT the inherent mechanism of PROLOG to support directly the existential quantifier for programmer. However, by defining the constant by himself, a programmer can get around this obstacle and still providing the efficient program to make the target. Because of this limitation, the very complex relationship could make the program more verbose and difficult to eliminate the bug. Despite this difficulty, there are numerous significant cases in which PROLOG can solve the huge problem such as the example of IBM WATSON supercomputer.
