Propositional and Predicate Logic
CS 536: Science of Programming, Spring 2018


A. Why
• Reviewing/overviewing logic is necessary because we’ll be using it in the course.
• We’ll be using predicates to write specifications for programs.
• Predicates and programs have meaning relative to states.

B. Outcomes
At the end of this lecture, you should
• Understand what a propositional formula is, how to write them, how to tell whether one is a tautology or contradiction using truth tables, and see a basic set of logical rules for transforming propositions.

C. In Case You Missed Class Monday
• The course webpages are at http://cs.iit.edu/~cs536/. Read it carefully for policies and refer to it often to download lecture notes and homework assignments. Check myIIT → Blackboard for lecture videos. Policy questions: Can you work with others on homework? How can you recover from a low Exam 1 score?
• Review the material: Propositions are built up from proposition variables, proposition constants, and the connectives ∧, ∨, →, ↔, and ¬. Can add/remove parentheses using precedence and associativity rules. Truth table semantics; tautologies, contradictions, contingencies.

D. Propositional Logic
• Last time: proposition variables, the connectives ∧, ∨, →, ↔, and ¬. Their precedences and associativies. Truth table semantics.

E. Truth vs Provability
• In addition to semantic truth based on truth tables, there is also a notion of “provable truth” based on syntactic manipulation of propositions. E.g., “if p ∧ q is provable then q ∧ p is provable” or “q ∧ p follows from p ∧ q”.
• Given a set of proof rules, two propositions are provably equivalent if each follows from the other according to those rules. E.g., p ∧ q and q ∧ p are provably equivalent.
• Two propositions are logically equivalent if they have the same truth table. E.g., p ∧ q and q ∧ p are logically equivalent. Note p₁ and p₂ are logically equivalent iff (p₁ ↔ p₂) is a tautology.
• Proofs of propositional equivalence often form a chain: p₁ is equivalent to p₂, which is equivalent to p₃ etc. But “p₁, p₂, and p₃ are logically equivalent” is different from p₁ ↔ (p₂ ↔ p₃).
• To indicate logical equivalence, we’ll use ⇔ (also common is “iff”): p₁ ⇔ p₂ ⇔ p₃ means (p₁ ⇔ p₂) and (p₂ ⇔ p₃) (and therefore p₁ ⇔ p₃).
• Similar to logical equivalence, there is a notion of logical implication that is often chained together, and this is different from →. We’ll use “⇒” (pronounced “implies”).
• $p_1 \Rightarrow p_2 \Rightarrow p_3$ means $(p_1 \Rightarrow p_2)$ and $(p_2 \Rightarrow p_3)$ (and therefore $p_1 \Rightarrow p_3$).

**Propositional Logic Rules**

You don't need to memorize these rules by name, but you should be able to give the name of a rule. For example, “$(p \rightarrow q) \land (q \rightarrow r) \Rightarrow (p \rightarrow r)$ is __________.”  (Answer: transitivity)

### Commutativity [8/24 minor format changes]

- $p \lor q \Leftrightarrow q \lor p$
- $p \land q \Leftrightarrow q \land p$
- $(p \leftrightarrow q) \Leftrightarrow (q \leftrightarrow p)$

### Distributivity/Factoring

- $(p \lor q) \land r \Leftrightarrow (p \land r) \lor (q \land r)$
- $(p \land q) \lor r \Leftrightarrow (p \lor r) \land (q \lor r)$

### Transitivity [Note: \(\Rightarrow\), not \(\Leftrightarrow\) here]

- $(p \rightarrow q) \land (q \rightarrow r) \Rightarrow (p \rightarrow r)$
- $(p \leftrightarrow q) \land (q \leftrightarrow r) \Rightarrow (p \leftrightarrow r)$

### Identity:

$p \land T \Leftrightarrow p$ and $p \lor F \Leftrightarrow p$

### Idempotency:

$p \lor p \Leftrightarrow p$ and $p \land p \Leftrightarrow p$

### Domination:

$p \lor T \Leftrightarrow T$ and $p \land F \Leftrightarrow F$

### Absurdity:

$(F \rightarrow p) \Leftrightarrow T$

### Contradiction:

$p \land \neg p \Leftrightarrow F$

### Excluded middle:

$p \lor \neg p \Leftrightarrow T$

### Double negation$^\dagger$:

$\neg \neg p \Leftrightarrow p$

### DeMorgan’s Laws

- $\neg(p \land q) \Leftrightarrow (\neg p \lor \neg q)$
- $\neg(p \lor q) \Leftrightarrow (\neg p \land \neg q)$

### Negation of Comparisons: added [9/21]

- $\neg(e_1 \leq e_2) \Leftrightarrow e_1 > e_2$ (similar for $<$, $>$, $\geq$, $=$, $\neq$)

**Substitution:** Given $p$, $q$, and $r$, let $r'$ be the result of substituting a $q$ for one or more occurrences of $p$ inside $r$. If $p \Leftrightarrow q$ then $r \Leftrightarrow r'$. Example: Since $(p \rightarrow q) \Leftrightarrow (\neg p \lor q)$, we know $(p \rightarrow q) \land s \Leftrightarrow (\neg p \lor q) \land s$.

### F. Sample Proofs

- Here is a proof of $\neg(p \rightarrow q) \Leftrightarrow (p \land \neg q)$ (also known as “negation of $\rightarrow$”).

  $\neg(p \rightarrow q)$
  \[\Leftrightarrow \neg(\neg p \lor q)\]  
  Defn $\rightarrow$
  \[\Leftrightarrow \neg\neg p \land \neg q\]  
  DeMorgan’s Law
  \[\Leftrightarrow p \land \neg q\]  
  Pierce’s Law (double negation)

- Here is a proof of  $((r \rightarrow s) \land r) \rightarrow s \Leftrightarrow T$ (“Modus ponens”).

  $(r \rightarrow s) \land r \rightarrow s$
  \[\Leftrightarrow \neg((r \rightarrow s) \land r) \lor s\]  
  Defn of $\rightarrow$

$^\ast$ You don't need to memorize these rules by name, but you should be able to give the name of the rule given its body. I.e., “$(p \rightarrow q) \land (q \rightarrow r) \Rightarrow (p \rightarrow r)$ is the _______ rule”.

$^\dagger$ Also known as Pierce’s Law
\[ (\neg(r \rightarrow s) \lor \neg r) \lor s \quad \text{DeMorgan’s Law} \]
\[ \equiv (r \land \neg s) \lor \neg r \lor s \quad \text{Negation of } \rightarrow \text{ [see above]} \]
\[ \equiv (r \lor \neg r) \land (\neg s \lor \neg r) \lor s \quad \text{Distribute } \lor \text{ over } \land \]
\[ \equiv (T \land (\neg s \lor \neg r)) \lor s \quad \text{Excluded middle} \]
\[ \equiv (\neg s \lor \neg r) \lor s \quad \text{Identity} \]
\[ \equiv T \lor \neg r \quad \text{Excluded middle (see below)} \]
\[ \equiv T \quad \text{Domination} \]

**Avoid Unpleasant Levels of Detail [8/24]**

In the proof above, if we’re being very technical, the use of excluded middle to go from \((\neg s \lor \neg r) \lor s\) to \(T \lor \neg r\) actually uses
\[ (\neg s \lor \neg r) \lor s \]
\[ \equiv \neg s \lor (\neg r \lor s) \quad \text{Associativity of } \lor \]
\[ \equiv \neg s \lor (s \lor \neg r) \quad \text{Commutativity of } \lor \]
\[ \equiv (\neg s \lor s) \lor \neg r \quad \text{Associativity of } \lor \]
\[ \equiv T \lor \neg r \quad \text{Excluded middle} \]

[8/24] And if we’re being even pickier, we should go from \((\neg s \lor s) \lor \neg r\) to \((s \lor \neg s) \lor \neg r\), since the rule for excluded middle says \(p \lor \neg p \equiv T\), not \(\neg p \lor p \equiv T\). To avoid this level of detail, let’s agree that associativity and commutativity can be used without mentioning them specifically.

**G. Derived Rules**

- If \(p \iff q\) is a tautology (i.e., \((p \iff q) \equiv T\)), then we can use \(p \iff q\) as a **derived rule**. E.g., to prove modus ponens, we showed \((r \rightarrow s) \land r \rightarrow s \equiv T\). Here’s an example of using modus ponens (for \(r\) we substitute \(p \land q\); for \(s\) we substitute \(r\); sorry if it’s confusing).

\[ ((p \land q) \rightarrow r) \land (p \land q) \rightarrow r \]
\[ \equiv T \quad \text{Modus ponens} \]

- Similarly, if \(p \rightarrow q\) is a tautology then \(p \Rightarrow q\) can be a derived rule. E.g., proving \((p \land q \rightarrow p) \equiv T\) lets us use \(p \land q \Rightarrow p\) as a rule (sometimes called “and-elimination”).

\[ p \land q \rightarrow p \]
\[ \equiv \neg(p \land q) \lor p \quad \text{Defn } \rightarrow \]
\[ \equiv (\neg p \lor \neg q) \lor p \quad \text{DeMorgan’s Law} \]
\[ \equiv T \lor \neg q \quad \text{Excluded middle} \]
\[ \equiv T \quad \text{Domination} \]

- Some other common derived rules: [you don’t have to memorize these]
  - **contraposition**: \((p \rightarrow q) \equiv (\neg q \rightarrow \neg p)\)
  - **and-introduction**: \(p \rightarrow (q \rightarrow r) \equiv p \land q \rightarrow r\)
  - **or-introduction**: \(p \Rightarrow p \lor q\)
  - **or-elimination**: \((p \rightarrow r) \land (q \rightarrow r) \land (p \lor q) \Rightarrow r\)
  - **not-introduction**: \((p \rightarrow F) \equiv \neg p\)
H. Predicate Logic

- In propositional logic, we assert truths about boolean values; in predicate logic, we assert truths about values from one or more “domains of discourse” like the integers.
- We extend propositional logic with domains (sets of values), variables whose values range over these domains, and operations on values (e.g. addition). E.g., for the integers we add the set \( \mathbb{Z} \), operations +, −, *, /, \% (mod), and relations =, ≠, <, >, ≤, and ≥. We’ll also add arrays of integers and array indexing, as in \( b[0] \).
- A predicate is a logical assertion that describes some property of values.
- To describe properties involving values, we add basic relations on values (e.g., less-than). We also have rules over these relations, like \( x \ast 0 = 0 \) being a rule of arithmetic.

Quantifiers

- When a predicate includes a variable, we have to ask for what values of the variable we think the predicate might be true: Some current value? Every value? Some value?
- We use quantifiers to specify all values, some value, and exactly one value.

Universal Quantification

- A universally quantified predicate (or just “universal” for short) has the form \((\forall x \in S . p)\) where \( S \) is a set and \( p \) (the body of the universal) is a predicate involving \( x \). E.g., every natural number greater than 1 is less than its own square: \((\forall x \in \mathbb{N} . x > 1 \rightarrow x < x^2)\).‡
- Often we leave out the set if it is understood. E.g., \((\forall x . x > 1 \rightarrow x < x^2)\).

Existential Quantification

- An existentially quantified predicate (or just “existential” for short) has the form \((\exists x \in S . p)\) where \( S \) is a set and \( p \) (the body of the existential) is a predicate involving \( x \). E.g., there is a nonzero integer that equals its own square: \((\exists x \in \mathbb{Z} . x \neq 0 \land x = x^2)\).
- Usually the set is understood to be \( \mathbb{Z} \) and we leave out. E.g., \((\exists x . x \neq 0 \land x = x^2)\).

Syntactic Equality of Quantified Predicates

- For syntactic equality, which variable you quantify over will make a difference for us, so we'll treat: \((\forall x . x > x-1) \neq (\forall y . y > y-1)\). These will be logically equivalent, however: \((\forall x . x > x-1) \Leftrightarrow (\forall y . y > y-1)\).

Bounded Quantifiers

- For bounded quantifiers, we write \( Q p . q \) instead of \( Q x \ op r \) (where \( Q \) is \( \forall \) and \( op \) is \( \rightarrow \) or \( Q \) is \( \exists \) and \( op \) is \( \land \)). More specifically,
  - \( \forall p . q \) means \( \forall x . p \rightarrow q \) where \( x \) appears in \( p \) and \( x \) is understood to be the variable we are quantifying over.
  - Example 1: \((\forall x . x > x^2) \equiv (\forall x . x > 1 \rightarrow x < x^2)\)
  - \( \exists p . q \) means \( \exists x . p \land q \) where \( x \) appears in \( p \) and \( x \) is understood to be the variable we are quantifying over. (Note: it's \( p \land q \) here; compare with \( p \rightarrow q \) for bounded universals.)

‡ For us, 0 \( \in \mathbb{N} \); if you know a source that says otherwise, please let me know.
**Example 2:** $(\exists x \neq 0 . \ x = x^2) \equiv (\exists x . \ x \neq 0 \land x = x^2)$

**[8/24]** It’s important to use the right connective ($\rightarrow$ for bounded $\forall$ and $\land$ for bounded $\exists$). For example, there’s a big difference between $(\exists x \in \mathbb{Z} . \ x > 1 \land x = x^2)$ and $(\exists x \in \mathbb{Z} . \ x > 1 \rightarrow x = x^2)$; the first one is false, the second one is true (any $x \leq 0$ makes the implication true).

### Parentheses For Quantified Predicates

- We’ll treat $\forall$ and $\exists$ as having low precedence. (Note: Some people use high precedence). So the body of a quantified predicate is as long as possible.
- **Example 3.** $\forall x \in \mathbb{N} . \ x > 1 \rightarrow x < x^2$ means $(\forall x \in \mathbb{N} . ((x > 1) \rightarrow (x < x^2)))$.
- **Example 4.** $\forall x \in \mathbb{Z} . \exists y \in \mathbb{Z} . \ y \leq x^2$ means $(\forall x \in \mathbb{Z} . (\exists y \in \mathbb{Z} . (y \leq x^2)))$.
- If we have $(\ldots . Q x . \ldots )$ where the two parentheses shown match, then the body can’t extend past the right parenthesis, and we get $(\ldots Q x . (\ldots ))$.

**Example 5:** $(\exists y \in \mathbb{Z} . \ y > 0 \land x > y) \rightarrow x \geq 1$

$$\equiv ((\exists y \in \mathbb{Z} . ((y > 0) \land (x > y))) \rightarrow (x \geq 1))$$

- For full parenthesizations, we add parentheses around basic tests, but we still omit them around variables and constants. Let’s also omit them around array indexes, so we’ll write $b[(x+1)] > y$, not $(b[(x+1)]) > y$. [It’s still okay to write $b[(x+1)] > y$ for $b[(x+1)] > y$.]

**Example 6:** $x > 0 \land y \leq 0$ expands to $((x > 0) \land (y \leq 0))$.

### DeMorgan’s Laws Extended

- For quantified predicates, there are two more **DeMorgan’s Laws**:
  - $(\neg \forall x . p) \equiv (\exists x . \neg p)$ and $(\neg \exists x . p) \equiv (\forall x . \neg p)$
- With bounded quantifiers, because of how $\rightarrow$, $\neg$, and $\land$ are related,
  - $(\neg \forall p . q) \equiv (\exists p . \neg q)$. I.e., $(\neg \forall x . p \rightarrow q) \equiv (\exists x . \neg(p \rightarrow q)) \equiv (\exists x . p \land \neg q) \equiv (\exists p . \neg q)$.
  - $(\neg \exists p . q) \equiv (\forall p . \neg q)$ I.e., $(\neg \exists x . p \land q) \equiv (\forall x . \neg(p \land q))$
    $$\equiv (\forall x . \neg p \lor \neg q)) \equiv (\forall x . p \rightarrow \neg q)) \equiv (\forall p . \neg q).$$
- **Example 7:** $\neg(\forall x . \ x > 0) \equiv (\exists x . \neg(x > 0)) \equiv (\exists x . \ x \leq 0)$
- **Example 8:** $\neg(\forall x > 0 . \ x^2 = x) \equiv (\exists x > 0 . \ x^2 \neq x)$
- **Example 9:** $\neg(\exists x . \ x \leq 0 \land x > 0) \equiv (\forall x . \neg(x \leq 0 \land x > 0)) \equiv (\forall x . \ x > 0 \lor x \leq 0))$.

### Proofs of Quantified Predicates

- Formal systems for proving predicates are pretty complicated; rather than study one of them, let’s rely on an informal idea of how to prove universally and existentially quantified predicates.
- In general, to prove $\forall x . p$, you prove $p$ but without imposing any restrictions on $x$. If you need to restrict $x$, then this needs to be part of the body of the quantified predicate.
  - **Example 10:** To prove $\forall x \in \mathbb{Z} . \ x \neq 0 \rightarrow x \leq x^2$, we can say “Let $x$ be an integer. Assume that $x$ isn’t zero. In that case, $x \leq x^2$.”
  - To prove $\exists x \in S . p$, you name a **witness value** for $x$ and prove $p$ holds if $x$ has that value.
Example 11: To prove $\exists x \in \mathbb{Z} : x \neq 0 \land x \geq x^2$, the only value that works as a witness is 1. More generally, there may be multiple witness values that work; we just need to name one.

- If a predicate includes unquantified variables, then for it to be a tautology, it has to hold for all possible values of those quantified variables. It’s a contradiction if it fails for all values, and it’s a contingency if it holds for some values but not some others. (I.e., unquantified variables are “implicitly universally quantified”.)

Example 12: $x > 0 \rightarrow \exists y : y^2 < x$ is a tautology because $\forall x : (x > 0 \rightarrow \exists y : y^2 < x)$ holds.

Example 13: $x > 0 \rightarrow y^2 < x$ is a contingency because it holds for some $x$ and $y$ (like $x = 2$ and $y = 1$) but fails for others (like $x = y = 1$).

Example 14: $\exists y : (y < 0 \land y > x^2)$ is a contradiction because it fails for every value of $x$.

Predicate Functions

- Often, we’ll give names to predicates and parameterize them. In programming languages, these predicate functions are written as functions that yield a boolean result.

Example 15: we might define Even$(x) \equiv (x \% 2) = 0$, where $\%$ is the remainder operator. E.g., Even$(3) \equiv (3 \% 2) = 0 \leftrightarrow 1 = 0 \leftrightarrow F$.

- In a programming language, the body of a predicate function can be a general program — one that uses loops and decisions. We want our predicates to be simpler than that: We’re going to use predicates to augment our programs with specifications, and it won’t help if debugging a predicate function body is exactly as hard as debugging a general program.

- So we’ll restrict ourselves to predicate functions that basically have the format Name(parameter variables) $\equiv$ predicate. The predicate can use the parameter variables and the built-in relations for our datatypes (for integers, $<$, $\leq$, etc.) along with the propositional connectives ($\land$, $\lor$, etc.)

Example 16: Let’s define IsZero$(b, m)$ to be true if the first $m$ elements of $b$ are all zero. To help, let’s assume size$(b)$ gives the number of elements in $b$.

- Rewriting, IsZero$(b, m)$ means that $b[0], b[1], ..., b[m-1]$ all $= 0$.

- It might be tempting to write IsZero$(b, m) \equiv b[0] = 0 \land b[1] = 0 \land ... \land b[m-1] = 0$

- But the right hand side is not a predicate; a predicate needs a fixed number of conjuncts being and’ed together.

- To write this as a predicate, we look for a pattern in our informal description: “$b[0], b[1], ..., b[m-1]$ all $= 0$” is equivalent to “$b[i] = 0$ for (every) $i = 0, 1, ..., m-1$”. The implied “every” $i$ tells us we need a universal quantifier $\forall$.

- So we can get IsZero$(b, m) \equiv \forall j. 0 \leq j < m \leq \text{size}(b) \rightarrow b[j] = 0$. (With bounded quantifiers, IsZero$(b, m) \equiv \forall 0 \leq j < m \leq \text{size}(b). b[j] = 0$.)

- Another way to look at a description and find an equivalent predicate is to imagine writing a loop to calculate whether the property is true or false.

- E.g., with “$b[0], b[1], ..., b[m-1]$ all $= 0$” we might imagine a loop

  for $i = 0$ to $m-1$
  
  if $b[i] \neq 0$ then return false
  
  return true
• The “for i = 0 to m-1” tells us we need to search for i in the range 0 ≤ i < m.
• The loop returns true only if all the b[i] pass the = 0 test; this tells us we need ∀ i.
• The general translation for a universal is ∀ loop var .((var in search range) → (test on var)). For this example, the search range is 0 ≤ i < m (plus an out of bounds test i < size(b)) and the test on i is b[i] = 0. This gives us ∀ i. 0 ≤ i < m ≤ size(b) → b[i] = 0.
• We need a ∃ search if our loop needs to return true as soon as it finds a b[i] that passes the test. E.g., if the property had been “At least one of b[0], b[1], ..., b[m-1] = 0”, we might imagine a loop
  for i = 0 to m-1
  if b[i] = 0 then return true
  return false
• For an existential we translate the loop to ∃ loop var .((var in search range) ∧ (test on var)). For this example, we get ∃ i. 0 ≤ i < m ≤ size(b) ∧ b[i] = 0.

• Example 17: Define SortedUp(b, m, n) so that it is true when array b is sorted ≤ on the segment m..n.
  As an example, if b[0..3] is [1, 3, 5, 2], then SortedUp(b, 0, 2) is true because 1 ≤ 3 and 3 ≤ 5 but
  SortedUp(b, 0, 3) is false because we don’t have (1 ≤ 3 and 3 ≤ 5 and 5 ≤ 2).
• Another way to describe SortedUp(b, m, n) is that each element in the list b[m], b[m+1], ..., b[n-2],
  b[n-1] is ≤ the element to its right.
  • Or expanding further, b[m] ≤ b[m+1], b[m+1] ≤ b[m+2], ..., b[n-1] ≤ b[n]. We can generalize this to
    b[i] ≤ b[i+1] for i = m, m+1, m+2, ..., n-1. To get a formal predicate, we’ll need a ∀ over i:
    • SortedUp(b, m, n) ≡ ∀ i. 0 ≤ m ≤ i < n < size(b) → b[i] ≤ b[i+1]
    • or, SortedUp(b, m, n) ≡ ∀ 0 ≤ m ≤ i < n < size(b). b[i] ≤ b[i+1].
  • We can hoist the parts of this that don’t depend on the quantified variable i:
    • SortedUp(b, m, n) ≡ 0 ≤ m < n < size(b) ∧ ∀ m ≤ i < n . b[i] ≤ b[i+1].
• Note: Different generalizations of a property can lead us to different predicates.
• For SortedUp, if we generalize (b[m] ≤ b[m+1], b[m+1] ≤ b[m+2], ..., b[n-1] ≤ b[n]) to (b[m+j] ≤
  b[m+j+1] for j = 0, 1, ..., n-1-m), we get ∀ 0 ≤ j < n-m . b[m+j] ≤ b[m+j+1] (and 0 ≤ m ≤ n <
  size(b)).
• Example 18: Let’s find a definition for Extends(b, b’) so that it’s true if b’ is an extension of b. I.e., b[0] =
  b’[0], b[1] = b’[1], ... for all elements of b. Note b’ can be the same length as b or can be longer. E.g., if
  b is [1, 6, 2] and b’ is [1, 6, 2, 8], then Extends(b, b’) is true and Extends(b’, b) is false. Here’s one
  solution:
  • Extends(b, b’) ≡ size(b) ≤ size(b’) ∧ ∀ 0 ≤ k < size(b) . b[k] = b’[k].
Propositional and Predicate Logic
CS 536: Science of Programming, Spring 2018

A. Why?
- Reviewing/overviewing logic is necessary because we’ll be using it in the course.
- We’ll be using predicates to write specifications for programs.

B. Objectives
At the end of this activity you should be able to:
- Follow and generate simple proofs of propositional formulas, given a set of rules.
- Read and write predicates and logically negate predicates.
- Translate informal descriptions of properties on integers and arrays into formal predicates and predicate functions.

C. Questions
1. Fill in the missing rule names in the proof below of \( \neg(p \iff q) \iff (q \land \neg p) \lor (p \land \neg q) \), using the rules from the lecture notes. (See page 2.)

\[
\neg(p \iff q) \\
\iff \neg((p \rightarrow q) \land (q \rightarrow p)) \\
\iff \neg(p \rightarrow q) \lor \neg(q \rightarrow p) \\
\iff (p \land \neg q) \lor (q \land \neg p) \\
\iff (q \land \neg p) \lor (p \land \neg q)
\]

2. Write a formal proof that shows that \( (p \rightarrow p \lor q) \) (sometimes called the “\( \lor \) introduction” rule) is a tautology:
Prove \( (p \rightarrow p \lor q) \iff T \)

\[
p \rightarrow p \lor q \\
\iff \text{__________} \\
\text{by _________}
\]
etc.

3. Some logical rules can be derived from others. Prove the rule of contraposition by proving
\( (p \rightarrow q) \rightarrow (\neg q \rightarrow \neg p) \iff T \), using only these rules: Definition of \( \rightarrow \), Pierce’s law, commutativity of \( \lor \), and excluded middle. (You may need to use a rule more than once.)

4. Let \( q(x, y) = x < y \rightarrow y < z \land f(x) = 2 \). Expand \( \neg q(x, y) \) to remove \( \neg \) signs: Use the rules to find a predicate equivalent to \( \neg(x < y \rightarrow y < z \land f(x) = 2) \) that doesn’t use \( \neg \). Hint: Use DeMorgan’s laws to move the negation “inward” to smaller and smaller subexpressions. Show your reasoning as a formal proof. (Don't forget the rule names.)

5. What are the minimal and full parenthesization for
a. \( (\forall x. (\exists y. x > y) \land (\exists y. x < y)) \)
b. \( \forall x. \neg(\exists y. p \land \forall z. q) \)
c. \( \forall x. \forall y. \exists z. (x \neq y \rightarrow x \leq z \land z \leq y \lor x > z \land z \geq y) \)
6. In general, if $\forall x \forall y. P(x, y)$ is valid, is $\forall y \forall x. P(x, y)$ also valid? What about $\exists x. \exists y. P(x, y)$ and $\exists y. \exists x. P(x, y)$?

7. Using propositional and predicate proof rules, find a predicate equivalent to $\neg(\forall x. \exists y. P(x, y))$ that has no negation symbols (i.e., $\neg$), except possibly in front of $P(x, y)$. Write a formal proof that shows each step needed (don't forget the rule names!). Hint: Use DeMorgan's laws to move the negation inward.

8. Repeat the previous question on $\neg(\exists y. \forall x. P(x, y))$.

9. Write the definition of a predicate function $\text{Repeats}(b, m)$ that is true exactly when the first $m$ elements of $b$ match the second $m$ elements of $b$: i.e., $b[0] = b[m], b[1] = b[m+1], \ldots, b[m-1] = b[2m-1]$. (Alternatively, $b[0..m-1]$ and $b[m..2m-1]$ are pointwise equal.) Example: If $b$ is $[1, 3, 5, 1, 3, 5]$, then $\text{Repeats}(b, 3)$ is true but $\text{Repeats}(b, 2)$ is false.
CS 536: Solution to Activity 2 (Propositional and Predicate Logic)

1. DeMorgan's law; Negation of $\rightarrow$ twice; commutativity of $\lor$.

2. $p \rightarrow p \lor q$
   
   $\Leftrightarrow \neg p \lor (p \lor q)$
   
   $\Leftrightarrow (\neg p \lor p) \lor q$
   
   $\Leftrightarrow T \lor q$
   
   $\Leftrightarrow T$
   
   Defn $\rightarrow$
   
   Asscociativity of $\lor$
   
   Excluded middle
   
   Domination

3. $(p \rightarrow q) \rightarrow (\neg q \rightarrow \neg p)$
   
   $\Leftrightarrow (\neg p \lor q) \rightarrow (\neg q \rightarrow \neg p)$
   
   $\Leftrightarrow (\neg p \lor q) \rightarrow (\neg q \lor \neg p)$
   
   $\Leftrightarrow (\neg p \lor q) \rightarrow (q \lor \neg p)$
   
   Pierce's law
   
   $\Leftrightarrow (\neg p \lor q) \rightarrow (\neg p \lor q)$
   
   Comm. of $\lor$
   
   $\Leftrightarrow \neg (\neg p \lor q) \lor (\neg q \lor \neg p)$
   
   $\Leftrightarrow T$
   
   Excluded middle (on $(\neg p \lor q)$)

4. If $q(x, y) = x < y \rightarrow y < z \land f(x) = 2$, then
   
   $\neg q(x, y)$
   
   $\Leftrightarrow \neg (x < y \rightarrow y < z \land f(x) = 2)$
   
   Defn of $\neg$
   
   $\Leftrightarrow x < y \land \neg (y < z \land f(x) = 2)$
   
   Negation of $\rightarrow$
   
   $\Leftrightarrow x < y \land (\neg (y < z) \lor \neg (f(x) = 2))$
   
   DeMorgan's Law
   
   $\Leftrightarrow x < y \land (y \geq z \lor f(x) \neq 2)$
   
   Negation of comparison, 3 times

5. (Minimal and full parenthesizations)
   
   I've included the outermost parentheses in the full parenthesization.
   
   a. $(\forall x . ((\exists y . x > y) \land (\exists y . x < y)))$
      
      Minimal: $(\forall x . (\exists y . x > y) \land \exists y . x < y)$
      
      Full: $(\forall x . ((\exists y . (x > y)) \land (\exists y . (x < y))))$
   
   b. $(\forall x . \neg (\exists y . p \land \forall z . q))$
      
      Minimal: $(\forall x . \neg (\exists y . p \land \forall z . q))$ [2/18]
      
      Full: $(\forall x . (\neg (\exists y . (p \land (\forall z . q))))))$
   
   c. $(\forall x . \forall y . \exists z . (x \neq y \rightarrow x \leq z \land z \leq y \land x > z \land z \geq y)$
      
      Minimal: $(\forall x . \forall y . \exists z . x \neq y \rightarrow x \leq z \land z \leq y \land x > z \land z \geq y)$
      
      Full: $(\forall x . (\forall y . (\exists z . ((x \neq y) \rightarrow (((x \leq z) \land (z \leq y)) \lor ((x > z) \land (z \geq y)))))$)

6. $(Q \land \forall y. Q \land \forall x)$
   
   a. Yes: $(\forall x. (\forall y . P(x, y)))$ is valid if and only if $(\forall y . (\forall x . P(x, y)))$ is valid
   
   b. Yes: $(\exists y . (\exists x . P(x, y)))$ is valid if and only if $(\exists y . (\exists x . P(x, y)))$ is valid

7. $(\neg (\forall x . \exists y . P(x, y)))$
   
   $\Leftrightarrow \exists x . \neg \exists y . P(x, y)$
   
   DeMorgan's Law $\neg \forall$
   
   $\Leftrightarrow \exists x . \forall y . \neg P(x, y)$
   
   DeMorgan's Law $\neg \exists$
8. \( \neg(\exists y \cdot \forall x . P(x, y)) \)  
\[ \Leftrightarrow \forall y . \neg(\forall x . P(x, y)) \quad \text{DeMorgan's Law } \neg \exists \]  
\[ \Leftrightarrow \forall y . \exists x . \neg P(x, y) \quad \text{DeMorgan's Law } \neg \forall \]

9. Here's one solution:

\[ \text{Repeats}(b, m) = \forall j. (0 \leq j < m \rightarrow m + j < \text{size}(b) \land b[j] = b[m+j]) \]

or \[ \text{Repeats}(b, m) = \forall 0 \leq j < m.(m+j < \text{size}(b) \land b[j] = b[m+j]), \text{with bounded quantifiers.} \]

There are other solutions, for example:

\[ \text{Repeats}(b, m) = 0 \leq 2 \times m \leq \text{size}(b) \land \forall j. (0 \leq j < m \rightarrow b[j] = b[m+j]) \]