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. There is a nuance here; what if they take in a different number of variables? Could they still be logically equivalent? **TASK**: Discuss this with your partner.

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

e. A **contradiction** is a proposition that evaluates to false on any choice of inputs; it is false no matter what. That is, a contradiction is logically equivalent to the expression \((p \land \neg p)\).

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

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>Q</td>
<td>¬P</td>
<td>P ∧ Q</td>
<td>P ∨ Q</td>
<td>P ⊕ Q</td>
<td>P ⇒ Q</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<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>
<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>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
</tbody>
</table>

and here’s a pronunciation guide for the notation:

- ¬ means “not”
- ∧ means “and”
- ∨ means “or”
- ⊕ means “or, but not both”, which we call “XOR”
- ⇒ means “implies”
- ⇔ means “if and only if”
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. We’ll call these two ways the semantic approach and syntactic approach.

The semantic approach works like this. Given two expressions, you write out the truth table for each one. You make sure that their input columns are organized the same. If their final columns match, then you have that the two expressions are logically equivalent.

The syntactic approach works like this. Given two expressions, you use syntactic rules of transformation 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 \((\neg x \land \neg y)\). Now you may be wondering, where do these laws, or rules of transformation, come from?

**TASK:** Discuss with your neighbors where these laws could come from, and then call over a TA.

Sometimes, a blend of these techniques are used to prove two expressions are logically equivalent. There’s no need to stick strictly to one or the other.

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

1. \( p \lor q \equiv \neg p \land \neg q \)
ii. \[ ((p \land q) \land z) \equiv (p \land (q \land z)) \]

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

iv. \[ \neg p \Rightarrow q \equiv p \lor q \]

v. \[ p \leftrightarrow q \equiv (p \land q) \lor (\neg p \land \neg q) \]

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

vii. \[ (p \Rightarrow (z \Rightarrow q)) \equiv ((p \land z) \Rightarrow q) \]
Checkpoint - call over a TA

Understanding negation through truth tables

We say a proposition is in **DNF form** when it is the disjunction (a sling of ORs) of conjunctions (a sling of ANDs).

**TASK:** Consider a truth table for an arbitrary proposition, \( P \).

a. Suppose the truth table of \( P \) has \( n \) rows. If \( P \) evaluates to true in \( k \) of the rows, and in \( n - k \) of them to false, in how many rows will the negation of \( P \) evaluate to true? To false?

b. Using \( P \)'s truth table, create a proposition for the negation of \( P \), \( \neg P \), in DNF form.

c. Explain how the DNF expression you created explains the following statement: “it takes just one counterexample to \( P \) for \( P \) to be false; that is, for the negation of \( P \) to be true.”
More Practice

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

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

b. Is the following logical expression a tautology? Explain your answer.

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

c. Come up with a logical expression with three variables which has only one assignment to the variables which makes it true.
d. Given inputs $p$ and $q$, create an expression which outputs $p \oplus q$ using only not, or, and and.

Checkpoint - call over a TA

From Propositions to Circuits

Circuits are a more procedural way of representing truth tables, with one caveat. 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.

\[
\begin{align*}
\text{AND Gate} & \quad \begin{array}{c}
\text{OR Gate} \\
\text{Not Gate} \\
\text{XOR Gate}
\end{array}
\end{align*}
\]
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.

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

3. Create a 3-input circuit that outputs true when exactly 2 of the inputs are true, and false otherwise.

4. Create a 3-input circuit with two outputs. The first output should be true only when 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.
5. In what situations are the first and second output of the previous circuit equal?

On the last homework, you proved that there are infinitely many distinct syntactically but logically equivalent propositions. Let’s think about if this is true when it comes to one-output circuits: are there infinitely many distinct one-output circuits we can use to express one truth table?

**TASK:** Suppose you have a one-output circuit for some arbitrary truth table. Are there infinitely many distinct one-output circuits representing that same truth table? Prove your answer.

**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?

**Different Representations for Truth Tables**

We’ve now gone through both propositions and circuits as representations for truth tables. Part of our job as computer scientists is to compare these representations and to think about the contexts in which one is preferable over the other.
**TASK:** What are some reasons one could prefer a truth table represented via proposition over a circuit? And what are some reasons one could prefer a circuit over a proposition?

**TASK:** Propositions and circuits are just two ways to represent truth tables, but we could think of a novel way to represent them! With your neighbors, think about a novel way in which we could represent truth tables, and try to motivate your representation to a TA (i.e. think of situations where your representation would be useful).