Review

First, let’s review some terminology.

a. A **propositional formula** is a function of one or more variables, each of which can be set to true or false, that evaluates to true or false. We call a propositional formula a **proposition** for short.

b. The term **logical expression** is often used synonymously with the word proposition.

c. Two propositions are **logically equivalent** when they have the same truth tables.

d. A proposition is **valid** if it evaluates to true on any choice of inputs; it is true no matter what. That is, a valid proposition is logically equivalent to the expression \((p \lor \neg p)\). This is also called a **tautology**.

e. A proposition is **satisfiable** if it evaluates to true on some choice of inputs. A valid proposition is satisfiable, but so are many propositions which sometimes evaluate to false.

f. If a proposition is not satisfiable, it evaluates to false on any choice of inputs; it is false no matter what. That is, it is logically equivalent to the expression \((p \land \neg p)\). This is called a **contradiction**.

Let’s now review the interpretation of each of the following logical connectives:

<table>
<thead>
<tr>
<th>(P)</th>
<th>(Q)</th>
<th>(\neg P)</th>
<th>(P \land Q)</th>
<th>(P \lor Q)</th>
<th>(P \oplus Q)</th>
<th>(P \implies Q)</th>
<th>(P \iff Q)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
</tbody>
</table>

and here’s a guide for the notation:

- \(\neg\) means “not”
- \(\land\) means “and”
- \(\lor\) means “or”
- \(\oplus\) means “or, but not both”, which we call “XOR”
Why do we have propositions?

Propositions are condensed ways of representing truth tables. Because they’re more condensed, we oftentimes can more easily extract the meaning from them. It can be hard to look at a truth table and see the “general rule” that transforms input to output. A proposition is that rule.

Remember when we represented functions in the form $f(x) = ...$? This notation is a general rule for creating the set of ordered pairs that is the function. In the same way, a proposition is a general rule for creating the truth table.

Logical Equivalence: Two approaches

You have two techniques at your disposal to determine if two expressions are logically equivalent: by using truth tables or by using logical rewrite rules.

Given two expressions, you can write out the truth table for each one. If they have the same inputs and their truth tables are the same, they are logically equivalent.

Or, given two expressions, you use logical equivalence rules to try to get from one expression to the other. For example, take the expression $\neg(x \lor y)$. Using DeMorgan’s law, you could get that the not “distributes” in this expression, and it is therefore equivalent to $(\neg x \land \neg y)$. A full list of rules you can use can be found on the course website.

TASK: Now you may be wondering, where do these rules come from? Discuss with your neighbors where these laws could come from, and then call over a TA.

TASK: Determine if each pair of expressions below are logically equivalent (i.e. determine if the equivalence stated holds). Use truth tables for some and rewrite rules for others, and note which one you use.

i. $p \lor q \equiv \neg p \land \neg q$

<table>
<thead>
<tr>
<th>True</th>
<th>False</th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>True</td>
</tr>
</tbody>
</table>
ii. \((p \land q) \land z \equiv (p \land (q \land z))\)

True

iii. \((p \lor q) \land z \equiv (p \lor (q \land z))\)

False

iv. \(\neg p \Rightarrow q \equiv p \lor q\)

True

v. \(p \Leftrightarrow q \equiv (p \land q) \lor (\neg p \land \neg q)\)

True

vi. \(p \land \neg q \equiv \neg (\neg p \lor q)\)

True

vii. \((p \Rightarrow (z \Rightarrow q)) \equiv ((p \land z) \Rightarrow q)\)

True

Checkpoint - call over a TA

Normal Forms

We say a proposition is in **DNF** (disjunctive normal form) when it is the disjunction (clauses ORed together) of conjunctions (terms ANDed together).

We say a proposition is in **CNF** (conjunctive normal form) when it is the conjunction (clauses ANDed together) of disjunctions (terms ORed together).

Here’s a truth table, and propositions in DNF and CNF which represent it:

<table>
<thead>
<tr>
<th>(P)</th>
<th>(Q)</th>
<th>(R)</th>
<th>?</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>
DNF: \((P \land Q \land \neg R) \lor (P \land \neg Q \land \neg R) \lor (\neg P \land Q \land \neg R) \lor (\neg P \land \neg Q \land \neg R)\)

CNF: \((\neg P \lor \neg Q \lor \neg R) \land (\neg P \lor Q \lor \neg R) \land (P \lor \neg Q \lor R) \land (P \lor Q \lor R)\)

If we have an arbitrary truth table, here are two ways we can think about describing it:

- Listing the true rows
- Listing the false rows

Since every row must be either true or false, both of these ways will uniquely describe our truth table.

These two ways correspond to DNF and CNF, respectively. To write a proposition in DNF, we can think about it like this: we find all rows where our proposition should evaluate to true, and we say that we must be in one of those rows. On the other hand, to write a proposition in CNF, we find all rows where our proposition should evaluate to false, and say we are not in any of those rows.

**TASK:** How do we specify that we are in one of the true rows (DNF)? How do we specify that we are not in any of the false rows (CNF)? *Hint:* Look at the DNF and CNF representations of the truth tables above. How do they relate to this idea?

For DNF, we AND the true variables and negations of the false variables (to be in the row, the inputs must exactly correspond to the row). For CNF, we OR the false variables and the negations of the true variables (to not be in the row, we just need at least one variable to be different).

**TASK:** Write two propositions corresponding to the following truth table: one in DNF and one in CNF.

<table>
<thead>
<tr>
<th>(P)</th>
<th>(Q)</th>
<th>(R)</th>
<th>?</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>

DNF: \((P \land Q \land R) \lor (P \land Q \land \neg R) \lor (P \land \neg Q \land \neg R) \lor (\neg P \land Q \land \neg R) \lor (\neg P \land \neg Q \land \neg R)\)

CNF: \((\neg P \lor Q \lor \neg R) \land (\neg P \lor \neg Q \lor R) \land (P \lor Q \lor R) \land (P \lor Q \lor R)\)
More Practice

a. Give an assignment to the variables $x_1, x_2, x_3$ which makes the following logical expression evaluate to true.

$$(x_1 \lor x_2) \land (\neg x_1 \lor \neg x_2 \lor \neg x_3) \land x_3$$

- $x_1$ is true, $x_2$ is false, $x_3$ is true

b. Is the following logical expression valid? Explain your answer.

$$(x_1 \land x_2) \lor (\neg x_1 \land \neg x_2 \land \neg x_3) \lor (x_3 \land \neg x_3)$$

- Nope, consider $x_1$ is false and $x_2$ is true

c. Come up with a logical expression with three variables which has only one assignment to the variables which makes it true.

- All the variables anded together

d. Given inputs $p$ and $q$, create an expression which outputs $p \oplus q$ using only not, or, and and.

- Something like: $(p \land \neg q) \lor (\neg p \land q)$

Checkpoint - call over a TA
From Propositions to Circuits

Circuits are another way of representing truth tables. However, circuits can have multiple outputs, unlike propositions; that is circuits can represent more than one truth table. The number of outputs they have corresponds to the number of truth tables they represent.

Their representation is supposed to help visualize how the output is computed given the inputs, and because computer science is very much about the how, computer scientists often like these representations of truth tables very much.

Here’s a quick review of the notation.

Exercises

1. Given inputs $x_1$, $x_2$ and $x_3$, create a circuit which outputs true if and only if exactly one input is true.

This isn’t a circuit but... something like: $(x_1 \land \neg x_2 \land \neg x_3) \lor (\neg x_1 \land x_2 \land \neg x_3) \lor (\neg x_1 \land \neg x_2 \land x_3)$

2. Is the circuit you just created logically equivalent to $(x_1 \oplus x_2) \oplus x_3$?

    Nope

3. Create a 3-input circuit that outputs true when exactly 2 of the inputs are true, and false otherwise.
There are lots of ways to do this! Beware that the third input needs to be false if 2 of the inputs are true for the circuit to evaluate to true.

4. Create a 3-input circuit with two outputs. The first output should be true only when all three of the inputs are true, and the second output should be true only when all three inputs are true or all three inputs are false.

Lots of ways to do this!

5. In what situations are the first and second output of the previous circuit equal?

If they are both true, then all three inputs are true. If they are both false, then the three inputs do not agree with each other.

Checkpoint - call over a TA

**TASK:** Suppose you have a one-output circuit for some arbitrary truth table. Is there a quick way you could make a different circuit representing that same truth table? *Hint:* try adding gates to the end of the circuit. What gate can take in just one input?

Append two NOT gates to the end

**TASK:** Using this method, are there infinitely many distinct one-output circuits representing that same truth table? How about infinitely many propositions?

Yes, just keep adding two NOTs

**TASK:** Suppose $C_1$ and $C_2$ are distinct one-output circuits but represent the same truth table. What reasons would we have to prefer one over the other?

Size, meaning