Chapter Two

# REASONING

elaine rich
alan kaylor cline

The Logicians on our cover are:

Euclid (? - ?)

Augustus De Morgan (1806 – 1871)

Charles Babbage (1791 – 1871)

George Boole (1815 – 1864)

Aristotle (384 BCE – 322 BCE)

George Cantor (1845 – 1918)

Gottlob Frege (1848 – 1925)

John Venn (1834 – 1923)

Bertand Russell (1872 – 1970)

# REASONING

# AN INTRODUCTION TO LOGIC, SETS, AND FUNCTIONS

## CHAPTER 2
## BOOLEAN LOGIC

Elaine Rich
Alan Kaylor Cline

*The University of Texas at Austin*

**Image credits:**

George Boole: http://www.kerryr.net/pioneers/gallery/ns_boole2.htm

I-9: http://www.learnlogiconline.org/I-9.jpg



http://www.cs.utexas.edu/learnlogic
Library of Congress Cataloging-in-Publication Data

Rich, Elaine, 1950 -
Reasoning—An Introduction to Logic Sets and Functions / Elaine Rich.— 1st ed. p. cm.
ISBN x-xxx-xxxxx-x 1

# Table of Contents

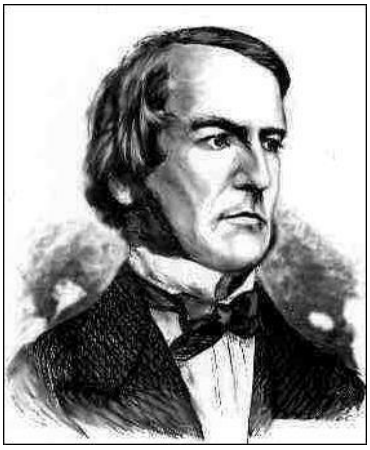
# Boolean Logic

## Statements

### Introduction

Boolean logic is simple. It's useful. (For example, computer circuits implement Boolean logic.) And most other logical frameworks have it at their core. So it is a good place for us to start.

Boolean logic gets its name from George Boole, a nineteenth century British logician.



You may also sometimes see Boolean logic called ***sentential calculus***, ***propositional logic*** or ***$0^{th}$ order logic***.

There is a natural analogy between Boolean logic and the systems of arithmetic and algebra with which you are already familiar. You'll see this analogy emerge as we go along, starting on the next page.

## The Building Blocks of Statements

The first question we have to answer is: What are the fundamental objects that we can talk about?

- In arithmetic, there are numbers, such as 0, 127, -4, and 7.865.
- In Boolean logic, there are ***truth values***. More specifically, there are two truth values, True (which we'll often write just as *T*) and False (written as *F*).

You've probably noticed one difference already between arithmetic and Boolean logic: there is an infinite number of numbers but only two truth values. We'll see later that this matters; it is one reason why Boolean logic is as useful as it is. But, lest you think that logics can only deal with very small sets of objects, we'll just mention here that, later in this course, we'll describe a more powerful logical system that lets us talk about infinite universes (including numbers).

Next, we need ***variables*** (or names for things that may take on any of the fundamental values):

- In algebra, there are variables. You are used to giving them names like *x* and *y*.
- In Boolean logic, there are also variables. We will give them names like *p* and *q*.

In algebra, we use variables to stand for quantities that we want to reason about. For example, we might let *x* stand for Mary's age and *y* stand for John's age.

In Boolean logic, we do the same thing except that now we want to reason about claims that may be true or false. So, for example, we might let *p* stand for the claim, "It is raining," and *q* stand for the claim, "The sidewalks are wet."

Recall that a ***statement*** (sometimes also called a ***proposition***) is something to which we can attach a truth value. So we now have the beginning of a formal definition of a Boolean logic statement (also sometimes called a ***well-formed formula*** or ***wff***, pronounced "woof"):

- True (usually abbreviated *T*) is a statement. It is always true.
- False (usually abbreviated *F*) is a statement. It is always false.
- Individual claims (which we may write in English or we may denote with variables like *p*) are statements. They may be either true or false.

In the next section, we'll see how to combine simple statements to make more expressive ones.

## Operators and Operands

We won't get very far if all we have to use are simple statements alone – we need to be able to combine statements to form new, more expressive ones. This is similar to working with numbers: if you have no operations (such as addition and multiplication), you can't do much.

In logic, statements take the place of numbers. Logical operations will replace arithmetic operations. Otherwise, you will see that the two systems, arithmetic and logic, are very similar.

We shall call the logical operations ***operators***. Examples are *not*, *or*, *and*, and *implies* (just as $+$, $-$, and $\times$ are arithmetic operators). We shall call statements ***operands***. That just means the objects on which the operators act.

Thus, if we say, "Jim is tall *or* Joe is short", there are two operands: "Jim is tall" and "Joe is short", and there is one operator: *or*.

If we say, "Jim is *not* tall", there is one operand, "Jim is tall", and there is one operator: *not*.

***English Aside***

English, in the interest of efficient communication, sometimes lets us shorten sentences so that it's not completely obvious what the operands are. To find them, we may need to expand sentences into their full constituent parts.

For example, consider the sentence, "Chris likes tacos and guacamole". There are really two statements here, conjoined with "and". The two statements (operands) that are lurking here are, "Chris likes tacos" and "Chris likes guacamole".

## Problems

1. Consider the statement, “Kittens are cute and lizards are slimy”.

(Part 1) How many operands does it contain?

(a) 0
(b) 1
(c) 2
(d) 3
(e) 4

(Part 2) How many operators does it contain?

(a) 0
(b) 1
(c) 2
(d) 3
(e) 4

2. Consider the statement, “Kittens are cute and are not slimy”.

(Part 1) How many operands does it contain?

(a) 0
(b) 1
(c) 2
(d) 3
(e) 4

(Part 2) How many operators does it contain?

(a) 0
(b) 1
(c) 2
(d) 3
(e) 4

## Using Operators to Build Complex Statements

In arithmetic, operators, such as + and × have meanings. If we know the value of $y$, then we know the value of $y+2$. Similarly, the logical operators that we are about to define also have meanings. In particular, we will define them so that they correspond to natural ways of reasoning. For now, assume that they mean approximately what you think they mean. For example, "$p$ and $q$" is true just in the case where both $p$ and $q$ are true. Similarly, "not $p$" is true only in the case where $p$ is false. We will give precise meanings to all of our operators shortly (using a clever device called a truth table).

But, before we do that, we can notice that the job of all of the operators is to construct truth values from the truth value(s) of their operand(s). This means that every logical expression, no matter how many operators it contains, has a truth value. So it meets our definition of a statement. Thus:

- Any logical expression that is formed by combining operands with operators is a statement. It may be either true or false. Using the definitions of the logical operators it contains, we can compute its truth value from the truth values of its operands

For example:

- "Jim is tall" is a statement. It may be true or false.
- "Joe is short" is a statement. It may be true or false.

So:

- "Jim is tall or Joe is short" is a statement. It is true if at least one of the two simple claims is true.
- "Jim is tall and Joe is short" is a statement. It is true only if both of the two simple claims are true.

***Big Idea***

We can build logical expressions from operands and meaningful operators in much the same way that we build arithmetic expressions.

## Problems

1. Which of the following expressions is *not* a logical statement?

   (a) Rome is the capital of France.
   (b) The sidewalks are wet and Paris is the capital of France.
   (c) The sidewalks are wet and Rome is the capital of France.
   (d) What is the capital of France?

2. Which of the following expressions is *not* a logical statement?

   (a) The moon is made of lemon juice.
   (b) The sidewalks are wet Paris is the capital of France.
   (c) Dragons breathe fire or thunder.
   (d) Not not Rome is the capital of Italy.

3. Assume the world in which we live. Which of the following is *not* a *true* statement:

   (a) Rome is the capital of France or Rome is not the capital of France.
   (b) Paris is the capital of France.
   (c) Paris is the capital of France and Paris is not the capital of France.
   (d) Paris is the capital of France and Rome is the capital of Italy.

## Operator Precedence and Parentheses (or How to Use Multiple Operators in a Single Statement)

"Jim is tall or Joe is short" is a statement (that happens to be the *or* of two more basic statements). Since it is a statement, we can turn around and use it as an operand to form an even larger statement. Combining it with the simple statement "Mary is young", we get the more complex statement:

"Mary is young and Jim is tall or Joe is short".

We now have two operators. But how do we know the order in which they should be applied? Just looking at the statement (without knowing that we started with "Jim is tall or Joe is short"), it is possible that we meant:

"Mary is young" *and* "Jim is tall or Joe is short"

Or we could have meant:

"Mary is young and Jim is tall" *or* "Joe is short"

Notice that our analogy with arithmetic continues here. Suppose we write:

$3+5\times 2$

We could mean:

$(3+5)\times 2$, which would be 16.

Or we could mean:

$3+(5\times 2)$, which would be 13.

We will solve this problem with logical expressions in exactly the same way we solve it with arithmetic expressions:

- We will assign a ***precedence level*** to each of our operators. All operators with higher precedence get applied before any operators of lower precedence do. So, in arithmetic, we know that -3 + 5 is 2 because we apply the minus sign to the value 3 before we add 5. We don't first add 3 to 5 and then apply the minus sign to get -8.Minus has higher precedence than addition does. We will define precedence levels for our logical operators after we have described them in detail in the next few sections.

- We can use parentheses whenever the arbitrary precedence levels don't do what we need them to do (or, often, when we're not sure what they are going to do and

we don't want to take a chance of getting it wrong). So we can write either of these:

Mary is young and (Jim is tall or Joe is short).

(Mary is young and Jim is tall) or Joe is short.

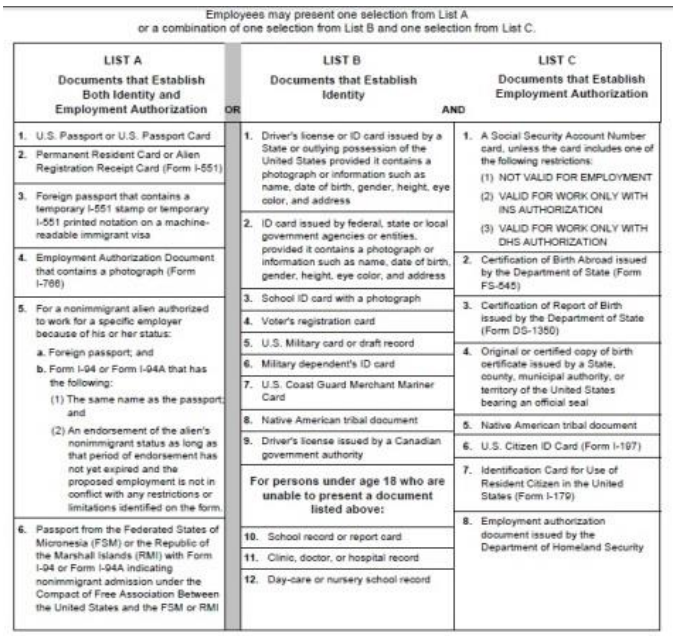
***English Aside***

Of course, English doesn't get to use parentheses willy nilly. Good writers try to get the same effect with careful punctuation. And sometimes signs exploit layout and spacing. A lot of the time, writers count on the ability of the rest of us to figure out what makes sense. See the next two examples.

Can you figure out what a tourist's options are at this café (in Cartagena, Spain)?

Some restaurants seem to get it, though. Can you see how our local Russian restaurant has indicated *and*s and *or*s and shown the correct groupings of them? (The *or*s here correspond to the logical operator that we're going to call *exclusive-or*. You may have exactly one of the items in the list.)


Legal documents must solve this problem. For some reason, they don't like just to use parentheses, which would make it all very easy. So they have to try other things. Here is an example that uses blocks on the page to describe one's choices for documents that can be submitted along with a US I-9 form (that certifies that one is legally allowed to work in the US).



Employees may present one selection from List A or a combination of one selection from List B and one selection from List C.

| LIST A<br>Documents that Establish Both Identity and Employment Authorization | OR | LIST B<br>Documents that Establish Identity | AND | LIST C<br>Documents that Establish Employment Authorization |
|---|---|---|---|---|
| 1. U.S. Passport or U.S. Passport Card | | 1. Driver's license or ID card issued by a State or outlying possession of the United States provided it contains a photograph or information such as name, date of birth, gender, height, eye color, and address | | 1. A Social Security Account Number card, unless the card includes one of the following restrictions:<br>(1) NOT VALID FOR EMPLOYMENT<br>(2) VALID FOR WORK ONLY WITH INS AUTHORIZATION<br>(3) VALID FOR WORK ONLY WITH DHS AUTHORIZATION |
| 2. Permanent Resident Card or Alien Registration Receipt Card (Form I-551) | | 2. ID card issued by federal, state or local government agencies or entities, provided it contains a photograph or information such as name, date of birth, gender, height, eye color, and address | | 2. Certification of Birth Abroad issued by the Department of State (Form FS-545) |
| 3. Foreign passport that contains a temporary I-551 stamp or temporary I-551 printed notation on a machine-readable immigrant visa | | 3. School ID card with a photograph | | 3. Certification of Report of Birth issued by the Department of State (Form DS-1350) |
| 4. Employment Authorization Document that contains a photograph (Form I-766) | | 4. Voter's registration card | | 4. Original or certified copy of birth certificate issued by a State, county, municipal authority, or territory of the United States bearing an official seal |
| 5. For a nonimmigrant alien authorized to work for a specific employer because of his or her status:<br>a. Foreign passport; and<br>b. Form I-94 or Form I-94A that has the following:<br>(1) The same name as the passport; and<br>(2) An endorsement of the alien's nonimmigrant status as long as that period of endorsement has not yet expired and the proposed employment is not in conflict with any restrictions or limitations identified on the form. | | 5. U.S. Military card or draft record | | 5. Native American tribal document |
| | | 6. Military dependent's ID card | | 6. U.S. Citizen ID Card (Form I-197) |
| | | 7. U.S. Coast Guard Merchant Mariner Card | | 7. Identification Card for Use of Resident Citizen in the United States (Form I-179) |
| | | 8. Native American tribal document | | |
| 6. Passport from the Federated States of Micronesia (FSM) or the Republic of the Marshall Islands (RMI) with Form I-94 or Form I-94A indicating nonimmigrant admission under the Compact of Free Association Between the United States and the FSM or RMI | | 9. Driver's license issued by a Canadian government authority | | 8. Employment authorization document issued by the Department of Homeland Security |
| | | For persons under age 18 who are unable to present a document listed above: | | |
| | | 10. School record or report card | | |
| | | 11. Clinic, doctor, or hospital record | | |
| | | 12. Day-care or nursery school record | | |

But it's not always possible, and sometimes it's next to *impossible*, to figure out what was meant.

Can you figure out how to add parentheses to this menu fragment to make it clear what a hungry eater's choices are:

OCEANIA CHEESE PLATE SELECTION
Stilton – Crottin - Brique Chateau
*Served with Homemade Cranberry and Fig Chutney, Cake of the Day, and Pressed Dry Fig and Almond or Date and Almond Cake*

***Big Idea***

One of the big reasons that we've defined our logical language in the way we have is that it doesn't force readers (be they people or machines) to guess what writers meant.

## Problems

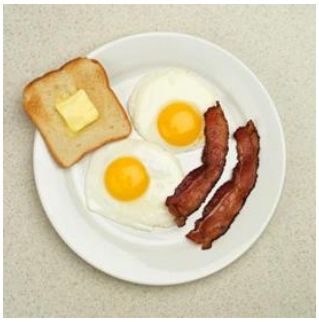
1. Let's assume that a word, like "pancakes", corresponds to the claim, "You can have pancakes." Then consider this expression:

pancakes or bacon and eggs

Which of these parenthesizations is much more likely to correspond to what a menu-writer meant:

(a) pancakes or (bacon and eggs)

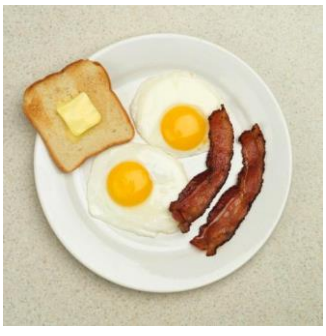
or



(b) (pancakes or bacon) and eggs

or

.



## Formal Definitions of the Logical Operators

Suppose we know that the statement "Jim is tall" is true but that the statement "Joe is short" is false; what is the truth value of "Jim is tall or Joe is short"? You say, "It's true," and you are correct. However, had we asked what is the truth value of "Jim is tall *and* Joe is short" (under the same circumstances), you should say, "It's false". Obviously, then, the action of the operator *and* versus the action of *or* is critical for determining truth values.

In answering the questions that we just asked, you used our everyday notions of what *and* and *or* mean. We have those notions because they are useful. They help us to reason about the world.

We are embarking on a study of formal logic. Our goal is to describe a system so precisely that there can be no disagreement about what we mean or about whether an argument that we make is valid. So we need to provide precise definitions of every operator that we'll use. But, of course, we still want operators that let us do useful things (as opposed to formally justified, yet silly things).

So, in the next several sections, we will define precisely what we mean by each of the operators that we will use. We hope that you'll agree that we've chosen to define a set of operators that correspond well to our intuitions about sound reasoning and, thus, that we'll be able to use them in useful ways.

## Truth Table Definitions of Operators

### Introduction

An individual proposition has a truth value, which can be either true (*T*) or false (*F*). A compound statement derives its truth value from the truth values of its components. We'd like to have a simple tool that lets us consider a compound statement and then see what truth value it would have, given any of the possible combinations of truth values of its components.

Truth tables let us do precisely this. We can use them to define the individual operators that we want to use. And we can simply extend them as a way to compute the truth value of an arbitrarily complex statement.

To simplify the writing out of truth tables (and do a tiny bit of abstraction), we will use variable names, $p$ and $q$, for statements. Also, we'll let *T* stand for "true" and *F* stand for "false".

A truth table for a logical statement has one title row. Next, it has one row for each logically possible combination of truth values of the variables in that statement. It has one column for each constituent statement, including the smallest ones (the individual variables), all the intermediate ones, and the complete statement itself.

For example, here's the outline of a truth table for a statement that contains two variables and uses two operators:

| *p* | *q* | *p and q* | *not* (*p and q*) |
|---|---|---|---|
| *T* | *T* | | |
| *T* | *F* | | |
| *F* | *T* | | |
| *F* | *F* | | |

Notice that, after the title row, which we won't count, there are four rows, each corresponding to one of the four logically possible combinations of the two truth values.

> ***Big Idea***
>
> Truth tables are a powerful tool for defining the meaning of Boolean logic statements. We'll see in the next chapter that they're also a tool for creating Boolean logic proofs.

In the next several sections, we'll see how to fill in truth tables, starting with definitions of the individual operators.

## A Truth Table Definition of the Operator *and*

We'll begin with the operator *and*. We'll define it using a truth table. You'll see that our definition corresponds to your everyday intuition about what *and* means.

In the rest of this discussion, we'll use the symbol $\wedge$ to mean *and*. Thus, $p \wedge q$ is read "*p* and *q*". The symbol & is also sometimes used to mean *and*.

By the way, you can remember this notation by noticing that the symbol for AND looks like a capital letter A that is simply missing its center bar:


Here is the truth table for the *and* operator:

| $p$ | $q$ | $p \wedge q$ |
|---|---|---|
| $T$ | $T$ | $T$ |
| $T$ | $F$ | $F$ |
| $F$ | $T$ | $F$ |
| $F$ | $F$ | $F$ |

We can read the first (non-title) line as, "When *p* is true and *q* is true, then $p \wedge q$ is true". Look at the entire table and notice that the *only* situation in which $p \wedge q$ is true is when both *p* is true *and* *q* is true. Notice, also, that the four lines in the table provide a definition for every possible combination of the truth values of *p* and *q*.

We'll call a statement that is formed by *and*-ing together other statements a ***conjunction***.

Some examples of simple conjunctions are

- $p \wedge q$
- $p \wedge (q \wedge r)$

***English Aside***

Consider the following English sentences:

- "Jim is tall and Joe is short."
- "Jim is tall but Joe is short."
- "Jim is tall yet Joe is short."
- "Jim is tall, however Joe is short."
- "Even though Jim is tall, Joe is short."

They all mean the same thing at the level of their literal propositional content. Given any actual state of affairs (in this case, the heights of Jim and Joe), either all of these sentences are true or none of them is. So, for the analysis that we will be doing, the English words (phrases) "and", "but", "yet", and "even though …" all correspond to the logical operator we are calling and.

Of course, there is a reason that English gives us these alternatives. They allow us to convey different additional kinds of information. For example, we use "but" to suggest that the second proposition is perhaps unexpected. We will not try to encode that extra information in the examples that we'll be considering.

## Problems

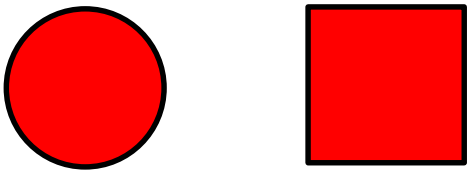
1. Recall the truth table for *and* that we just presented:

| $p$ | $q$ | $p \wedge q$ |
|---|---|---|
| $T$ | $T$ | $T$ |
| $T$ | $F$ | $F$ |
| $F$ | $T$ | $F$ |
| $F$ | $F$ | $F$ |

Let $p$ correspond to the claim, "The circle is red."
Let $q$ correspond to the claim, "The square is blue."

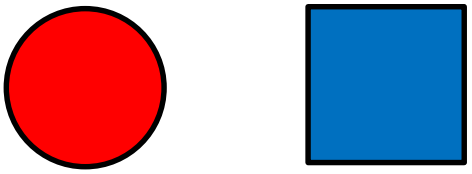
Consider the claim, "The circle is red and the square is blue."

(Part 1) Consider this situation:



Which row of the *and* truth table tells us whether our claim is true?

(Part 2) Consider this situation:



Which row of the *and* truth table tells us whether our claim is true?

2. Let $p$ correspond to the claim: One terabyte equals approximately $10^6$ megabytes.

Let $q$ correspond to the claim:

In 1950, the biggest computer in the world had less than a megabyte of memory.

Consider the claim: $p \wedge q$

Which row of the *and* truth table tells us whether our claim is true? (Hint: Go look up the facts.)

## A Truth Table Definition of the Operator *or*

Next we'll consider the *or* operator. We'll use the symbol $\vee$ for *or*, although many people use the | symbol instead. Here is the truth table for *or*:

| $p$ | $q$ | $p \vee q$ |
|---|---|---|
| $T$ | $T$ | $T$ |
| $T$ | $F$ | $T$ |
| $F$ | $T$ | $T$ |
| $F$ | $F$ | $F$ |

We read it just as we did the truth table for *and*. For example, the third line says, "When $p$ is false and $q$ is true, then $p \vee q$ is true". You can remember that $\vee$ stands for *or* just by remembering that it isn't *and*.

The operator that we have just defined is sometimes called ***inclusive-or***. So $p \vee q$ is true as long as *at least one* of $p$ and $q$ is true. This definition of *or* is what we want when we wish to say something like, "You can come to the prom if you are a member of our class *or* you are the guest of a member of our class." In other words, you can certainly come to the prom if you are both.

We could also define a version of *or* that is true in case *exactly one* of $p$ and $q$ is true. This version is called ***exclusive-or***. It's also useful. For example, it's what we want when we wish to say something like, "You can have a red hat *or* a blue one." It's also useful in computer circuit design. So we will define a separate operator (with a different symbol) for it later. For now, just remember that $p \vee q$ is always true except when both $p$ and $q$ are false

We'll call a statement that is formed by *or*-ing together other statements a ***disjunction***.

Some examples of simple disjunctions are

- $p \vee q$
- $(p \vee q) \vee r$

***English Aside***

In English, we can say, "*p* or *q*". We can also say, "either *p* or *q*." Sometimes, these two forms mean the same thing. For example, these two sentences are equivalent (meaning that if at least one of wind or rain is present, there won't be any helicopter flying):

- The helicopter won't fly in case of wind or rain.
- The helicopter won't fly in case of either wind or rain.

Sometimes, however, the word "either" is inserted specifically to indicate that an *exclusive-or* is intended. In other words, the combined statement is true only if exactly one, but not both, of *p* and *q* is true. For example, don't get your heart set on two desserts if you're told:

- You can have either cake or pie.

In cases like this, the logical operator $\vee$, as we have defined it, doesn't capture the intended meaning. To do that requires the separate *exclusive-or* operator that we'll define later.

Legal documents are supposed to be unambiguous. So this confusion between inclusive and exclusive *or* must be avoided. Thus legal documents often use one of these constructions:

- *p* and/or *q* (meaning *inclusive-or*, so it is possible that both are true)
- *p* or *q* but not both (meaning *exclusive-or*)

In this course, when we say simply, "or", we will always mean *inclusive-or*. We'll indicate *exclusive-or* explicitly when that's what we mean.

## Problems

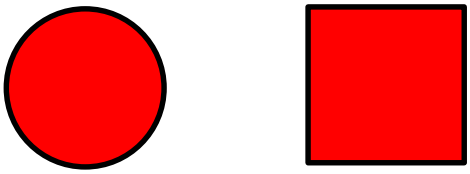
1. Recall the truth table for *or* that we just presented:

| $p$ | $q$ | $p \vee q$ |
|---|---|---|
| $T$ | $T$ | $T$ |
| $T$ | $F$ | $T$ |
| $F$ | $T$ | $T$ |
| $F$ | $F$ | $F$ |

Let $p$ correspond to the claim, “The circle is red.”
Let $q$ correspond to the claim, “The square is blue.”

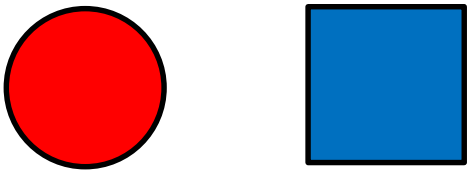
Consider the claim, “The circle is red or the square is blue.”

(Part 1) Consider this situation:



Which row of the *or* truth table tells us whether our claim, “The circle is red or the square is blue,” is true?

(Part 2) Consider this situation:



Which row of the *or* truth table tells us whether our claim, “The circle is red or the square is blue,” is true?

2. Let's practice understanding the distinction between *inclusive-or* and *exclusive-or*. Recall that:

- $p \vee q$ (*inclusive-or*) is True if and only if **at least one** of $P$, $q$ is true.
- $p$ XOR $q$ (one way to indicate *exclusive-or*) is True if **exactly one** of $p$, $q$ is true.

Assume that each of the following statements is reasonable in the world in which we live. Indicate whether *inclusive* or *exclusive* or is intended:

a) I will order chicken or fish for dinner tonight.
b) You'll like the concert if you're into percussion or you love everything Japanese.
c) You need to get a new license if yours is more than seven years old or lost.
d) Lightening or Black Thunder will win the race tomorrow.

3. Determine whether each of the following English statements is intended to be written using 'inclusive or' or 'exclusive or". Note that in English we use a shorthand. For example, the sentence, "My cell phone carrier is AT&T or T-Mobile," corresponds to the logical claim:

"My cell phone carrier is ATT" *or* "My cell phone carrier is T-Mobile"

In each of these cases, pick the most likely interpretation in the world in which we live.

a) My cell phone carrier is AT&T or T-Mobile.
b) To interview as a Software Engineer, you should be familiar with Java or C++.
c) I will buy my Calculus textbook from the University Co-op or Amazon.
d) "The Hunger Games" is available on Netflix or Amazon.
e) You can get your Bachelor's Degree or Master's Degree from UT.
f) You can procrastinate by going on Reddit or Facebook. (Inclusive)

## A Truth Table Definition of the Operator *not*

Next we'll define *not*. It is the only example we'll consider of a ***unary operator*** (i.e., it acts on a single operand). All of the other operators act on exactly two operands and, thus, are called ***binary operators***.

We will use $\neg$ as the symbol for *not*. (Notational aside: Sometimes people write $\sim p$ or $\overline{p}$. These mean the same thing as $\neg p$.)

Here is the truth table for *not*:

| $p$ | $\neg p$ |
|---|---|
| $T$ | $F$ |
| $F$ | $T$ |

Clearly *not* simply reverses the truth value of its operand. This corresponds to our everyday use of the word "not".

### *English Aside*

The concept of negation can be expressed in many ways in English. The most obvious is to use the word "not". But other words, such as "only" and "just" have implied negations. For example, if one says, "Of Joe, Bill, and Jim, only Bill is going to the store," what is meant is that Joe is *not* going to the store, Bill *is* going to the store, and Joe is *not* going to the store. This happens even though the word "not" did not appear anywhere in the sentence. Similarly, we could have said, "Of Joe, Bill, and Jim, just Bill is going to the store."

The words, "never", "nowhere", "none", "only", "just", "unless", "except", "without", and "but" all include implied negations. There are also prefixes that negate the property they precede. These include "un-", "il-", and "in-".

We've just seen that *not* reverses the truth value of its operand. Because there are only two truth values, if we apply *not* a second time, we'll get back the original value.

| $p$ | $\neg p$ | $\neg\neg p$ |
|---|---|---|
| $T$ | $F$ | $T$ |
| $F$ | $T$ | $F$ |

This observation will be the basis for the first of the Boolean identities that we'll present in the next chapter.

### *English Aside*

In logic, saying *not* more than once matters. If you say, "It is not the case that I am not going to the store", then you are going to the store.

There's a great example, in an episode of *Big Bang Theory*, of a quadruple negative that comes out positive.

Here's an example where one of the negatives is a prefix ("in-"). We actually heard this one on tv:

> The call on the [football] field is overturned only if the call is not incontrovertible.

In other words, the call is overturned only if it is controvertible.

But we should observe that some languages and some English dialects (including standard English up until about the 16th century) ignore all but the first instance of "not". For example, "I ain't seen nothing," generally means, "I didn't see anything". And we all know what, "Ain't Momma happy, ain't nobody happy," means. This usage is called a double negative.

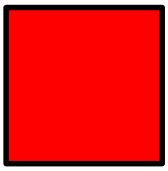
## Problems

1. Recall the truth table for *not* that we just presented:

| $p$ | $\neg p$ |
|---|---|
| $T$ | $F$ |
| $F$ | $T$ |

In both parts of this problem, let $p$ correspond to the claim, “The square is blue.”

(Part 1) Consider this situation:



Which row of the *not* truth table tells us whether the claim, “The square is not blue,” is true?

(Part 2) Consider this situation:

Which row of the *not* truth table tells us whether the claim, “The square is not blue,” is true?

2. Suppose that we want to represent the sentence, “Neither the King of France nor the Prince of Unicorns has red hair”.

Let $k$ correspond to the claim, “The King of France has red hair”.
Let $u$ correspond to the claim, “The Prince of Unicorns has red hair”.

Which of these logical statements corresponds to our claim:

a) $k \vee u$
b) $\neg k \vee \neg u$
c) $\neg k \wedge \neg u$
d) $\neg(k \wedge u)$

3. Suppose that I say, “No way am I not gonna come to the party.” Assuming that I speak standard English, which of these is true:

(a) I’m going to go to the party.
(b) I’m not going to go the party.

4. Consider the following quote from a newspaper story:

> The Supreme Court on Monday left intact a state court decision invalidating an Oklahoma law that effectively banned P.

Is P legal in Oklahoma?

### A Truth Table Definition of the Operator *is equivalent to*

Next we'll define the operator *is equivalent to*, for which we'll use the symbol ≡. (Notational aside: An alternative symbol is ↔. So $p \leftrightarrow q$ means the same thing as $p \equiv q$.)

Here is the truth table for *is equivalent to*:

| $p$ | $q$ | $p \equiv q$ |
|---|---|---|
| $T$ | $T$ | $T$ |
| $T$ | $F$ | $F$ |
| $F$ | $T$ | $F$ |
| $F$ | $F$ | $T$ |

This one is pretty simple: "$p \equiv q$" is true exactly when $p$ and $q$ have the same truth values.

It's important to note that "equivalent" and "identical" are different. The notion of equivalence that we want, and that we are capturing here, can be paraphrased as, "the same with respect to some particular purpose." And the purpose we care about is determining truth values. For example, we'll soon prove that *and* is commutative (in other words, it doesn't matter what order the operands to it are given). So we'll prove that the following two statements are equivalent (in the logical sense):

$r \wedge s$ $\qquad$ $s \wedge r$

Not identical, but equivalent.

Consider these two English sentences:

- The Morning Star is about the same size as the Earth.
- The Evening Star is about the same size as the Earth.

Astronomers now know that the Morning Star and the Evening Star refer to the same celestial object (namely Venus, which isn't actually a star at all). So these two sentences, while different, are equivalent. Either they are both true or they are both false.

***English Aside***

The statement, "*p* is equivalent to *q*", is quite often expressed as, "*p* if and only if *q*". A longer version is, "*p* is a necessary and sufficient condition for *q*". The usage of "if", "only", "necessary", and "sufficient" are better understood after we consider the meaning of the logical operator *implies*.

## Problems

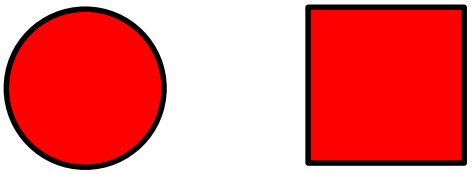
1. Recall the truth table for *is equivalent to* that we just presented:

| $p$ | $q$ | $p \equiv q$ |
|---|---|---|
| *T* | *T* | *T* |
| *T* | *F* | *F* |
| *F* | *T* | *F* |
| *F* | *F* | *T* |

Let $p$ correspond to the claim, "The circle is red."
Let $q$ correspond to the claim, "The square is blue."

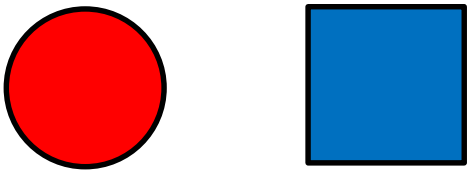
Consider the claim, "The circle is red is equivalent to the square is blue."

(Part 1) Consider this situation:



Which row of the *is equivalent to* truth table tells us whether our claim, "The circle is red is equivalent to the square is blue," is true?

(Part 2) Consider this situation:



Which row of the *is equivalent to* truth table tells us whether our claim, "The circle is red is equivalent to the square is blue," is true?

2. Let $p$ correspond to the claim: $x + 2 = 7$

Let $q$ correspond to the claim: $7 = 2 + x$

Assume the standard rules of algebra. True or false: $p \equiv q$

## A Truth Table Definition of the Operator *implies*

The next operator is *implies*. Of all the operators that we are defining, this is the one that may seem the least natural to you. But let's start with the definition that we are going to use. Then we'll look at why we have chosen to define *implies* in this way and how you can get your head around it.


https://www.youtube.com/watch?v=3RFamYOCEHA

Here is the truth table for *implies*, also called ***material implication***, (for which the symbol is $\rightarrow$):

| $p$ | $q$ | $p \rightarrow q$ |
|---|---|---|
| $T$ | $T$ | $T$ |
| $T$ | $F$ | $F$ |
| $F$ | $T$ | $T$ |
| $F$ | $F$ | $T$ |

When we have a statement of the form $p \rightarrow q$, it's common to call:

- $p$ the ***antecedent***, and
- $q$ the ***consequent***.

Back to the truth table. Here's perhaps the easiest way to interpret it. Suppose I assert, "$p \rightarrow q$". For example, suppose I claim, "rain $\rightarrow$ (implies) wet sidewalks". Under what circumstance(s) would you call me a liar? We are defining *implies* so that it returns False only in the case that I have lied. We are going to define it to return True in all other cases. So when have I lied? Surely if there is rain but not wet sidewalks, then I lied. Row 2 of the truth table describes that case. My claim is false. Are there any other cases in which you'd call me a liar? Surely an observation of rain and wet sidewalks (row 1) is consistent with my claim. But now suppose that there is no rain. Then you'll probably say that my claim is irrelevant. Okay. But not a lie, right? So we'll define (in rows 3 and 4) my claim to be true.

So we note that $p \rightarrow q$ is true in *every* situation except when (as shown in row 2) $p$ is true and $q$ is false.

Why is this definition so difficult for students (seeing it for the first time) to accept? Probably the biggest reason is that a very common interpretation of the English word "implies" is that there is some sort of causality. Thus, whenever we say, "$p$ implies $q$", many people read it as, "$p$ causes $q$". Think about the wet sidewalks example. Rain causes the sidewalks to be wet. When you think about it this way, the first two lines of our truth table make sense. The problem is the last two rows: it's hard to know what truth value we ought to assign when $p$ is false. The statement seems irrelevant. Causality is a very important phenomenon and we will want eventually to be able to reason about

it. But, precisely because it is important and also complex, we need to separate it from many simpler notions.

Just as we are asking you to accept the meaning of *or* to be the inclusive version (even though the English word "or" often means the exclusive version), we ask you to remove *all* notion of causality from our usage of the word "implies".

Suppose Jim is tall and Joe is short. Then, from the first line of the truth table, the statement, "Jim is tall implies Joe is short" is true (essentially because it isn't false). Yet, we are in no way suggesting that Jim's tallness causes Joe's shortness.

Meanwhile, keep in mind that assigning no truth value isn't a choice. We *must* assign some truth value to the expression, "*p* implies *q*", even when *p* is false and we may feel that the expression is irrelevant. If we don't, then "*p* implies *q*" won't be a logical statement (since a statement must have a truth value defined for all combinations of truth values of its constituents). Given that we're in the business of defining operators that we can use to build statements, that won't do. We must pick something. And the something we have picked is the "When have I lied?" criterion.

Another way to view our definition is this: Suppose that we want simply to claim that it's not possible to have *p* without also having *q* (for *whatever* reason). There may be no obvious linkage at all between *p* and *q* except what we've observed about them. Our definition of *implies* captures this simple notion. Look again at the truth table we have given. It tells us that the only way for "$p \rightarrow q$" to be false is that we do have *p* but do not have *q*.

Notice that, once we remove the notion of causality and look just at the simultaneous truth or falsity of two statements, it's possible to make a logical claim about two completely unrelated statements (silly as that may at first seem).

For example, in the world in which we live, the claim, "The capital of Texas is Dallas implies that hydrogen atoms have two protons," is true. Perhaps ridiculous. (Surely no one has ever said this before). But true. "The capital of Texas is Dallas," is false. So is, "Hydrogen atoms have two protons". So we use the last row of the truth table and determine that our claim is true. It could only be false if we lived in a world in which, "The capital of Texas is Dallas," were true but, "Hydrogen atoms have two protons," were still false. But we don't.

Let us give one final example that may help you accept as reasonable the definition that we have given for *implies*: Consider the statement:

"If x is greater than 1, then x is greater than 0."

Do you agree that this statement is true no matter what actual value x has? If you say "yes", then ask yourself, "What about when x is -1?". Both of the operand statements, "x is greater than 1" and "x is greater than 0," are false. Yet, you still believe that "If x is greater than 1, then x is greater than 0" is true. You believe it probably because you don't have an example of a value for x that makes it false. (Back to the "call me a liar" view of our definition.) In any case, we hope that you now have at least one situation in which you're okay with the fact that we've defined "$p \rightarrow q$" to be true whenever $p$ is false.

***Big Idea***

The logical operators mean what they mean because we have *defined* them to do so. We could have given other definitions. But we'll see, as this course progresses, that these definitions give us the basis for a useful approach to discerning truth.

Finally, we'll say simply that "$p \rightarrow q$" means what it means because we have defined it that way. We could possibly have avoided some of this discussion if we'd used the same truth table and claimed it as the definition for some logical operation *quilpled*. You'd then not have had any preconceived notion of what the definition ought to be. But trust us – we won't lead you astray. Using this as the definition of *implies* will actually cause no problems. In fact, it will turn out to be very useful. And, as all students have done in the past, you will soon be accepting it.

***English Aside***

English gives us several ways to describe the logical relation *implies*. For example, we can say:

- $p$ implies $q$. "Rain implies wet sidewalks."
- if $p$ then $q$. "If it's raining then the sidewalks will be wet."
- if $p$, $q$. "If it's raining, the sidewalks will be wet."
- $p$ only if $q$. "It's raining only if the sidewalks are wet."
- $q$ if $p$. "The sidewalks will be wet if it's raining."

## Problems

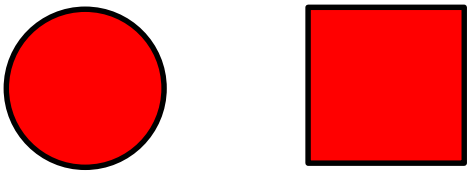
1. Recall the truth table for *implies* that we just presented (and argued hard for):

| *p* | *q* | *p*→*q* |
|---|---|---|
| *T* | *T* | *T* |
| *T* | *F* | *F* |
| *F* | *T* | *T* |
| *F* | *F* | *T* |

Let *p* correspond to the claim, "The circle is red."
Let *q* correspond to the claim, "The square is blue."

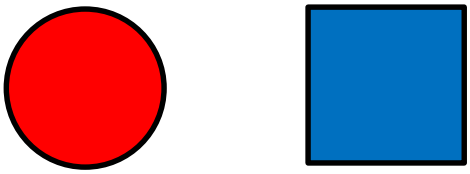
Consider the claim, "The circle is red implies the square is blue."

(Part 1) Consider this situation:



Which row of the *implies* truth table tells us whether our claim, "The circle is red implies the square is blue," is true?

(Part 2) Consider this situation:



Which row of the *implies* truth table tells us whether our claim, "The circle is red implies the square is blue," is true?

2. Assume the world in which we live. Mark each of the following statements True if it is in fact true, false otherwise:

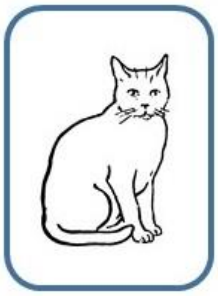
a) If 2 + 2 = 5 then 7 - 4 = 9.
b) If Snoopy is President then Lucy is Queen.
c) If Paris is in France then Italy is in Rome.
d) If the Colorado is a river then the Alamo is in Texas.
e) If 7 + 5 = 11 then Austin is not in Texas.

3. The ***Wason Selection Task*** is a logic puzzle that is widely used as a test of deductive reasoning. Click http://www.philosophyexperiments.com/wason/ for an explanation of the task and a chance to try your hand at three examples of it.

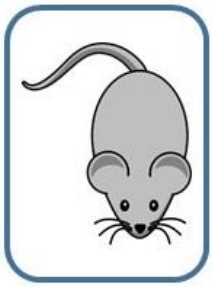
After you've done that, you should be able to solve this one. You are told that all cards you'll see have an animal on one side and a food on the other. You must check to see that the following rule has been observed:

> If a card has a mouse on one side, it must have cheese on the other side.

Here are the cards:

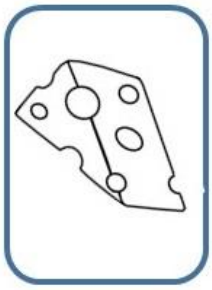


Card 1



Card 2

Card 3



Card 4

Which cards must be turned over to check that the rule has been followed?

**Summary – What is a Boolean Well-Formed Formula?**

Now that we have introduced the symbols that we will use to denote the standard logical operations, we can summarize the syntax of a Boolean logic statement, also called a Boolean well-formed formula, or wff, pronounced, "woof":

(1) True (usually abbreviated *T*) is a statement. It is always true.

(2) False (usually abbreviated *F*) is a statement. It is always false.

(3) Individual claims (which we may write in English or we may denote with variables like *p*) are statements. They may be either true or false.

(4) If *p* and *q* are statements, then so are:
   a) $\neg p$
   b) $p \wedge q$
   c) $p \vee q$
   d) $p \equiv q$
   e) $p \rightarrow q$

   In all of these cases, the truth value of the compound formula is given by their definitions as we've just presented them. Further, if we introduce additional operators (which we will do), then expressions formed using them as defined are also statements.

(5) If *p* is a statement, then so is (*p*).

Notice that when we say, "If *p* is a statement," we are using the symbol, "*p*," to stand for any expression (not solely the literal symbol, "*p*").

The following expressions are all statements:

| | |
|---|---|
| $T,\ p,\ p \wedge q$ | These are simple. |
| $\neg \neg p$ | $\neg p$ is an expression. So, using (4a) again and letting $p$ there correspond to $\neg p$, we have that $\neg \neg p$ is an expression. |
| $(p \wedge q) \rightarrow r$ | Notice here that $p \wedge q$ is an expression. Then so is $(p \wedge q)$. Then so is this entire expression. Think of $p$ in (4e) as corresponding to the entire statement $(p \wedge q)$. And $q$ in (4e) corresponds to $r$. |
| $G \vee H$ | We are not limited to the names $p$ and $q$. As we'll soon see, we may want to define many symbols to stand for the basic statements that we want to work with. |

## Problems

1. For each of the following expressions, mark True if it is a wff and False otherwise.

a) $(p \wedge q) \rightarrow (r \wedge q)$

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

c) $(p \neg q) \rightarrow \neg r$

d) $(((p \wedge q) \rightarrow r) \vee s)$

e) $((R \wedge D) \wedge G) \neg(D \vee R)$

f) $(((p \wedge q) \rightarrow r) \vee s$

## Larger Logical Expressions

### Building Truth Tables for More Complex Logical Expressions

If this were arithmetic, instead of logic, we would now know how to add, subtract, multiply, divide, and change the sign of a number. But we want to do more complex things (like adding three numbers). Similarly for logical expressions. We want to be able to combine multiple operators to form a single meaningful expression. To do this, we must be able to build more complex truth tables.

Let's start with a two-operator expression:

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

We want to build a truth table whose final column tells us the truth value of that expression given any set of truth values that may occur for the individual variables $p$ and $q$.

To see how to compute that column, we return to our analogy with arithmetic: If we wanted to evaluate (1+2)×3, we would start with the innermost expression and evaluate outward. So, in this arithmetic case, we'd start with (1+2). For our logic example, we'll start with $(p \wedge q)$. We will introduce "working columns" that specify the truth values for all the intermediate expressions. We'll be able to compute the final column by combining the working columns.

With one working column, our truth table looks just like the table for *and*:

| $p$ | $q$ | $p \wedge q$ |
|---|---|---|
| $T$ | $T$ | $T$ |
| $T$ | $F$ | $F$ |
| $F$ | $T$ | $F$ |
| $F$ | $F$ | $F$ |

Now we need to add a new column for $(p \wedge q) \vee p$:

| $p$ | $q$ | $p \wedge q$ | $(p \wedge q) \vee p$ |
|---|---|---|---|
| $T$ | $T$ | $T$ | |
| $T$ | $F$ | $F$ | |
| $F$ | $T$ | $F$ | |
| $F$ | $F$ | $F$ | |

The third column has the truth values that we just computed for $p \wedge q$. The first column still has the truth values for $p$. We need to *or* those two columns together to create the values for the new fourth column. We'll highlight the columns that we're working with in red.

Starting with the first row, we see that $p \wedge q$ is *T* and *p* is *T*. The truth table for the *or* operator tells us that $T \vee T$ is *T*, so the first entry in the fourth column should be *T*:

| *p* | *q* | $p \wedge q$ | $(p \wedge q) \vee p$ |
|---|---|---|---|
| *T* | *T* | *T* | *T* |
| *T* | *F* | *F* | |
| *F* | *T* | *F* | |
| *F* | *F* | *F* | |

In the second row, we see that $p \wedge q$ is *F* and *p* is *T*. The truth table for the *or* operator tells us that $F \vee T$ is *T*, so the second entry in the fourth column should be *T*:

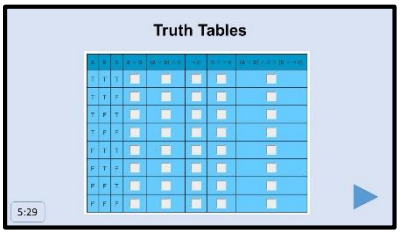
| *p* | *q* | $p \wedge q$ | $(p \wedge q) \vee p$ |
|---|---|---|---|
| *T* | *T* | *T* | *T* |
| *T* | *F* | *F* | *T* |
| *F* | *T* | *F* | |
| *F* | *F* | *F* | |

In the third row, we must compute $F \vee F$, which is *F*. And for the last row, we must do the same thing. So the complete truth table for the expression $(p \wedge q) \vee p$ is

| *p* | *q* | $p \wedge q$ | $(p \wedge q) \vee p$ |
|---|---|---|---|
| *T* | *T* | *T* | *T* |
| *T* | *F* | *F* | *T* |
| *F* | *T* | *F* | *F* |
| *F* | *F* | *F* | *F* |

## The Truth Table App

We have built a Truth Table app that you can use to practice building truth tables. Watch the video to see a demo. Then, to try it out, go to www.truthtables.org.




https://www.youtube.com/watch?v=P6ZNfxIyHkQ

There are two ways to use the app:

- You can go to the website and then enter a Boolean expression, using the symbols listed in the table that you'll find there.

- You can embed a Boolean expression into a url. For example, consider this url:

  www.truthtables.org/#/true/(p|!p)->!(q|!q)

  It will initialize a truth table for the expression $(p \vee \neg p) \rightarrow \neg(q \vee \neg q)$.

  We'll use this mode to set up problems for you to solve.

You will notice, when you click in a cell, that the cell(s) on which your current cell depends will be highlighted in blue.

The app can run in either of two modes:

- Training wheels: Every time you enter a value, it will be highlighted in green if it's correct, red if it's incorrect. This is the default mode if you just start at the app's website and enter an expression. And it's what you'll get if the first argument you send it in a url is "true" (as in our example above).

- Go for broke: No feedback. You can cascade possibly incorrect values to your heart's content. Set the first parameter to "false" to use this mode.

## Problems

1. Fill in a truth table for the expression: $((p \wedge q) \rightarrow \neg p) \vee q$

2. Fill in a truth table for the expression: $\neg ((q \vee r) \wedge \neg q)$

3. Fill in a truth table for tĵhe expression: $((p \rightarrow q) \vee r) \wedge \neg p$

## Truth Tables with Three or More Variables

In the previous section, we saw examples of:

One-variable truth tables:

| $p$ | $\neg p$ |
|---|---|
| $T$ | $F$ |
| $F$ | $T$ |

| $p$ | $q$ | $p \wedge q$ |
|---|---|---|
| $T$ | $T$ | $T$ |
| $T$ | $F$ | $F$ |
| $F$ | $T$ | $F$ |
| $F$ | $F$ | $F$ |

Two-variable truth tables:

Not counting the title row, how many rows will there be in a truth table with $n$ variables?

Let's make a chart:

| Number of variables | Number of rows | As powers of 2 |
|---|---|---|
| 1 | 2 | $2^1$ |
| 2 | 4 | $2^2$ |
| 3 | ? | ? |
| … | … | … |
| $n$ | ? | ? |

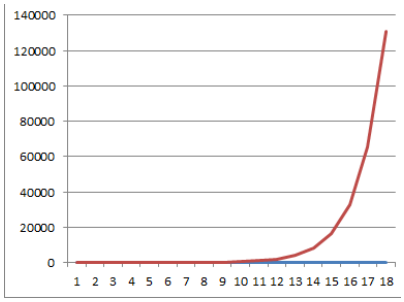
See if you can figure out values for the question marks. Then continue to the next page.

## The Size of the Truth Table Grows Quickly

Here's the filled in table:

| Number of variables | Number of rows | As powers of 2 |
|---|---|---|
| 1 | 2 | $2^1$ |
| 2 | 4 | $2^2$ |
| 3 | 8 | $2^3$ |
| … | … | … |
| $n$ | $2^n$ | $2^n$ |

The number of rows doubles each time we add a new variable since we must, for each combination of all of the earlier variables, now allow for the new variable to take on the value *T* and also for it to take on the value *F*. This graph of the powers of 2 shows how quickly the number of rows grows once the number of variables goes above 15 or so.




***Big Idea***

Truth tables can get very big very fast.

Let's do an example. We'll build a truth table for the following three-variable expression:

$$(p \wedge \neg q) \rightarrow \neg r$$

The first three columns of the truth table will be labeled *p*, *q*, and *r*. Just as we did when we only had two variables, we need to specify every possible combination of truth values. The order in which we write the truth values in our table doesn't matter. But, particularly as we move to larger numbers of variables, we need to write them systematically so that we're sure of getting them all. We can generalize the system that we have used for the two variable case. Begin with the column corresponding to the first variable (in this case, *p*). Enter *T* in the first half of the rows and *F* in the second half of them. At this point, we have:

| $p$ | $q$ | $r$ |
|---|---|---|
| $T$ | | |
| $T$ | | |
| $T$ | | |
| $T$ | | |
| $F$ | | |
| $F$ | | |
| $F$ | | |
| $F$ | | |

Then we move to the column corresponding to the second variable (i.e., $q$). Enter $T$ in the first quarter of the rows, $F$ in the second quarter, $T$ in the third quarter and $F$ in the last quarter. At this point, we have:

| $p$ | $q$ | $r$ |
|---|---|---|
| $T$ | $T$ | |
| $T$ | $T$ | |
| $T$ | $F$ | |
| $T$ | $F$ | |
| $F$ | $T$ | |
| $F$ | $T$ | |
| $F$ | $F$ | |
| $F$ | $F$ | |

Finally, we move to the column corresponding to the third (and last) variable (i.e., $r$). Enter $T$ in the first eighth of the rows (which, in the case of three variables, is exactly one row), $F$ in the next eighth, $T$ in the next eighth, and so forth, alternating values. At this point, we have:

| $p$ | $q$ | $r$ |
|---|---|---|
| $T$ | $T$ | $T$ |
| $T$ | $T$ | $F$ |
| $T$ | $F$ | $T$ |
| $T$ | $F$ | $F$ |
| $F$ | $T$ | $T$ |
| $F$ | $T$ | $F$ |
| $F$ | $F$ | $T$ |
| $F$ | $F$ | $F$ |

As we predicted, there are exactly eight rows.

Recall that we are trying to build a truth table for the expression $(p \wedge \neg q) \rightarrow \neg r$. So we will want a final column that tells us the truth value of that expression given any set of truth values that may occur for the individual variables $p$, $q$, and $r$. Thus the final column's title will be $(p \wedge \neg q) \rightarrow \neg r$. To compute the values for that column, we'll do the same thing we did in the case of complex statements involving just two variables: We'll introduce working columns that specify the truth values for the intermediate

expressions. We'll then compute the final answer by using those working columns. Here's what we get when we do that:

| $p$ | $q$ | $r$ | $\neg q$ | $p \wedge \neg q$ | $\neg r$ | $(p \wedge \neg q) \rightarrow \neg r$ |
|---|---|---|---|---|---|---|
| *T* | *T* | *T* | *F* | *F* | *F* | *T* |
| *T* | *T* | *F* | *F* | *F* | *T* | *T* |
| *T* | *F* | *T* | *T* | *T* | *F* | *F* |
| *T* | *F* | *F* | *T* | *T* | *T* | *T* |
| *F* | *T* | *T* | *F* | *F* | *F* | *T* |
| *F* | *T* | *F* | *F* | *F* | *T* | *T* |
| *F* | *F* | *T* | *T* | *F* | *F* | *T* |
| *F* | *F* | *F* | *T* | *F* | *T* | *T* |

We notice that the expression that we're working with is almost (but not quite) always true. It is false in exactly one case: $p$ is true, $q$ is false, and $r$ is true .

## Problems

1. Build a truth table for the expression: $(r \wedge (p \vee q)) \rightarrow \neg(r \vee p)$.

## Problems That Explore Operator Precedence

1. Suppose that you type the following arithmetic expression into Excel or a simple calculator app such as this one.

    3*12+8/4

(Part 1) What answer will you get:

(a) 15
(b) 42
(c) 38
(d) 11

(Part 2) In what order were the operators performed to get that answer?

(a) Multiply 3*12, then add 8, then divide by 4.
(b) Add 12 and 8, then multiply by 3, then divide by 4.
(c) Divide 8 by 4, add that to 12, then multiply by 3.
(d) Multiply 3 by 12. Divide 8 by 4. Add those two numbers

2. Suppose that you type the following arithmetic expression into Excel or a simple calculator app such as this one:

    12 + 16 * 2 + 1

(Part 1) What answer will you get:

(a) 57
(b) 45
(c) 51
(d) 84

(Part 2) In what order were the operators performed to get that answer?

(a) Multiply 16*2, then add that to 12. Finally, add 1.
(b) Add 12 and 16, then multiply by 2, then add 1.
(c) Add 2 and 1. Multiply 16 by that, then add to 12.
(d) Multiply 3 by 12. Divide 8 by 4. Add those two numbers.

## Operator Precedence

The experiment that we've just done tells us that we've agreed to define an order in which the various arithmetic operations will be performed.

We call this order ***operator precedence***. Higher precedence operators are done before lower precedence ones. The standard arithmetic operators are often arranged in the following order (although there is some disagreement about whether unary minus or exponentiation is done first):

| | | |
|---|---|---|
| Highest | *unary minus* | Done first |
| | *exponentiation* | |
| | *multiplication, division* | |
| Lowest | *addition, subtraction* | Done last |

## Operator Precedence for Logical Operators

Of course, the reason we're talking about operator precedence is not that we care to review what we know about arithmetic. What we do need to do is to define an analogous precedence scheme for logical operators. Here's the one that we will use:

| | **Arithmetic** | **Logic** |
|---|---|---|
| Highest: | *unary minus* | *not* |
| | *exponentiation* | *and* |
| | *multiplication* and *division* | *or* |
| | *addition* and *subtraction* | *implies* |
| Lowest: | | *is equivalent to* |

Recall that precedence is not the only tool we have for indicating the order in which operations should be performed. We can always use parentheses, either when the defined precedence levels don't lead to the interpretation that we want or when we're not sure.

In fact, we'll typically not count on you to remember this list. We'll assume that *not* (our only unary operator) is done first (it's got the tightest scope). For everything else, we'll usually use parentheses.

Here are some pairs of equivalent expressions:

| | | |
|---|---|---|
| $p \wedge \neg p$ | $p \wedge (\neg p)$ | $\neg$ is done before $\wedge$. |
| $p \wedge q \vee r$ | $(p \wedge q) \vee r$ | $\wedge$ is done before $\vee$. |
| $p \vee q \wedge r$ | $p \vee (q \wedge r)$ | again, $\wedge$ is done before $\vee$. |
| $\neg p \vee q \wedge r$ | $(\neg p) \vee (q \wedge r)$ | $\neg$ , then $\wedge$, then $\vee$. |
| $p \vee q \wedge r \rightarrow s$ | $(p \vee (q \wedge r)) \rightarrow s$ | $\rightarrow$ last. |

What happens when an expression contains two or more operators with the same precedence? The answer (by definition) is that the operations will be performed left to right.

So here are some more pairs of equivalent expressions:

| | |
|---|---|
| $p \wedge q \wedge r$ | $(p \wedge q) \wedge r$ |
| $p \rightarrow q \rightarrow r$ | $(p \rightarrow q) \rightarrow r$ |
| $p \rightarrow q \rightarrow r \wedge p$ | $(p \rightarrow q) \rightarrow (r \wedge p)$ |

In the case of logical expressions, the only way to have equal precedence operators is to have identical ones. (Note that this is true in the above examples.) And it turns out that all of our binary logical operators ($\wedge$, $\vee$, $\rightarrow$, and $\equiv$) are associative. In other words, we'll get the same result however we associate them. We'll prove this claim later. But if you can't wait, go ahead and prove (by showing that they have identical truth tables), one of the cases:

$((p \wedge q) \wedge r)$ is equivalent to $(p \wedge (q \wedge r))$

## Problems

1. Which of the following is equivalent to: $\neg p \vee q \rightarrow r$?

(a) $\neg(p \vee q \rightarrow r)$
(b) $\neg(p \vee q) \rightarrow r$
(c) $\neg p \vee (q \rightarrow r)$
(d) $((\neg p) \vee q) \rightarrow r$
(e) $(\neg p) \vee (q \rightarrow r)$

2. Which of the following is equivalent to: $q \rightarrow r \wedge p \vee s$?

(a) $(q \rightarrow (r \wedge p)) \vee s$
(b) $q \rightarrow ((r \wedge p) \vee s)$
(c) $q \rightarrow (r \wedge (p \vee s))$
(d) $(q \rightarrow r) \wedge (p \vee s)$
(e) $( (q \rightarrow r) \wedge p) \vee s$

## Boolean Expressions in Programming

### Controlling Execution

Programs without Boolean expressions are like drivers with their eyes closed. Some things may be possible, but without being able to check the environment and respond to it, nothing like the real power of a computer, or a car, can be exploited.

The following driving procedure contains no explicit Boolean expressions (although, hidden inside some of these commands there probably are such expressions):

```
Depress accelerator.
Drive for two minutes.
Stop.
```

Not very useful. Here's a more effective procedure:

```
Drive until AtElmStreet.
If LightIsRed OR LightIsYellow:
        Stop.
        Wait until LightIsGreen.
Turn right.
Drive until AtPetesHouse.
Stop.
```

We've indicated the Boolean expressions in bold. There are five variables (AtElmStreet, LightIsRed, LightIsYellow, LightIsGreen, and AtPetesHouse). Each of them will be either true or false. We've also used the OR operator to combine two of them to describe the conditions under which we should stop.

Boolean expressions evaluate to True or False. So they are exactly what we need when we want to describe a procedure that checks relevant conditions and branches based on what it discovers. Once you start to notice this, you'll see Boolean expressions everywhere.

## Problems

1. Consider the following procedure for making whipped cream:

```
Open cream container.
If smell knocks you over:
        Stop.
        Choose a different dessert.  Exit.
Pour cream into bowl.
Beat until peaks form.
Taste.
If not sweet enough:
        Add sugar.
        Beat a bit more.
```

Mark the Boolean expressions in this procedure. (Hint: Look for anything that will evaluate to either True or False). How many are there:

a) 1
b) 2
c) 3
d) 4
e) 5

2. Consider the following procedure for doing the dishes:

```
Until no dirty dishes, do the following:
        Pick up a dish.
        If it is dishwasher safe, put it in the dishwasher.
        Otherwise, wash it by hand.
Run the dishwasher.
When the dishwasher has finished:
        Unload and put away the dishes.
```

Mark the Boolean expressions in this procedure. (Hint: Look for anything that will evaluate to either True or False). How many are there:

a) 1
b) 2
c) 3
d) 4
e) 5

## Boolean Expressions in Programming Languages

So now you see that you've been using Boolean expressions since about the time you could walk. When we talk to each other, we can be very flexible in how we state them (and all the other expressions that we use).

When we write code, however, we have to be more careful. We have to stick to the rules of the programming language that we're using. Every programming language (since the very first one) provides one or more (typically more) ways that programmers can use Boolean expressions to describe the sequences of operations that their programs should perform.

Here's a really simple Python program. Assume that `NOW` is the current year and `BIRTHYEAR` is the year in which you were born. Then this program, which could be part of the self-checkout procedure at your local convenience store, runs whenever you attempt to buy alcohol:

```
If NOW - BIRTHYEAR > 20:
        print("Yea!  Buy all you want.")
    else:
        print("Sigh.  Can I interest you in some water?")
```

This program contains one Boolean expression, `NOW - BIRTHYEAR > 20`. When the program runs, the Boolean expression is evaluated. If it's true (in other words, if the difference between the current year and the year in which you were born is greater than 20), then the first print statement will be executed. Else, if it's not, the second one will be. Notice that this is an oversimplification of the real rules, which must look at what day it is. But you get the idea.

More complex Boolean expressions are generally allowed. We'll revisit this issue after we've developed some tools for working with compound expressions.

## Problems

1. Consider the following program:

```
read(x, y)
if x = 0 or y = 0:
   print ("error")
else if x > y:
   print (x * y)
else:
   print (x + y)
```

Indicate, for each of the following program fragments, whether or not it is a Boolean expression (i.e., something with a truth value):

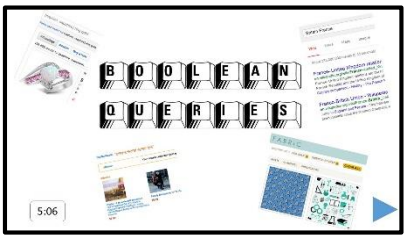
a) `x = 0`
b) `x = 0 or y = 0`
c) `x - y`
d) `print (x - y)`
e) `x > y`
f) `x + y`

## Boolean Queries

### What's a Boolean Query?

We've been thinking of logical expressions as statements that can be true or false. And we have been writing truth tables to describe the situations in which they take on one or the other of those values.

A slightly different way to think of them is as commands: Find me situations that match my expression (in other words, situations about which my expression is true).




We call commands like this ***queries*** and you've probably already used several of them just today.

https://www.youtube.com/watch?v=PanCqazPDcw

For example, suppose I enter this query into my Google search bar:

(chocolate pie) OR nuts

I expect Google to *make my statement true* by returning pages that match it. (Of course, there's more to Google's search algorithm than this; it also uses popularity and other factors. But a first approximation to what it should do is to find pages whose content matches what I've asked for.)

Lots of apps and web sites support Boolean queries. Most support three Boolean operators:

- *And*, which is generally the default operator. We don't write it explicitly at all. So, for example, suppose I type this query into Google:

  chocolate pie

  The search engine will return pages that contain both words, "chocolate" and "pie".

- *Or*. Google uses the keyword OR (it must be upper case). Ebay uses commas, which must separate terms enclosed in parentheses. For example, we could ask for rings with gold or silver by writing:

  ring (gold, silver)

- *Not*. Not is useful as a way to pare down the number of records that are returned. The most common way to indicate it is with a minus sign. In one experiment, the following ebay query returned 72,046 items:

Winnie the Pooh

But let's say I'm a purist. This ebay query returned only 33,477 items:

Winnie the Pooh -Disney

Most search engines don't support complex (arbitrarily nested) Boolean expressions. You can experiment with individual ones to see what they allow.

If you do want to experiment, we've found that it's much easier to see what's going on in ebay than it is in Google. That's because Google uses so many extra factors in deciding what pages to return. Ebay, on the other hand, will take you literally and only return items that actually match your query (or some simple variants like singular/plural forms).

For the next set of problems, use ebay because it's much easier there, than in Google or some other search engine, to see what effect changing a query has.

## Problems

1. Suppose that you want to find memorabilia from Carnegie-Mellon University (previously known as Carnegie Tech and sometimes called just Carnegie).

(Part 1) Go to ebay and type in the query, Carnegie. Jot down how many hits you got.

(Part 2) Now start using the minus operator to get rid of things that have nothing to do with the school. Are there a few words that, if ruled out, pare down the search a lot?

(Part 3) Suppose that you wrote a query such as:

Carnegie –purple –hamburger

(Of course, that's not one you'd write, but we want to talk about something of that form, so this will do.)

Which of the following Boolean expressions describes all the results that that query will return:

I. Carnegie $\wedge$ $\neg$purple $\wedge$ $\neg$hamburger
II. Carnegie $\wedge$ $\neg$(purple $\wedge$ hamburger)
III. Carnegie $\wedge$ $\neg$(purple $\vee$ hamburger)
IV. $\neg$(purple $\vee$ hamburger) $\wedge$ Carnegie

a) None of them.
b) Just one of them.
c) Just two of them.
d) Just three of them.
e) All four of them.

# English into Logic

## Boolean Logic Isn't English

English is a much more expressive language than Boolean logic is. The meanings of most English sentences can't be captured naturally in Boolean logic. For example:

(1) "All powers of 10 are even numbers."

(2) "Most English sentences are hard to represent exactly in Boolean logic."

(3) "I think that you're lying."

(4) "After you eat your dinner you can have dessert."

In the first of these sentences, the problem is that we want to make a single claim but have it apply to an infinite set of objects. There's a straightforward way to solve this problem; we'll see it when we move on to predicate logic and introduce quantifiers. The problems posed by the other examples are harder and require additional mechanisms that are mostly beyond what we'll be able to cover in this class.

So, if Boolean logic can't solve all of our problems, why are we spending so much time on it? Two reasons:

- It does solve a lot of practical problems. We've mentioned circuit design and search engine querying. We're about to look at some more examples.

- It is the basis for extensions that solve some harder problems. So we need to understand it before we can move on to them.

## Practice Converting Between English and Boolean Logic

To see how Boolean logic can be used, we should practice converting simple ideas into it.

Consider the sentence, "If it's cloudy, you can't see the stars." To convert this sentence to Boolean logic, we begin by assigning simple variable names to the basic statements. Then we can combine them. We'll give names to the following statements:

*C*: It's cloudy.
*S*: You can see the stars.

Then we have: $C \rightarrow \neg S$

Consider the sentence, "If there aren't any cookies, we'll have to buy or steal some." We'll give names to the following statements:

*C*: There are cookies.
*B*: We have to buy cookies.
*S*: We have to steal cookies.

Then we have: $\neg C \rightarrow (B \vee S)$

We don't actually need the parentheses here. Operator precedence will get us the interpretation that we want. But it's always safe to use them.

Consider the sentence, "We'll only have the party if it's not raining or snowing." We can give names to the following statements:

*R*: It's raining
*S*: It's snowing.
*P*: We'll have the party.

This one is more complex. Literally, this sentence just gives circumstances under which there will be no party. But most people interpret it as giving not just necessary conditions for partying but also sufficient ones. You would think me annoying, at the least, if I said this and I had no intention of having a party under any circumstances. So let's assume that this sentence says that we will have the party if the weather is good but we won't if it isn't. In other words, good weather (i.e., not rain and not snow) means party; bad weather means no party.

So we have: $(\neg R \wedge \neg S) \equiv P$

***English Aside***

In this last example, we exploited a convention for how English is used for efficient communication. We're generally expected not to mislead. Rather, we should make the strongest statement that we believe to be both true and relevant. Our listeners, then, can assume that if we didn't make a stronger statement it's because such a statement wouldn't be true. The Cooperative Principle describes a more general pattern of communication conventions that work in this way. A study of it is beyond the scope of this class, but we'll mention it occasionally. For more information, see:

http://en.wikipedia.org/wiki/Cooperative_principle.

We should point out here that there are other, logically equivalent ways of writing many complex logical statements. If you came up with something different for one of these examples, you can see whether your answer is equivalent to ours by writing out the truth tables for both our expression and yours and seeing if they're the same.

## Problems

1. Consider this sentence:

> "If a loaf of bread costs $4 and a pound of hamburger costs $8 then John eats tofu."

Define:

*B*: A loaf of bread costs $4.
*H*: A pound of hamburger costs $8.
*J*: John eats tofu.

Which of the following logical expressions captures the meaning of the sentence:

a) $B \rightarrow \neg H$
b) $(B \wedge H) \rightarrow J$
c) $\neg J \wedge \neg H \vee B$
d) $(B \wedge H) \wedge J$

2. Consider this sentence:

> "We'll have chips but no peanuts at the party."

Define:

*C* : We'll have chips at the party.
*P*: We'll have peanuts at the party.

Which of the following logical expressions captures the meaning of the sentence:

a) $C \rightarrow \neg P$
b) $C \wedge \neg P$
c) $\neg(C \wedge P)$
d) $C \vee \neg P$

3. Consider this sentence:

> "If you've got a costume, you can come to the party unless you like loud music."

Define:

*C*: You've got a costume.
*P*: You can come to the party.
*M*: You like loud music.

Which of the following logical expressions captures the meaning of the sentence:

a) $(C \rightarrow P) \wedge (\neg M \rightarrow P)$
b) $(C \rightarrow P) \vee (\neg M \rightarrow P)$
c) $(M \rightarrow P) \vee \neg C \vee \neg P$
d) $(C \wedge \neg M) \rightarrow P$
e) $\neg M \vee C \vee P$

4. Define:

*P*: Pat is on vacation.
*C*: Chris is on vacation.

(Part 1) Which of the following logical expressions corresponds to the sentence:

> "Pat and Chris are not both on vacation."

a) $\neg P \wedge \neg C$
b) $P \neg\wedge C$
c) $\neg(P \vee C)$
d) $\neg(P \wedge C)$
e) $\neg P \rightarrow \neg C$

(Part 2) Which of the following logical expressions corresponds to the sentence:

> "Pat and Chris are both not on vacation."

a) $\neg(P \wedge C)$
b) $\neg P \wedge \neg C$
c) $P \neg\wedge C$
d) $\neg(P \equiv C)$
e) $\neg P \vee \neg C$

## Validity, Satisfiability, Contradiction and Counterexamples

### Introduction

After we build a truth table for a logical expression, we can read off the values in its last column. It is that column that tells us the truth values for the entire expression as a function of the truth values of its variables. Think of each possible set of input values (represented as a single row) as a "possible world" that might occur. Then the whole column gives a picture of what might happen given any particular world in which we might find ourselves. All sequences of values are possible in that last column. But a few are particularly significant and we give names to expressions that exhibit those values.

Consider the following possible final columns for a truth table with three variables:

| |
|---|
| *T* |
| *T* |
| *T* |
| *T* |
| *T* |
| *T* |
| *T* |
| *T* |

All values are *T*.

| |
|---|
| *T* |
| *F* |
| *F* |
| *T* |
| *F* |
| *T* |
| *F* |
| *T* |

At least one value is *T*.

| |
|---|
| *F* |
| *F* |
| *F* |
| *F* |
| *F* |
| *F* |
| *F* |
| *F* |

All values are *F*.

When all values are *T*, we'll say that the corresponding formula is a ***tautology*** or that it is ***valid***. In other words, the formula will always be true, *no matter what*.

When at least one value is *T*, we'll say that the corresponding formula is ***satisfiable***. In other words, there's some circumstance(s) in which it may be true.

When all values are *F*, we'll say that the corresponding formula is a ***contradiction***. We'll also say that it is ***unsatisfiable***. In other words, "no way".

## Tautologies

Recall this truth table that we built for the expression $(p \wedge q) \vee p$:

| $p$ | $q$ | $p \wedge q$ | $(p \wedge q) \vee p$ |
|---|---|---|---|
| T | T | T | T |
| T | F | F | T |
| F | T | F | F |
| F | F | F | F |

Stare at it for a moment. Do you see that the $(p \wedge q) \vee p$ column is identical to the $p$ column? Remembering the notion of *is equivalent to*, can we claim that $(p \wedge q) \vee p$ is equivalent to $p$? Let's build a truth table for $((p \wedge q) \vee p) \equiv p$. It begins, just like the one above:

| $p$ | $q$ | $p \wedge q$ | $(p \wedge q) \vee p$ | $((p \wedge q) \vee p) \equiv p$ |
|---|---|---|---|---|
| T | T | T | T | |
| T | F | F | T | |
| F | T | F | F | |
| F | F | F | F | |

We obtain the four entries for the final column by consulting the truth table for *is equivalent to*:

| $p$ | $q$ | $p \equiv q$ |
|---|---|---|
| T | T | T |
| T | F | F |
| F | T | F |
| F | F | T |

So our final table is

| $p$ | $q$ | $p \wedge q$ | $(p \wedge q) \vee p$ | $((p \wedge q) \vee p) \equiv p$ |
|---|---|---|---|---|
| T | T | T | T | T |
| T | F | F | T | T |
| F | T | F | F | T |
| F | F | F | F | T |

Now notice the final column has all *T*'s. That means that whatever the truth values for $p$ and $q$, $((p \wedge q) \vee p) \equiv p$ must be true.

Recall that an expression that is true for all possible truth values of its variables is called a ***tautology***. We'll also say that such an expression is ***valid***. Tautologies correspond to our everyday idea of "true statements". In other words, they correspond to claims that are guaranteed to be true, in all cases.

$((p \wedge q) \vee p) \equiv p$ is the first expression that we have proved to be a tautology. We will soon prove many more.

## Problems

1. Indicate, for each of the following expressions, whether or not it is a tautology. (Hint: write out the truth table if you're not sure.)

a) $p \vee \neg q$
b) $p \vee \neg p$
c) $\neg p \rightarrow p$
d) $(p \wedge \neg p) \rightarrow q$

## Satisfiability

Let's return to the truth table for $(p \wedge q) \vee p$:

| $p$ | $q$ | $p \wedge q$ | $(p \wedge q) \vee p$ |
|---|---|---|---|
| *T* | *T* | *T* | *T* |
| *T* | *F* | *F* | *T* |
| *F* | *T* | *F* | *F* |
| *F* | *F* | *F* | *F* |

Here we see that the truth value of the expression $(p \wedge q) \vee p$ depends upon the truth values of the variables: it is neither always true nor always false.

We will say that expressions like this are ***satisfiable***. In other words, there is *some* way to satisfy them by making them true. If we are asked, "For what values is $(p \wedge q) \vee p$ satisfied?" we could answer, "For $p$ being true and $q$ being true." We could also answer, "For $p$ being true and $q$ being false." In fact, we can omit any mention of $q$ and simply say, "Whenever $p$ is true."

Proving that an expression is satisfiable is straightforward. We can simply build a truth table for it. If at least one row of the final column is *T*, then the expression can be made true somehow and thus it is satisfiable.

The problem of determining whether a formula is satisfiable is often nicknamed the ***SAT problem***. It turns out to play an important role in a lot of computing applications, including, for example, verifying that computer circuits do what they are supposed to do. You might then say, "Wonderful. We have an important problem and we have a simple way to solve it. All we have to do is build a truth table."

The difficulty is that the real world problems that we'd like to solve this way may involve hundreds of thousands of variables. And recall that the number of rows in a truth table of $n$ variables is $2^n$. Oops. If we really want to solve those problems, we'll need to be a lot cleverer. Because of the importance of doing so, a lot of effort gets devoted to finding exactly such clever techniques. Just as a quick measure of this, we'll point out that there's an important world-wide conference devoted to it every year.

> ***Nifty Aside***
>
> We could say a lot more about the SAT problem.

## Problems

1. Consider the following statement:

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

Prove that it is satisfiable. Construct its truth table. Indicate which values for $p$ and $q$ prove its satisfiability:

(a) $p$ is $T$ and $q$ is $T$.
(b) $p$ is $F$ and $q$ is $T$.
(c) $p$ is $T$ and $q$ is $T$.
(d) $p$ is $F$ and $q$ is $F$.
(e) There is more than one assignment that proves satisfiability.

2. Consider the following statements:

I. $(p \vee \neg q) \wedge \neg p$
II. $\neg(r \vee s) \wedge r$
III. $\neg(r \vee s) \vee r$

Which of them is/are satisfiable?

(a) Just I.
(b) Just II.
(c) Just III.
(d) At least two of them.
(e) None of them.

## Contradictions

Now consider this expression:

$$p \wedge \neg p$$

Since there is only one variable, the truth table has only two rows:

| $p$ | $\neg p$ | $p \wedge \neg p$ |
|---|---|---|
| $T$ | $F$ | $F$ |
| $F$ | $T$ | $F$ |

Now notice that the final column has all $F$'s. That means that, whatever the truth value for $p$, $p \wedge \neg p$ must be false. An expression that is false for all possible truth values of its variables is called a ***contradiction***. So we'll say that a contradiction is ***unsatisfiable***. Think of a contradiction as a statement that describes an impossible situation. Another way to think of it is that it describes a situation we'll never have to worry about. We've now proved our first such contradiction.

Here's another contradiction:

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

Here's its truth table:

| $p$ | $q$ | $\neg p$ | $p \vee \neg p$ | $\neg q$ | $q \vee \neg q$ | $\neg(q \vee \neg q)$ | $(p \vee \neg p) \rightarrow \neg(q \vee \neg q)$ |
|---|---|---|---|---|---|---|---|
| $T$ | $T$ | $F$ | $T$ | $F$ | $T$ | $F$ | $F$ |
| $T$ | $F$ | $F$ | $T$ | $T$ | $T$ | $F$ | $F$ |
| $F$ | $T$ | $T$ | $T$ | $F$ | $T$ | $F$ | $F$ |
| $F$ | $F$ | $T$ | $T$ | $T$ | $T$ | $F$ | $F$ |

Again, notice that the final column has all $F$'s. There is no circumstance in which this statement is true.

## Problems

1. Which of the following statements is a contradiction:
   (a) $(p \vee q) \rightarrow \neg p$
   (b) $\neg(p \wedge r) \rightarrow \neg r$
   (c) $(p \vee \neg p) \rightarrow \neg(p \vee \neg p)$
   (d) $\neg q \rightarrow \neg(\neg q \wedge \neg r)$

2. Which of the following statements is a contradiction:

(a) $(q \wedge r \wedge s) \rightarrow \neg s$
(b) $((q \vee r) \wedge s) \wedge \neg(q \vee r \vee s)$
(c) $(a \wedge b) \vee (\neg a \vee \neg\, b)$
(d) $(p \vee r) \rightarrow (p \wedge r \wedge s)$

3. Consider the following statements:

I. $(p \wedge q) \wedge q$
II. $\neg(\, p \wedge q) \vee q$
III. $(\, p \rightarrow q) \wedge q$
IV. $(\, p \rightarrow q) \wedge (p \wedge \neg q)$
V. $p \wedge (q \wedge \neg r\,)$

(Part 1) Which of these is a tautology?

(a) I.
(b) II.
(c) III.
(d) IV.
(e) V.

(Part 2) Which of these is a contradiction (in other words, it isn't satisfiable)?

(a) I.
(b) II.
(c) II.
(d) IV.
(e) V.

4. Using a truth table, show that the following expression is a contradiction:

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

5. Using a truth table, show that the following expression is a contradiction:

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

## Counterexamples

To show that a Boolean expression is a tautology, we must show that it is true for *all* values of its variables.

But to show that an expression is *not* a tautology, it suffices to show *just one* set of values that make the expression false. We call such a set of values a ***counterexample***.

Similarly, we can refute the claim that an expression is a contradiction by showing a single set of values that make the expression true.

Consider the expression:

$$((p \rightarrow r) \rightarrow s) \equiv ((s \rightarrow r) \rightarrow p)$$

We want to show that it is not a tautology. (Another way to think of it is that there are exceptions to it.) To prove that it's not a tautology, we can simply find one of those exceptions.

You can use the Truth Table app to do this yourself.

www.truthtables.org/#/true/((p->r)->s)=((s->r)->p)

There are two counterexamples to the claim that this expression is a tautology. One of them is $p$ = *True*, $r$ = *True* and $s$ = *False*.

## Problems

1. Consider the claim: $\neg(a \vee (b \vee c)) \equiv (\neg a \vee (\neg b \vee \neg c))$

Build a truth table for it. You can do it in the Truth Table app:

www.truthtables.org/#/true/!(a|(b|c))=(!a|(!b|!c))

Which of the following is true:

a) The claim is a tautology.
b) The claim is a contradiction.
c) The claim is neither a contradiction nor a tautology and a proof that it isn't a tautology is the case in which $a$ = *True*, $b$ = *True*, and $c$ = *True*.
d) The claim is neither a contradiction nor a tautology and a proof that it isn't a tautology is the case in which $a$ = *False*, $b$ = *True*, and $c$ = *True*.
e) The claim is neither a contradiction nor a tautology and a proof that it isn't a contradiction is the case in which $a$ = *False*, $b$ = *False*, and $c$ = *True*.

# More Boolean Operators

## How Many Boolean Operators Could There Be?

So far, we've introduced the following binary Boolean operators:

- $\wedge$ (*and*)
- $\vee$ (*or*), by which we mean *inclusive or*
- $\equiv$ (*is equivalent to*), also called *if and only if* or *iff*
- $\rightarrow$ (*implies*)

It's clear why we've picked these. They correspond to natural notions about which we want to reason. But are they the only possible binary Boolean operators?

The answer is no. In fact, we've already mentioned, just in passing, another version of *or*. Are there still more useful operators? The answer is yes.

Let's consider the truth tables that we've used to define our binary operators so far. They all look like this:

| $p$ | $q$ | $p$ operator $q$ |
|---|---|---|
| $P$ | $T$ | $v_1$ |
| $T$ | $F$ | $v_2$ |
| $F$ | $T$ | $v_3$ |
| $F$ | $F$ | $v_4$ |

Each of the values, $v_1$ through $v_4$, is either $T$ or $F$.

Now let's look at a single truth table that shows us the four final columns in the four tables that define our binary operators so far:

| $p$ | $q$ | $p \vee q$ | $p \rightarrow q$ | $p \equiv q$ | $p \wedge q$ |
|---|---|---|---|---|---|
| $T$ | $T$ | $T$ | $T$ | $T$ | $T$ |
| $T$ | $F$ | $T$ | $F$ | $F$ | $F$ |
| $F$ | $T$ | $T$ | $T$ | $F$ | $F$ |
| $F$ | $F$ | $F$ | $T$ | $T$ | $F$ |

If we think of each of those result columns as a list of values, we've got:

- *T, T, T, F,*
- *T, F, T, T,*
- *T, F, F, T,*
- *T, F, F, F*

Are those the only possible lists of four values of *T* and *F*? Clearly not. For example, we could have:

- *T*, *F*, *T*, *F*,
- *F*, *T*, *T*, *T*,
- *F*, *T*, *T*, *F*,
- and others

How many different four-item lists are there? Well, the first item can have one of two values (*T* or *F*). For each of them, there are two values for the second item. So there are four different two-element lists. For each of them, there are two values for the third item, so there are eight different three-element lists. Finally, the fourth item can have one of two values, so there are 16 different four-element lists.

Each of those lists could be a final column in a truth table that defines an operator. So, in principle, there could be 16 different binary Boolean operators. But not all of them are useful enough to bother defining.

### Some Additional Useful Operators

There are, however, three more operators that can be useful. One of them corresponds to an everyday notion. We've mentioned it before: *exclusive or*. The other two aren't going to seem intuitive to you, but they are useful in circuit design (as is *exclusive or*). The three we may care about are:

- *exclusive or* (written, "$p$ XOR $q$" or "$p \oplus q$") is true when exactly one (but not both) of its two operands is true.
- NAND (standing for, *not and*) is equivalent to $\neg(p \wedge q)$. In other words, it is true when $p \wedge q$ is false and thus the *not* of it is true. Alternatively, at least one of $p$ and/or $q$ is false.
- NOR (standing for, *not or*) is equivalent to $\neg(p \vee q)$. In other words, it is true when $p \vee q$ is false and thus the *not* of it is true. Alternatively, neither $p$ nor $q$ is true.

So that we can see the whole picture here's a single truth table that includes all 16 possible binary Boolean operators:

| $p$ | $q$ | $T$ | $p \vee q$ | $q \rightarrow p$ | $p$ | $p \rightarrow q$ | $q$ | $p \equiv q$ | $p \wedge q$ | $p$ NAND $q$ | $p$ XOR $q$ | $\neg q$ | | $\neg p$ | | $p$ NOR $q$ | $F$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $T$ | $T$ | $T$ | $T$ | $T$ | $T$ | $T$ | $T$ | $T$ | $T$ | $F$ | $F$ | $F$ | $F$ | $F$ | $F$ | $F$ | $F$ |
| $T$ | $F$ | $T$ | $T$ | $T$ | $T$ | $F$ | $F$ | $F$ | $F$ | $T$ | $T$ | $T$ | $T$ | $F$ | $F$ | $F$ | $F$ |
| $F$ | $T$ | $T$ | $T$ | $F$ | $F$ | $T$ | $T$ | $F$ | $F$ | $T$ | $T$ | $F$ | $F$ | $T$ | $T$ | $F$ | $F$ |
| $F$ | $F$ | $T$ | $F$ | $T$ | $F$ | $T$ | $F$ | $T$ | $F$ | $T$ | $F$ | $T$ | $F$ | $T$ | $F$ | $T$ | $F$ |

***Nifty Aside***

Boolean logic forms the basis for electronic circuit design. If we let *T* correspond to 1 and *F* correspond to 0, we can build circuits, for example, to perform binary arithmetic. In addition to the operators that we'd already defined, three of our new ones, XOR, NAND, and NOR are particularly useful in building circuits.

## Problems

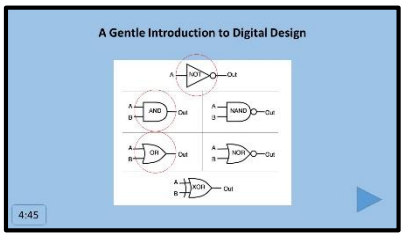
1. For what values of $p$, $q$, and $r$ is the expression, "($p$ XOR $q$) $\wedge$ $r$" true? (Let $v_1$, $v_2$, $v_3$ be the values of $p$, $q$, and $r$, respectively.)

(a) *T*, *F*, *T*
(b) *T*, *F*, *F*
(c) *F*, *T*, *T*
(d) *T*, *T*, *T*
(e) two of these

2. For how many values of $p$, $q$, and $r$ is the expression, "($p$ NAND $q$) NAND $r$" true? (In other words, how many rows of the expression's truth table contain *T* in the last column?)

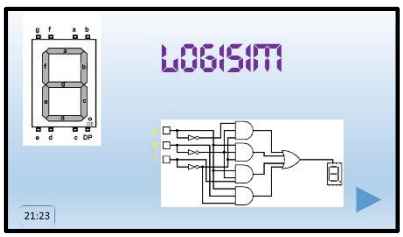
## Boolean Logic in Circuit Design

Boolean logic plays a key role in the design of the modern digital circuits that sit inside all of our phones, pods, tablets, and many other things.




This video introduces, with examples, the key ideas of digital design.

https://www.youtube.com/watch?v=WNXip6gwKh4




This video introduces Logisim (http://www.cburch.com/logisim), a free tool for designing and testing digital circuits.

https://www.youtube.com/watch?v=R7UeuJgo2zE