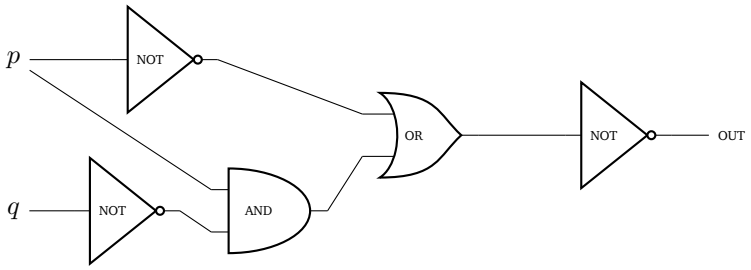
Section 02: Domain Restriction and Translations

1. Circuitous

Translate the following circuit into a logical expression.



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 $p \leftrightarrow q \equiv (p \rightarrow q) \wedge (q \rightarrow p)$.

(b) $\neg p \rightarrow (q \rightarrow r) \equiv q \rightarrow (p \vee r)$

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

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

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 $F(A, B, C)$.

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

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 $\texttt{143Student}(x)$ and $\texttt{311Student}(x)$ mean the student is in CSE 143 and 311, respectively. $\texttt{BioMajor}(x)$ means x is a bio major, `DidHomeworkOne(x)` means the student did homework 1 (of 311). Finally $\texttt{KnowsJava}(x)$ and $\texttt{KnowsDeMorgan}(x)$ mean x knows Java and knows DeMorgan's Laws, respectively.

(a) $\forall x(\texttt{143Student}(x) \rightarrow \texttt{KnowsJava}(x))$

(b) $\exists x(\texttt{143Student}(x) \wedge \texttt{BioMajor}(x))$

(c) $\forall x([\texttt{311Student}(x) \wedge \texttt{DidHomeworkOne(x)}] \rightarrow \texttt{KnowsDeMorgan}(x))$

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: `Even`, `Prime`, `Equal`
"There is only one positive integer that is prime and even."

(b) Domain: Real numbers; Predicates: `Even`, `Prime`, `Equal`
"There are two different prime numbers that sum to an even number."

(c) Domain: Real numbers; Predicates: `Even`, `Prime`, `Equal`
"The product of two distinct prime numbers is not prime."

(d) Domain: Real numbers; Predicates: `Even`, `Prime`, `Equal`, `Postivite`, `Greater`, `Integer`
"For every positive integer, there is a greater even integer"

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.).

(b) Suppose that $\texttt{P}(x)$ is always true (i.e. $\forall x\, \texttt{P}(x)$). 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.

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) $\forall x\ \forall y\ P(x,y)$ $\qquad$ $\forall y\ \forall x\ P(x,y)$

(b) $\exists x\ \exists y\ P(x,y)$ $\qquad$ $\exists y\ \exists x\ P(x,y)$

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

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

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

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).

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

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?

(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;
}
```