Problem Set 2

This second problem set explores mathematical logic. We've chosen the questions here to help you get a more nuanced understanding for what first-order logic statements mean (and, importantly, what they don't mean) and to give you a chance to apply first-order logic in the realm of proofs. By the time you've completed this problem set, we hope that you have a much better grasp of mathematical logic and how it can help improve your proofwriting structure.

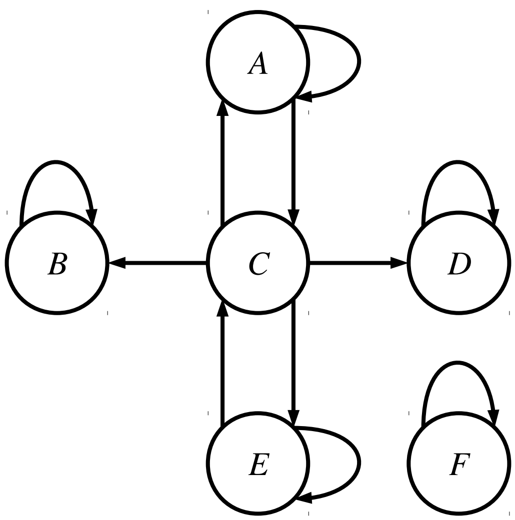
Before attempting this problem set, we recommend that you do the following:

- Familiarize yourself with the online Truth Table Tool and play around with it a bit to get a feel for the propositional connectives.
- Read the online "Guide to Negating Formulas" for more information about how to find the negations of propositional and first-order formulas.
- Read the online "Guide to Logic Translations" for more information about how to translate statements into first-order logic.

Checkpoint Questions Due Monday, April 17th at the start of class.
Remaining Questions Due Friday, April 21st at the start of class.

Write your solutions to the following problems and submit them electronically on GradeScope by this Monday, April 17th at the start of lecture. As before, these problems will be graded on a 0/1/2 scale based on whether or not you have made a good, honest effort to complete all of them. We will try to get these problems returned to you with feedback on your proof style this upcoming Wednesday.

Checkpoint Problem: Interpersonal Dynamics (2 Points if Submitted)



The diagram to the left represents a set of people named A, B, C, D, E, and F. If there's an arrow from a person x to a person y, then person x loves person y. We'll denote this by writing $Loves(x, y)$. For example, in this picture, we have $Loves(C, D)$ and $Loves(E, E)$, but not $Loves(D, A)$.

There are no "implied" arrows anywhere in this diagram. For example, even though A loves C and C loves E, the statement $Loves(A, E)$ is false because there's no direct arrow from A to E. Similarly, even though C loves D, the statement $Loves(D, C)$ is false because there's no arrow from D to C.

Below is a list of 16 formulas in first-order logic about the above picture. In those formulas, the letters A through F refer to individual people, and P represents the set of all the people. For each formula, determine whether that formula is true or false. No justification is necessary.

i. $Loves(A, C) \lor Loves(D, C)$

ii. $Loves(C, B) \lor Loves(C, D)$

iii. $Loves(B, E) \to Loves(D, A)$

iv. $Loves(A, E) \to Loves(A, A)$

v. $Loves(A, C) \to Loves(E, C)$

vi. $Loves(C, B) \to Loves(B, C)$

vii. $Loves(A, B) \leftrightarrow Loves(E, E)$

viii. $Loves(C, A) \leftrightarrow Loves(B, B)$

ix. $\forall x \in P.\ Loves(x, x)$

x. $\exists x \in P.\ Loves(x, x)$

xi. $\forall x \in P.\ \forall y \in P.\ (Loves(x, y) \leftrightarrow Loves(y, x))$

xii. $\exists x \in P.\ \exists y \in P.\ (Loves(x, y) \leftrightarrow Loves(y, x))$

xiii. $\forall x \in P.\ (Loves(x, x) \to \exists y \in P.\ Loves(x, y))$

xiv. $\forall x \in P.\ (Loves(x, x) \to \exists y \in P.\ (Loves(x, y) \land x \neq y))$

xv. $\exists x \in P.\ (Loves(x, x) \land \forall y \in P.\ Loves(x, y))$

xvi. $\exists x \in P.\ (Loves(x, x) \to \forall y \in P.\ Loves(x, y))$

The remainder of these problems should be completed and submitted online by Friday, April 21st at the start of class.

Problem One: Implies and False

Although propositional logic has many different connectives, it turns out that any formula in propositional logic can be rewritten as an equivalent propositional formula that uses only the ¬, ∧, and ⊤ connectives. (You don't need to prove this). In this problem, you will prove a different result: every formula in propositional logic can be rewritten as an equivalent logical formula purely using the → and ⊥ connectives.

i. Find a formula that's logically equivalent to $\neg p$ that uses only the variable p and the → and ⊥ connectives. No justification is necessary.

ii. Find a formula that's logically equivalent to ⊤ that uses only the → and ⊥ connectives. No justification is necessary.

iii. Find a formula that's logically equivalent to $p \land q$ that uses only the variables p and q and the → and ⊥ connectives. No justification is necessary.

Since you can express ¬, ∧, and ⊤ using just → and ⊥, every possible formula in propositional logic can be expressed using purely the → and ⊥ connectives. Nifty!

Problem Two: Ternary Conditionals

Many programming languages support a *ternary conditional operator*. For example, in C, C++, and Java, the expression x `?` y `:` z means "evaluate the boolean expression x. If it's true, the entire expression evaluates to y. If it's false, the entire expression evaluates to z."

In the context of propositional logic, we can introduce a new ternary connective `?:` such that p `?` q `:` r means "if p is true, the connective evaluates to the truth value of q, and otherwise it evaluates to the truth value of r."

i. Based on this description, write a truth table for the `?:` connective.

ii. Find a propositional formula equivalent to p `?` q `:` r that does not use the `?:` connective. Justify your answer by writing a truth table for your new formula.

It turns out that it's possible to rewrite any formula in propositional logic using only `?:`, ⊤, and ⊥. The rest of this question will ask you to show this.

iii. Find a formula equivalent to $\neg p$ that does not use any connectives besides `?:`, ⊤, and ⊥. No justification is necessary.

iv. Find a formula equivalent to $p \to q$ that does not use any connectives besides `?:`, ⊤, and ⊥. No justification is necessary.

Since all remaining connectives can be written purely in terms of ¬ and →, any propositional formula using the seven standard connectives can be rewritten using only the three connectives `?:`, ⊤, and ⊥.

The fact that all propositional formulas can be written purely in terms of `?:`, ⊤, and ⊥ forms the basis for the *binary decision diagram*, a data structure for compactly encoding propositional formulas. Binary decision diagrams have applications in program optimization, graph algorithms, and computational complexity theory. Take CS166 or CS243 for more info!

Problem Three: Executable Logic

Consider the following first-order logic formula, where P is a set of people:

$$\exists x \in P.\ (Happy(x) \land Loves(x, x))$$

i. Write C++ or Java code for a method

```
boolean isFormulaTrueFor(List<Person> P)
```

that accepts as input a list of people P and returns whether the above formula is true for that particular list of people. Assume you have access to these helper methods, which have already been written for you:

```
boolean isHappy(Person p)
boolean loves(Person p1, Person p2)
```

Briefly justify your answer. You do not need to worry about efficiency, but please make sure that your code is clear and easy to understand.

ii. Repeat the above exercise with this first-order logic formula:

$$\forall x \in P.\ (Happy(x) \to Loves(x, x))$$

iii. Repeat the above exercise with this first-order logic formula:

$$\forall x \in P.\ (Happy(x) \to \\ \quad \exists y \in P.\ (Happy(y) \land Loves(x, y)) \\ )$$

iv. Repeat the above exercise with this first-order logic formula:

$$\exists x \in P.\ (Happy(x) \leftrightarrow \\ \quad \forall y \in P.\ (Loves(x, y)) \\ )$$

In the course of solving the above problems, use this syntax to iterate over the contents of a list:

```
for (Person x: P) {
    /* … do something with person x … */
}
```

You should be able to solve all these problems using standard loops, if statements, etc. and without needing to use any libraries. Feel free to write helper functions if you'd like.

Problem Four: First-Order Negations

For each of the first-order logic formulas below, find a first-order logic formula that is the negation of the original statement. Your final formula must not have any negations in it except for direct negations of predicates. For example, the negation of the formula $\forall x.\ (P(x) \to \exists y.\ (Q(x) \land R(y)))$ could be found by pushing the negation in from the outside inward as follows:

$$\neg(\forall x.\ (P(x) \to \exists y.\ (Q(x) \land R(y))))$$
$$\exists x.\ \neg(P(x) \to \exists y.\ (Q(x) \land R(y)))$$
$$\exists x.\ (P(x) \land \neg\exists y.\ (Q(x) \land R(y)))$$
$$\exists x.\ (P(x) \land \forall y.\ \neg(Q(x) \land R(y)))$$
$$\exists x.\ (P(x) \land \forall y.\ (Q(x) \to \neg R(y)))$$

Show every step of the process of pushing the negation into the formula (along the lines of what is done above). You don't need to formally prove that your negations are correct. We strongly recommend reading over the Guide to Negations before starting this problem.

i. $\exists p.\ (Problem(p) \land$
   $\forall g.\ (Program(g) \to \neg Solves(g, p))$
   $)$

ii. $\forall x \in \mathbb{R}.\ \forall y \in \mathbb{R}.\ (x < y \to$
   $\exists q \in \mathbb{Q}.\ (x < q \land q < y)$
   $)$

iii. $(\forall x.\ \forall y.\ \forall z.\ (R(x, y) \land R(y, z) \to R(x, z))) \quad \to \quad (\forall x.\ \forall y.\ \forall z.\ (R(y, x) \land R(z, y) \to R(z, x)))$

iv. $\forall x.\ \exists S.\ (Set(S) \land$
   $\forall z.\ (z \in S \leftrightarrow z = x)$
   $)$

v. $\forall k.\ (SixClique(k) \to$
   $\exists t.\ (Triangle(t, k) \land$
   $(\forall e.\ (Edge(e, t) \to Red(e)) \lor \forall e.\ (Edge(e, t) \to Blue(e)))$
   $)$
   $)$

Problem Five: This, But Not That

Below is a series of pairs of statements about a group of people. For each pair, draw a picture of a single group of people where the first statement is true about that group of people and the second statement is false about that group of people. Then, briefly justify your answer. For simplicity, please draw your pictures in the style of the one in the checkpoint problem.

	Make this statement true…	***… and this statement false.***
i.	$\forall y \in P.\ \exists x \in P.\ Loves(x, y)$	$\exists x \in P.\ \forall y \in P.\ Loves(x, y)$
ii.	$\forall x \in P.\ (Smiling(x) \lor Huge(x))$	$(\forall x \in P.\ Smiling(x)) \lor (\forall x \in P.\ Huge(x))$
iii.	$(\exists x \in P.\ Smiling(x)) \land (\exists x \in P.\ Huge(x))$	$\exists x \in P.\ (Smiling(x) \land Huge(x))$
iv.	$(\forall x \in P.\ Happy(x)) \to (\forall y \in P.\ Huge(y))$	$\forall x \in P.\ \forall y \in P.\ (Happy(x) \to Huge(y))$
v.	$\exists x \in P.\ (Loves(x, x) \to$ $\forall y \in P.\ (Loves(y, y))$ $)$	$(\forall x \in P.\ Loves(x, x)) \lor (\forall x \in P.\ \neg Loves(x, x))$

Problem Six: Translating into Logic

In each of the following, you will be given a list of first-order predicates and functions along with an English sentence. In each case, write a statement in first-order logic that expresses the indicated sentence. Your statement may use any first-order construct (equality, connectives, quantifiers, etc.), but you ***must*** only use the predicates, functions, and constants provided. You do not need to provide the simplest formula possible, though we'd appreciate it if you made an effort to do so. ☺

We *strongly* recommend reading the Guide to First-Order Logic Translations before starting this problem.

i. Given the predicate

   Natural(x), which states that x is an natural number

   and the functions

   $x + y$, which represents the sum of x and y, and
   $x \cdot y$, which represents the product of x and y

   write a statement in first-order logic that says "for any $n \in \mathbb{N}$, n is even if and only if n^2 is even." Remember that numbers are not a part of first-order logic, so you cannot use the number 2 here.

ii. Given the predicates

   Person(p), which states that p is a person;
   Kitten(k), which states that k is a kitten; and
   HasPet(o, p), which states that o has p as a pet,

   write a statement in first-order logic that says "someone has exactly two pet kittens and no other pets."

iii. The ***axiom of pairing*** is the following statement: given any two distinct objects x and y, there's a set containing x and y and nothing else. Given the predicates

   $x \in y$, which states that x is an element of y, and
   Set(S), which states that S is a set,

   write a statement in first-order logic that expresses the axiom of pairing.

iv. Given the predicates

   $x \in y$, which states that x is an element of y, and
   Set(S), which states that S is a set,

   write a statement in first-order logic that says "every set has a power set."

v. Given the predicates

   Lady(x), which states that x is a lady;
   Glitters(x), which states that x glitters;
   IsSureIsGold(x, y), which states that x is sure that y is gold;
   Buying(x, y), which states that x buys y; and
   StairwayToHeaven(x), which states that x is a Stairway to Heaven;

   write a statement in first-order logic that says "there's a lady who's sure all that glitters is gold, and she's buying a Stairway to Heaven."*

* Let's face it – the lyrics to Led Zeppelin's "Stairway to Heaven" are impossible to decipher. Hopefully we can gain some insight by translating them into first-order logic!

The remaining three problems on this problem set are designed to give you some more practice writing proofs. Remember that, when writing up your final answers, you should still obey all the proofwriting stylistic conventions we've used throughout the quarter: write in complete sentences, don't use mathematical symbols instead of plain English, etc. In particular, *please do not use first-order logic symbols or notation in your proofs*; while FOL is extremely useful for expressing definitions and reasoning about negations, it's generally not encountered in written proofs.

Problem Seven: Hereditary Sets

Let's begin with a fun little definition:

A set S is called a ***hereditary set*** if all its elements are hereditary sets.

This definition might seem strange because it's self-referential – it defines the hereditary sets in terms of other hereditary sets! However, it turns out that this is a perfectly reasonable definition to work with, and in this problem you'll explore some properties of these types of sets.

i. Given the self-referential nature of the definition of hereditary sets, it's not even clear that hereditary sets even exist at all! As a starting point, prove that there is at least one hereditary set.

ii. Prove that if S is a hereditary set, then $\wp(S)$ is also a hereditary set.

Problem Eight: Rational and Irrational Numbers

A quadratic equation is an equation of the form $ax^2 + bx + c = 0$. A ***root*** of this equation is a real number x such that $ax^2 + bx + c = 0$. For example, the quadratic equation $x^2 - 3x + 2 = 0$ has 1 and 2 as roots.

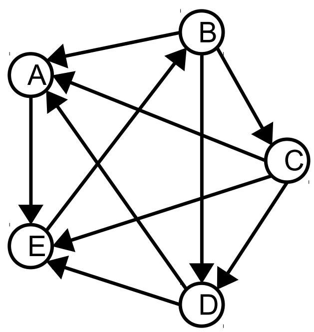
Here's an interesting fact: if a, b, and c are odd integers, then $ax^2 + bx + c = 0$ cannot have any roots that are rational numbers. Surprisingly, one of the easiest ways to prove this result is to use properties of odd and even numbers.

Prove that if a, b, and c are odd integers, then $ax^2 + bx + c = 0$ has no rational roots. Although you're probably tempted to pull out the quadratic formula here, we recommend that you *not* do this. Instead, look at what happens if you plug $x = {}^{p}/_{q}$ into the formula $ax^2 + bx + c = 0$, and consider the possible parities for p and q (the ***parity*** of a number is whether it's even or odd).

As a hint, 0 is even.

Problem Nine: Tournament Winners

Here's one more problem to help you practice your proofwriting. It's a classic CS103 problem, and we hope that you enjoy it!



A ***tournament*** is a contest among n players. Each player plays a game against each other player, and either wins or loses the game (let's assume that there are no draws). We can visually represent a tournament by drawing a circle for each player and drawing arrows between pairs of players to indicate who won each game. For example, in the tournament to the left, player A beat player E, but lost to players B, C, and D.

A ***tournament winner*** is a player in a tournament who, for each other player, either won her game against that player, or won a game against a player who in turn won his game against that player (or both). For example, in the graph on the left, players B, C, and E are tournament winners. However, player D is ***not*** a tournament winner, because he neither beat player C, nor beat anyone who in turn beat player C. Although player D won against player E, who in turn won against player B, who then won against player C, under our definition player D is ***not*** a tournament winner. *(Make sure you understand why!)*

One of the most fundamental results about tournaments is the following: any tournament with at least one player has a tournament winner. The first part of this problem asks you to prove this.

i. Let T be an arbitrary tournament and p be any player in that tournament. Prove the following statement: if p won more games than anyone else in T or is tied for winning the greatest number of games, then p is a tournament winner in T. This shows that every tournament with at least one player has at least one winner.

ii. A ***pseudotournament*** is like a tournament, except that exactly one pair of people don't play each other. Show that for any $n \geq 2$, there's a pseudotournament P with n players and no tournament winners.

Extra Credit Problem: Logic and Language (1 Point Extra Credit)

A while back, someone posted a blog entry where they [translated several famous quotes into first-order logic](). (The syntax they use is a little bit different than what we're using, and they reference some concepts we haven't covered yet, but it's still quite a fun read.) Find two quotes, one that can nicely be translated into first-order logic and one that, for some reason, can't be easily translated into first-order logic. Provide a translation of the first statement, briefly explain why it's accurate, and explain why the second one can't easily be translated.