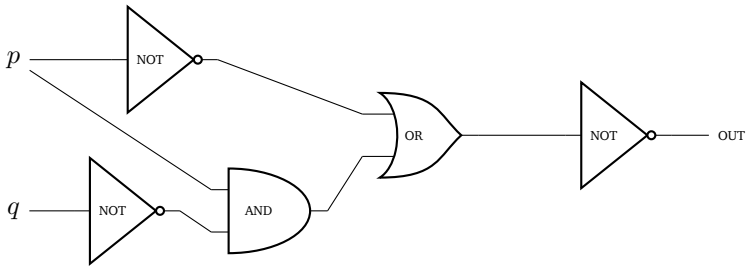
Section 02: Solutions

1. Circuitous

Translate the following circuit into a logical expression.



Solution:

$\neg(\neg p \vee (p \wedge \neg q))$

2. Equivalences

Prove that each of the following pairs of propositional formulae are equivalent using propositional equivalences.

(a) $p \leftrightarrow q \equiv (p \wedge q) \vee (\neg p \wedge \neg q)$ You may use the rule $p \leftrightarrow q \equiv (p \to q) \wedge (q \to p)$. **Solution:**

Note: Normally, we wouldn't include line numbers in an equivalence proof, but it makes it easier for us to discuss the intuition below.

$p \leftrightarrow q$	$\equiv$	$(p \to q) \wedge (q \to p)$	[iff is two implications]	(1)
	$\equiv$	$(\neg p \vee q) \wedge (q \to p)$	[Law of Implication]	(2)
	$\equiv$	$(\neg p \vee q) \wedge (\neg q \vee p)$	[Law of Implication]	(3)
	$\equiv$	$((\neg p \vee q) \wedge \neg q) \vee ((\neg p \vee q) \wedge p)$	[Distributivity]	(4)
	$\equiv$	$(\neg q \wedge (\neg p \vee q)) \vee ((\neg p \vee q) \wedge p)$	[Commutativity]	(5)
	$\equiv$	$((\neg q \wedge \neg p) \vee (\neg q \wedge q)) \vee ((\neg p \vee q) \wedge p)$	[Distributivity]	(6)
	$\equiv$	$((\neg q \wedge \neg p) \vee (q \wedge \neg q)) \vee ((\neg p \vee q) \wedge p)$	[Commutativity]	(7)
	$\equiv$	$((\neg q \wedge \neg p) \vee F) \vee ((\neg p \vee q) \wedge p)$	[Negation]	(8)
	$\equiv$	$(\neg q \wedge \neg p) \vee ((\neg p \vee q) \wedge p)$	[Identity]	(9)
	$\equiv$	$(\neg p \wedge \neg q) \vee ((\neg p \vee q) \wedge p)$	[Commutativity]	(10)
	$\equiv$	$(\neg p \wedge \neg q) \vee (p \wedge (\neg p \vee q))$	[Commutativity]	(11)
	$\equiv$	$(\neg p \wedge \neg q) \vee ((p \wedge \neg p) \vee (p \wedge q))$	[Distributivity]	(12)
	$\equiv$	$(\neg p \wedge \neg q) \vee (F \vee (p \wedge q))$	[Negation]	(13)
	$\equiv$	$(\neg p \wedge \neg q) \vee ((p \wedge q) \vee F)$	[Commutativity]	(14)
	$\equiv$	$(\neg p \wedge \neg q) \vee (p \wedge q)$	[Identity]	(15)
	$\equiv$	$(p \wedge q) \vee (\neg p \wedge \neg q)$	[Commutativity]	(16)
				(17)

Intuition: We're going to focus this explanation on the $(\neg p \wedge \neg q)$ side of this proof. The other side is similar. After the first two applications of the Law of Implication (Steps 2 and 3), we notice that $\neg p$ and $\neg q$ are in separate sub-expressions. We need to get them into the same sub-expression, so we use Distributivity (Step 4). Then in the left sub-expression we have $\neg p$ and $\neg q$, but we also have this extra q term that we don't want, so we have to get rid of it. The crux here is realizing that we can use Distributivity again to accomplish two things: 1. we can get $\neg p$ and $\neg q$ into their own sub-expression; 2. we can combine $\neg q$ and q to make that pesky q term disappear. So, we use Distributivity again (Step 6), and now we have $(\neg q \wedge \neg p)$, which is really close to what we want, and this $(\neg q \wedge q)$ term that we can get rid of by applying Commutativity (Step 7) and using Negation and Identity (Steps 8 and 9). And of course we have to apply Commutativity one more time (Step 10) to turn $(\neg q \wedge \neg p)$ into $(\neg p \wedge \neg q)$.

(b) $\neg p \to (q \to r) \equiv q \to (p \vee r)$ **Solution:**

$$
\begin{array}{lll}
\neg p \to (q \to r) & \equiv \neg\neg p \vee (q \to r) & \text{[Law of Implication]} \\
& \equiv p \vee (q \to r) & \text{[Double Negation]} \\
& \equiv p \vee (\neg q \vee r) & \text{[Law of Implication]} \\
& \equiv (p \vee \neg q) \vee r & \text{[Associativity]} \\
& \equiv (\neg q \vee p) \vee r & \text{[Commutativity]} \\
& \equiv \neg q \vee (p \vee r) & \text{[Associativity]} \\
& \equiv q \to (p \vee r) & \text{[Law of Implication]}
\end{array}
$$

3. Boolean Algebra

For each of the following parts, write the logical expression using boolean algebra operators. Then, simplify it using axioms and theorems of boolean algebra.

(a) $\neg p \vee (\neg q \vee (p \wedge q))$ **Solution:**

First, we replace $\neg, \vee$, and $\wedge$ with their corresponding boolean operators, giving us $p' + q' + pq$; We can use DeMorgan's Law to get $(pq)' + pq$. Then, we can use commutativity to get $pq + (pq)'$ and complementarity to get 1. (Note that this is another way of saying the formula is a tautology.)

(b) $\neg(p \vee (q \wedge p))$ **Solution:**

First, we replace $\neg, \vee$, and $\wedge$ with their corresponding boolean operators, giving us $(p + (qp))'$. Applying DeMorgan's Law once gives us $p'(qp)'$, and a second time gives us $p'(q' + p')$, which is $p'(p' + q')$ by commutativity. By absorption, this is p'.

4. Canonical Forms

Consider the boolean functions $F(A, B, C)$ and $G(A, B, C)$ specified by the following truth table:

A	B	C	$F(A,B,C)$	$G(A,B,C)$
1	1	1	1	0
1	1	0	1	1
1	0	1	0	0
1	0	0	0	0
0	1	1	1	1
0	1	0	1	0
0	0	1	0	1
0	0	0	1	0

(a) Write the DNF and CNF expressions for $F(A, B, C)$. **Solution:**

DNF: $ABC + ABC' + A'BC + A'BC' + A'B'C'$
CNF: $(A' + B + C')(A' + B + C)(A + B + C')$

(b) Write the DNF and CNF expressions for $G(A, B, C)$. **Solution:**

DNF: $ABC' + A'BC + A'B'C$
CNF: $(A' + B' + C')(A' + B + C')(A' + B + C)(A + B' + C)(A + B + C)$

5. ctrl-z

Translate these logical expressions to English. For each of the translations, assume that domain restriction is being used and take that into account in your English versions.

Let your domain be all UW Students. Predicates `143Student`(x) and `311Student`(x) mean the student is in CSE 143 and 311, respectively. `BioMajor`(x) means x is a bio major, `DidHomeworkOne(x)` means the student did homework 1 (of 311). Finally `KnowsJava`(x) and `KnowsDeMorgan`(x) mean x knows Java and knows DeMorgan's Laws, respectively.

(a) $\forall x($`143Student`$(x) \rightarrow$ `KnowsJava`$(x))$ **Solution:**

Every 143 student knows java.

"If a UW student is a 143 student, then they know java" is a valid translation of the original sentence, but it is not taking advantage of the domain restriction.

(b) $\exists x($`143Student`$(x) \wedge$ `BioMajor`$(x))$ **Solution:**

There is a 143 student who is a bio major.

"There is a UW student who is a 143 student and is a bio major" is a valid translation of the original sentence, but is not taking advantage of the domain restriction.

(c) $\forall x([$`311Student`$(x) \wedge$ `DidHomeworkOne(x)`$] \rightarrow$ `KnowsDeMorgan`$(x))$ **Solution:**

Every 311 student who did homework one knows DeMorgan's Laws.

"If a UW student is a 311 student and did homework 1, then they know DeMorgan's Laws" is a valid translation of the original sentence, but it is not taking advantage of the domain restriction.

6. Domain Restriction

Translate each of the following sentences into logical notation. These translations require some of our quantifier tricks. You may use the operators $+$ and $\cdot$ which take two numbers as input and evaluate to their sum or product, respectively. Remember:

- To restrict the domain under a $\forall$ quantifier, add a hypothesis to an implication.
- To restrict the domain under an $\exists$ quantifier, AND in the restriction.
- If you want variables to be different, you have to explicitly require them to be not equal.

(a) Domain: Positive integers; Predicates: $\texttt{Even}, \texttt{Prime}, \texttt{Equal}$
"There is only one positive integer that is prime and even." **Solution:**

$\exists x(\texttt{Prime}(x) \land \texttt{Even}(x) \land \forall y[\neg \texttt{Equal}(x, y) \rightarrow \neg(\texttt{Even}(y) \land \texttt{Prime}(y))])$

(b) Domain: Real numbers; Predicates: $\texttt{Even}, \texttt{Prime}, \texttt{Equal}$
"There are two different prime numbers that sum to an even number." **Solution:**

$\exists x \exists y(\texttt{Prime}(x) \land \texttt{Prime}(y) \land \neg Equal(x, y) \land \texttt{Even}(x + y))$

(c) Domain: Real numbers; Predicates: $\texttt{Even}, \texttt{Prime}, \texttt{Equal}$
"The product of two distinct prime numbers is not prime." **Solution:**

$\forall x \forall y([\texttt{Prime}(x) \land \texttt{Prime}(y) \land \neg \texttt{Equal}(x, y)] \rightarrow \neg \texttt{Prime}(xy))$

(d) Domain: Real numbers; Predicates: $\texttt{Even}, \texttt{Prime}, \texttt{Equal}, \texttt{Postivite}, \texttt{Greater}, \texttt{Integer}$
"For every positive integer, there is a greater even integer" **Solution:**

$\forall x(\texttt{Positive}(x) \land \texttt{Integer}(x) \rightarrow [\exists y(\texttt{Integer}(y) \land \texttt{Even}(y) \land \texttt{Greater}(y, x))])$
Or equivalently: $\forall x \exists y(\texttt{Positive}(x) \land \texttt{Integer}(x) \rightarrow (\texttt{Integer}(y) \land \texttt{Even}(y) \land \texttt{Greater}(y, x)))$ **Note:** Pulling quantifiers out to the front of a statement can sometimes be valid. It can also get tricky. Think about what the two statements mean, and if they don't quite seem equivalent or you're just not sure, don't do it.

7. There Exists an Implication

Implications are uncommon under existential quantifiers. Consider this expression (which we'll call "the original expression"): $\exists x(\texttt{P}(x) \rightarrow \texttt{Q}(x))$

(a) Suppose that $\texttt{P}(x)$ is not always true (i.e. there is an element in the domain for which $\texttt{P}(x)$ is false). Explain why the original expression is true in this case. (1-2 sentences should suffice.).

Solution:

Consider some y such that $\mathtt{P}(y)$ is false. Plugging this y in for x we get a true implication (by vacuous truth). Thus, this y is the y that must exist to make the quantified statement true.

(b) Suppose that $\mathtt{P}(x)$ is always true (i.e. $\forall x\, \mathtt{P}(x)$). There is a simpler statement which conveys the meaning of the original expression (i.e. is equivalent to it for all domains and predicates. By simpler, we mean "uses fewer symbols"). Give that expression, and briefly (1-2 sentences) explain why it works.

Solution:

$\exists x\, \mathtt{Q}(x)$. Since $\mathtt{P}(x)$ is always true, the hypothesis is always satisfied, so the implication is equivalent to the conclusion.

8. Quantifier Switch

Consider the following pairs of sentences. For each pair, determine if one implies the other, if they are equivalent, or neither.

(a) $\forall x\ \forall y\ P(x, y)$ $\qquad$ $\forall y\ \forall x\ P(x, y)$ **Solution:**

These sentences are the equivalent; switching the order of universal quantifiers makes no difference.

(b) $\exists x\ \exists y\ P(x, y)$ $\qquad$ $\exists y\ \exists x\ P(x, y)$ **Solution:**

These sentences are the equivalent; switching the order of existential quantifiers makes no difference.

(c) $\forall x\ \exists y\ P(x, y)$ $\qquad$ $\forall y\ \exists x\ P(x, y)$ **Solution:**

It depends. These are only equivalent if P is symmetric (i.e., the order of the arguments doesn't matter). If the order of the arguments does matter, then these are different statements. For instance, if $P(x, y)$ is "$x < y$", and our domain of discourse is "positive integers", then the first statement says "for every x, there is a corresponding y such that $x < y$" (i.e., every positive integer has a larger positive integer), which is true. The second says "for every y, there is a corresponding x such that $x < y$" (i.e., every positive integer has a smaller positive integer), which is not true. Note that in the first statement y depends on x, and in the second x depends on y.

(d) $\forall x\ \exists y\ P(x, y)$ $\qquad$ $\exists x\ \forall y\ P(x, y)$ **Solution:**

These two statements are different. Consider our example from part (c) ($P(x, y) := x < y$, and the domain of discourse is positive integers). $\forall x\ \exists y\ P(x, y)$ says "for every positive integer x, there is a positive integer y such that $x < y$" (i.e., every positive integer has a larger positive integer). $\exists x\ \forall y\ P(x, y)$ says "there is a positive integer x such that for all positive integers y, $x < y$" (i.e., there is a smallest positive integer). In this example, both of our quantified statements happen to be true, but they have very different meanings.

(e) $\forall x\ \exists y\ P(x, y)$ $\qquad$ $\exists y\ \forall x\ P(x, y)$ **Solution:**

The second statement is "stronger" than the first (that is, the second implies the first). For the first, y is allowed to depend on x. For the second, one specific y must work for all x. Thus if the second is true,

whatever value of y makes it true, can also be plugged in for y in the first statement for every x. On the other hand, if the first statement is true, it might be that different y's work for the different x's and no single value of y exists to make the latter true.

As an example, let your domain of discourse be positive real numbers, and let $P(x, y)$ be $xy = 1$. The first statement is true (always take y to be $1/x$, which is another positive real number). The second statement is not true; it asks for a single number y whose product with every real number x is 1.

9. Domain Restriction Negated

When we negate a sentence with a domain restriction, the restriction itself remains intact (i.e. not negated) after we have fully simplified. That should make sense; if I claim something is true for every even number, I won't be convinced by you showing me that that thing is false for odd numbers. In this problem, you'll do the algebra to see why.

(a) Consider the statement $\neg\exists x\exists y(\texttt{Domain1}(x) \wedge \texttt{Domain2}(y) \wedge [P(x, y) \wedge Q(x, y)])$. We know that we should end up with $\forall x\forall y(\texttt{Domain1}(x) \wedge \texttt{Domain2}(y) \rightarrow [\neg P(x, y) \vee \neg Q(x, y)])$ (that is flip the quantifiers, rewrite the domain restriction, and negate the other requirements).

But it can help to see the full algebra written out – write out a step-by-step simplification to get the simplified form (you don't have to label with rules). **Solution:**

$$\begin{aligned}
\neg\exists x\exists y(\texttt{Domain1}(x) \wedge \texttt{Domain2}(y) \wedge [P(x, y) \wedge Q(x, y)]) &\equiv \forall x\neg[\exists y(\texttt{Domain1}(x) \wedge \texttt{Domain2}(y) \wedge [P(x, y) \wedge Q(x, y)])] \\
&\equiv \forall x\forall y\neg[(\texttt{Domain1}(x) \wedge \texttt{Domain2}(y) \wedge [P(x, y) \wedge Q(x, y)])] \\
&\equiv \forall x\forall y(\neg\,\texttt{Domain1}(x) \vee \neg\,\texttt{Domain2}(y) \vee \neg[P(x, y) \wedge Q(x, y)]) \\
&\equiv \forall x\forall y(\neg\,\texttt{Domain1}(x) \vee \neg\,\texttt{Domain2}(y) \vee [\neg P(x, y) \vee \neg Q(x, y)]) \\
&\equiv \forall x\forall y([\neg\,\texttt{Domain1}(x) \vee \neg\,\texttt{Domain2}(y)] \vee [\neg P(x, y) \vee \neg Q(x, y)]) \\
&\equiv \forall x\forall y(\neg[\texttt{Domain1}(x) \wedge \texttt{Domain2}(y)] \vee [\neg P(x, y) \vee \neg Q(x, y)]) \\
&\equiv \forall x\forall y([\texttt{Domain1}(x) \wedge \texttt{Domain2}(y)] \rightarrow [\neg P(x, y) \vee \neg Q(x, y)])
\end{aligned}$$

(b) Now do the same process for: $\neg\forall x\forall y([\texttt{Domain1}(x) \wedge \texttt{Domain2}(y)] \rightarrow [P(x, y) \wedge Q(x, y)])$ **Solution:**

$$\begin{aligned}
\neg\forall x\forall y([\texttt{Domain1}(x) \wedge \texttt{Domain2}(y)] \rightarrow [P(x, y) \wedge Q(x, y)]) &\equiv \exists x\neg[\forall y([\texttt{Domain1}(x) \wedge \texttt{Domain2}(y)] \rightarrow [P(x, y) \wedge Q(x, y)])] \\
&\equiv \exists x\exists y\neg[([\texttt{Domain1}(x) \wedge \texttt{Domain2}(y)] \rightarrow [P(x, y) \wedge Q(x, y)])] \\
&\equiv \exists x\exists y\neg[(\neg[\texttt{Domain1}(x) \wedge \texttt{Domain2}(y)] \vee [P(x, y) \wedge Q(x, y)])] \\
&\equiv \exists x\exists y(\neg\neg[\texttt{Domain1}(x) \wedge \texttt{Domain2}(y)] \wedge \neg[P(x, y) \wedge Q(x, y)]) \\
&\equiv \exists x\exists y(\texttt{Domain1}(x) \wedge \texttt{Domain2}(y) \wedge \neg[P(x, y) \wedge Q(x, y)]) \\
&\equiv \exists x\exists y(\texttt{Domain1}(x) \wedge \texttt{Domain2}(y) \wedge [\neg P(x, y) \vee \neg Q(x, y)])
\end{aligned}$$

10. Quantifier Ordering

Let your domain of discourse be a set of `Element` objects given in a list called `Domain`. Imagine you have a predicate $\texttt{pred}(x, y)$, which is encoded in the java method `public boolean pred(int x, int y)`. That is you call your predicate `pred` true if and only if the java method returns `true`.

(a) Consider the following Java method:

```
public boolean Mystery(Domain D){
    for(Element x : D) {
```

```
        for(Element y : D) {
            if(pred(x,y))
                return true;
        }
    }
            return false;
}
```

`Mystery` corresponds to a quantified formula (for `D` being the domain of discourse), what is that formula?
Solution:

$\exists x \exists y(\texttt{pred}(x, y))$. If any combination of x and y causes pred to evaluate to true, we return true; that is we just want x, y to exist.

(b) What formula does `mystery2` correspond to

```
public boolean Mystery2(Domain D){
    for(Element x : D) {
        boolean thisXPass = false;
        for(Element y : D) {
            if(pred(x,y))
                thisXPass = true;
        }
        if(!thisXPass)
            return false;
    }
    return true;
}
```

Solution:

$\forall x \exists y(\texttt{pred}(x, y))$.

For a given x, when we come across a y that makes $\texttt{pred}(x, y)$ true, we set the given x to pass (so one y suffices for a given x) but we require **every** x to pass, so x is universally quantified. Since y is allowed to depend on x, we have x as the outermost variable.