ECS 20 Chapter 4, Logic using Propositional Calculus Handout

0. Introduction to Discrete Mathematics.
   0.1. Discrete = Individually separate and distinct as opposed to continuous and capable of infinitesimal change. Integers vs. real numbers, or digital sound vs. analog sound.
   0.2. Discrete structures include sets, permutations, graphs, trees, variables in computer programs, and finite-state machines.
   0.3. Five themes: logic and proofs, discrete structures, combinatorial analysis, induction and recursion, algorithmic thinking, and applications and modeling.
1. Introduction to Logic using Propositional Calculus and Proof
   1.1. "Logic" is "the study of the principles of reasoning, especially of the structure of propositions as distinguished from their content and of method and validity in deductive reasoning." (thefreedictionary.com)
2. Propositions and Compound Propositions
   2.1. Proposition (or statement) = a declarative statement (in contrast to a command, a question, or an exclamation) which is true or false, but not both.
      2.1.1. Examples: "Obama is president." is a proposition. "Obama will be re-elected." is not a proposition.
      2.1.2. "This statement is false." is a paradox that many would not consider a proposition.
      2.1.3. For ECS 20, we will use *p, q,* and *r* to symbolize propositions, subpropositions, or logical variables which take true or false values.
   2.2. Compound Proposition = a proposition that has its truth value completely determined by the truth values of two or more subpropositions and the operators (also called connectives) connecting them.
3. Basic Logical Operations
   3.1. Conjunction =$\wedge$ = "and." If both p and q are true, then $p \wedge q$ is true, otherwise $p \wedge q$ is false.
   3.2. Disjunction = $\vee$ = "or." If both p and q are false, then $p \wedge q$ is false, otherwise $p \wedge q$ is true.
   3.3. Negation = $\neg$ = "not." If p is true, then $\neg p$ is false. If p is false, then $\neg p$ is true.
4. Propositions and Truth Tables
   4.1. Let $P(p, q, \ldots)$ denote a propositional function constructed from the logical variables $p, q, \ldots,$ and logical operators.
   4.2. For order of precedence think of $\neg$ as unary minus, $\wedge$ as multiplication, and $\vee$ as addition. Parentheses have highest precedence. So (=) > $\neg$ > $\wedge$ > $\vee$.
   4.3. Two ways to construct truth tables. Both start by listing all the possible combinations the truth values of the primitive subpropositions in adjacent columns. One easy method for this part is to do binary counting and replace the 1's with T's, and the 0's with F's. Another method is have each primitive column have its truth value change half as often as the column to its right. The rightmost primitive column is initialized with alternating T's and F's.
      4.3.1. Method 1: Complete a new column for each operator based on the order of precedence, with the evaluated proposition written in the heading of the column.
      4.3.2. Method 2: Write the complete proposition with a column under each variable and under each operator. Fill in the values of the primitive subpropositions, and then complete the columns of the operators based on the order of precedence.
      4.3.3. Example: Use both methods to construct truth tables for $\neg(\neg p \vee q)$
         4.3.3.1. Method 1

4.3.3.2. Method 2

4.4. Example: Riddle 1105 from 4Chest.blogspot.com

There are four treasure chests. Where is the treasure?

P(1) Chest #1: "The treasure is not here."

P(2) Chest #2: "This chest is empty."

P(3) Chest #3: "Chest #1 is lying."

P(4) Chest #4: "Chest #1 tells the truth."

Rule(1) At least one tells the truth, and at least one is lying.

Rule(2) The treasure chest tells the truth.

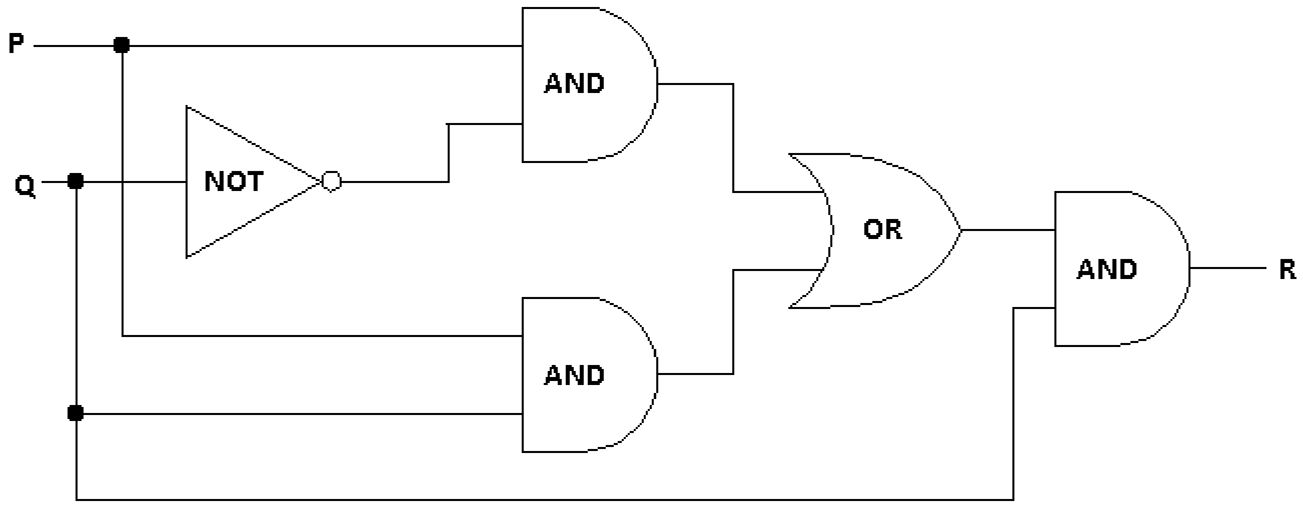
Chest #1	Chest #2	Chest #3	Chest #4	Treasure Chest	Reason to eliminate

5. Tautologies and Contradictions
   5.1. Tautology = a proposition that is always true regardless of the truth values of its subpropositions, e.g. $p \vee \neg p$
   5.2. Contradiction = a proposition that is always false regardless of the truth values of its subpropositions, e.g. $p \wedge \neg p$
6. Logical Equivalence = $\equiv$ = two propositions have identical truth values for all possible values of their logical variables.
   6.1. Prove by constructing the truth tables of the two propositions, and check that the truth values match for every combination of the logical variables, e.g.
   6.2. Example: Prove both De Morgan's Laws $\neg(p \wedge q) \equiv \neg p \vee \neg q$, and $\neg(p \vee q) \equiv \neg p \wedge \neg q$.
      6.2.1. English example of De Morgan's "and" Law: The negation of "Linda is a CS major, and she has at least a 3.0 GPA" would be "Linda is not a CS major, or her GPA is less than 3.0."
      6.2.2. English example of De Morgan's "or" Law: The negation of "Linda is a CS major, or she has at least a 3.0 GPA" would be "Linda is not a CS major, and her GPA is less than 3.0."
   6.3. Nonequivalence example $\neg(p \wedge q) \neq \neg p \wedge \neg q$

7. Algebra of Propositions: Laws of Logical Equivalence

Name	Or version	And version
Commutative	$p \vee q \equiv q \vee p$	$p \wedge q \equiv q \wedge p$
Associative	$(p \vee q) \vee r \equiv p \vee (q \vee r)$	$(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$
Distributive	$p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$	$p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$
Idempotent	$p \vee p \equiv p$	$p \wedge p \equiv p$
Identity	$p \vee \mathrm{F} \equiv p$	$p \wedge \mathrm{T} \equiv p$
	$p \vee \mathrm{T} \equiv \mathrm{T}$	$p \wedge \mathrm{F} \equiv \mathrm{F}$
Complement	$p \vee \neg p \equiv \mathrm{T}$	$p \wedge \neg p \equiv \mathrm{F}$
	$\neg\mathrm{T} \equiv \mathrm{F}$	$\neg\mathrm{F} \equiv \mathrm{T}$
Double Negative	$\neg(\neg p) \equiv p$	
De Morgan's	$\neg(p \vee q) \equiv \neg p \wedge \neg q$	$\neg(p \wedge q) \equiv \neg p \vee \neg q$
Absorption	$p \vee (p \wedge q) \equiv p$	$p \wedge (p \vee q) \equiv p$

8. Digital Logic Circuits



8.1. Simplify the above digital logic circuit using propositional algebra.

$((P \wedge \neg Q) \vee (P \wedge Q)) \wedge Q$

$\equiv (P \wedge (\neg Q \vee Q)) \wedge Q$ by the distributive law

$\equiv (P \wedge (Q \vee \neg Q)) \wedge Q$ by the commutative law for $\vee$

$\equiv (P \wedge \mathbf{T}) \wedge Q$ by the complement law for $\vee$

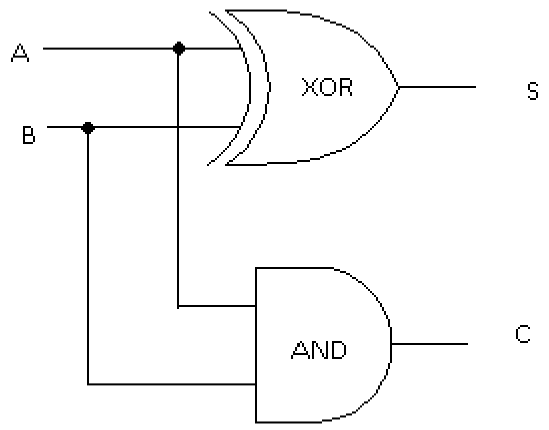
$\equiv P \wedge Q$ by the identity law for $\wedge$

8.2. Construct a digital logic circuit that adds two one-bit binary numbers *A* and *B*. It has two outputs, *S* and *C* (the value theoretically carried on to the next addition); the final sum is $2C + S$. Such circuits are called half adders.

8.2.1. Truth Table

8.2.2. Compound propositions: $C \equiv A \wedge B$, $S \equiv (A \vee B) \wedge \neg(A \wedge B) \equiv A \oplus B$, where $\oplus$ is XOR

8.2.3. Diagram



9. Conditional and Biconditional Statements

9.1. Conditional statement = $p \rightarrow q$ = "if p then q" = "p implies q" = "p only if q" = "p is sufficient for q" = "q is necessary for p" = "q, if p."

9.1.1. It is false only when the first part, p, is true and the second part, q, is false. (Write truth table on board)

9.1.2. Note that if the first part, p, is false then the statement is always true, so $p \rightarrow q \equiv \neg p \vee q$.

9.1.2.1. Converting an "Or" to an "If – then": $\neg p \vee q$ to a $p \rightarrow q$.

9.1.2.2. English Example "Either you pass ECS 20, or you cannot take ECS 60." becomes "If you do not pass ECS 20, then you cannot take ECS 60."

9.2. Example: Use the equivalence laws to show that the negation of "if p then q" is equivalent to "p and not q."

9.3. The "contrapositive" of a conditional statement of the form "if p then q" is "if $\neg q$ then $\neg p$."

9.3.1. Show that a conditional statement is logically equivalent to its contrapositive.

9.3.1.1. Method 1: Use truth tables.

9.3.1.2. Method 2: using equivalence laws

9.3.2. The statement "p only if q" means "if not q then not p," which is the contrapositive of "if p then q."

9.3.2.1. Example: "Jim will enjoy the movie only if it is an action film." becomes "If it is not an action film, then Jim will not enjoy it." Which has the contrapositive "If Jim will enjoy the movie, then it is an action film."

9.4. Biconditional statement = $p \leftrightarrow q$ = "p if and only if q." It is only true when p and q have the same truth value.

9.5. Truth tables of conditional, contrapositive, and biconditional statments

		Conditional		Contrapositive	Biconditional
p	q	$p \to q$	$\neg p \vee q$	$\neg q \to \neg p \equiv \neg\neg q \vee \neg p$	$p \leftrightarrow q$
T	T	T	T	T	T
T	F	F	F	F	F
F	T	T	T	T	F
F	F	T	T	T	T

9.6. Conditional and biconditional operators have the lowest precedence, and are coequal.

9.7. Code Examples from http://cnx.org/content/m10514/latest/

When writing a complicated conditional that involves multiple pieces of data, it is easy to incorrectly oversimplify. One strategy for avoid mistakes is to write such code in a two-step process. First, write a conditional with a case for **every** possible combination, as in a truth table. Second, simplify the conditional.

9.7.1. Example: Using this approach, we might obtain the following code after the first step. Simplify this code.

```
list merge_sorted_lists(list list1, list list2)
{
   if (is_empty(list1) and is_empty(list2))                  // P(1)
     return empty_list;
   else if (is_empty(list1) and not is_empty(list2))         // P(2)
     return list2;
   else if (not is_empty(list1) and is_empty(list2))         // P(3)
     return list1;
   else if (not is_empty(list1) and not is_empty(list2)) {
     if (first_element(list1) < first_element(list2))        // P(4)
       return make_list(first_element(list1), merge_sorted_lists(rest_elements(list1),
         list2));
     else if (first_element(list1) >= first_element(list2))  // P(5)
       return make_list(first_element(list2), merge_sorted_lists(list1,
         rest_elements(list2)));
   }
}
```

Second step solution:

is_empty(list1)	is_empty(list2)	first_element(list1) < first_element(list2)	Proposition	return
F	F	F	P(5)	make_list(first_element(list2)…
F	F	T	P(4)	make_list(first_element(list1)…
F	T	F	P(3)	list1
F	T	T	P(3)	list1
T	F	F	P(2)	list2
T	F	T	P(2)	list2
T	T	F	P(1)	empty_list = list1 = list2
T	T	T	P(1)	empty_list = list1 = list2

```
list merge_sorted_lists(list list1, list list2)
{
  if (is_empty(list1))
    return list2;
  else if (is_empty(list2))
    return list1;
   else {
     if (first_element(list1) < first_element(list2))
      return make_list(first_element(list1), merge_sorted_lists(rest_elements(list1),
        list2));
      else
         return make_list(first_element(list2), merge_sorted_lists(list1,
           rest_elements(list2)));
   }
}
```

9.7.2. Example :Consider the following function, which returns a boolean value.

```
bool foo(int  i, bool a, bool b)
{
   if (a and (i > 0))            // P(1)
      return b;
   else if (a and i <= 0)        // P(2)
      return false;
   else if (a or b)              // P(3)
      return a;
   else                          // P(4)
      return (i > 0);
}
```

Simplify it by filling in the following blank with a single Boolean expression. Do not use a conditional (such as **if**).


```
   return _______________________;
```

10. Argument = a sequence of propositions $P_1, P_2, \ldots P_n, Q$ with all but the final proposition called "premises," and the final proposition called the "conclusion." An argument is "valid" if the truth of all its premises implies the conclusion is true. Symbolically this would be stated as $P_1, P_2, \ldots P_n \vdash Q$ is valid if and only if $(P_1 \wedge P_2 \wedge \ldots \wedge P_n) \to Q$ is a tautology.

10.1. If an argument is valid, and in addition it started from true premises, then it is called a *sound* argument. A sound argument must arrive at a true conclusion.

10.2. Rules of inference

Name	Rule of Inference	Tautology
Modus [ponendo] ponens = "the way that affirms by affirming"	p $\underline{p \to q}$ $\therefore q$	$(p \wedge (p \to q)) \to q$
Modus [tollendo] tollens = "the way that denies by denying"	$\neg q$ $\underline{p \to q}$ $\therefore \neg p$	$(\neg q \wedge (p \to q)) \to \neg p$
Transitivity	$p \to q$ $\underline{q \to r}$ $\therefore p \to r$	$((p \to q) \wedge (q \to r) \to (p \to r)$
Elimination	$p \vee q$ $\underline{\neg p\quad}$ $\therefore q$	$((p \vee q) \wedge \neg p) \to q$
Generalization	$\underline{p\quad}$ $\therefore p \vee q$	$p \to (p \vee q)$
Simplification	$\underline{p \wedge q}$	$(p \wedge q) \to p$

	$\therefore p$	
Conjunction	p $\underline{q\quad}$ $\therefore p \wedge q$	$((p) \wedge (q)) \rightarrow (p \wedge q)$
Resolution	$p \vee q$ $\underline{\neg p \vee r}$ $\therefore q \vee r$	$((p \vee q) \wedge (\neg p \vee r)) \rightarrow (q \vee r)$

10.3. Valid argument using modus ponens: If it snows today, then we will go skiing. It is snowing today. Therefore we will go skiing.

10.4. Valid argument using modus ponens, though the conclusion is false because a premise is false:

If $\sqrt{3} < \frac{3}{2}$ then $\left(\sqrt{3}\right)^2 < \left(\frac{3}{2}\right)^2$. We know that $\sqrt{3} < \frac{3}{2}$. Therefore $\left(\sqrt{3}\right)^2 = 3 < \left(\frac{3}{2}\right)^2 = \frac{9}{4}$.

10.5. Is the following argument valid? Why? Is it true? Is it sound?

Either we are all doomed or we are all saved; we are not all saved; therefore, we are all doomed.

11. Show the following argument is valid (derived from *Discrete Mathematics and Its Applications* by Kenneth Rosen)

If today is Tuesday, I have a test in Mathematics or Economics. If my Economics professor is sick, I will not have a test in Economics. Today is Tuesday, and my Economics professor is sick. Therefore, I will have a test in Mathematics.

Converting to logical notation:

t = today is Tuesday
m = I have a test in Mathematics
e = I have a test in Economics
s = My Economics Professor is sick.

So the argument is:

The logical proof is:

Assertion	Reason

12. Propositional Functions, Quantifiers

12.1. Let A be a given set. A "propositional function" defined on A is an expression $p(x)$ which has the property that $p(x)$ is true or false for each $x \in A$. A is said to be the domain of $p(x)$.

12.2. T_p is the "truth set" of $p(x)$ which contains all elements of set A for which $p(x)$ is true.

12.2.1. $T_p = \{x \mid x \in A, p(x) \text{ is true}\}$, or more simply $T_p = \{x \mid p(x)\}$

12.3. Examples: Let $p(x)$ be "x is a factor of 12"

12.3.1. For the domain of the set of positive integers $\mathbf{Z}^+$, $T_p = \{1, 2, 3, 4, 6\}$

12.3.2. For the domain of the set of integers $\mathbf{Z}$, $T_p = \{-6, -4, -3, -2, -1, 1, 2, 3, 4, 6\}$

12.4. How many in the domain are in the truth set? Use "logical quantifiers" to specify the number of elements a set in terms of "all", or "some."

12.4.1. Universal quantifier = $\forall$= all elements.

12.4.2. Existential quantifier = $\exists$ = at least one element, ie. some.

12.4.3. Examples: Let $p(x)$ be "$2x > x$"

12.4.3.1. $(\forall x \in \mathbf{Z}^+)p(x) \equiv T_p = \{x \in \mathbf{Z}^+, p(x)\} = \mathbf{Z}^+$

12.4.3.2. $(\exists x \in \mathbf{Z})p(x) \equiv T_p = \{x \in \mathbf{Z}, p(x)\} \neq \varnothing$

12.5. Rules of Inference for quantified statements

12.5.1. Instantiation rules allow us to replace variables with names, thereby transforming quantified statements into statements about particular instances. "c/x" means that c is an instance of x.

12.5.2. Generalization rules allow us to move in the opposite direction, replacing names with variables and adding quantifiers to bind the variable.

Name	Rule of Inference
Universal instantiation	$\underline{\forall x\, p(x)}$ $\therefore p(c)$
Universal generalization	$\underline{p(c) \text{ for an arbitrary } c}$ $\therefore \forall x\, p(x)$
Existential instantiation	$\underline{\exists x\, p(x)}$ $\therefore p(c)$
Existential generalization	$\underline{p(c) \text{ for some element } c}$ $\therefore \exists x\, p(x)$

13. Negation of Quantified Statements allow for specifying “not all,” and “none.”
    - 13.1. De Morgan for the universal quantifier: $\neg(\forall x \in A)p(x) \equiv (\exists x \in A)\, \neg p(x)$
        - 13.1.1. Example: “Not all CS majors play video games.” ≡ “Some CS majors do not play video games.”
    - 13.2. De Morgan for the existential quantifier: $\neg(\exists x \in A)p(x) \equiv (\forall x \in A)\, \neg p(x)$
        - 13.2.1. Example: “No CS major likes bugs in their programs.” ≡ “All CS majors don’t like bugs in their programs.”
14. Quantization Proof Example: Show that the premises “A student in this class has not read the book,” and “Everyone in this class passed the first exam” imply the conclusion “Someone who passed the first exam has not read the book.”

Symbolization:

Symbolic argument:

Proof

Assertion	Reason