Chapter 9: Overview of Formal Logic

A Little More Logical | Brendan Shea, Ph.D. (Brendan.Shea@rctc.edu)

In this chapter, we'll be introducing the basic ideas of **formal** logic. The "formal" part here means that we'll be translating arguments in "natural" language (such as English, Spanish, or Chinese) into a "formal" language with specified rules. Formal logic has been crucial to the development of modern computers, among other things (in fact, modern computer **programming languages** are simply formal languages used to communicate with computers).

We'll be talking about three different varieties of formal logic: (1) categorical logic, (2) propositional logic, and (3) predicate logic.

NOTE: The goal here is NOT to master every small detail, but instead to get a sense of the "Big Picture" of how formal languages work. If all of the symbols are overwhelming, I'd encourage you to take a look at the reading in the second half of the chapter, which can help provide some "context" as to why all of this matters!

1 CONTENTS

2 ARISTOTLE'S "CATEGORICAL LOGIC"

In this section, we'll be starting our study of **categorical logic,** which is a type of formal, deductive logic that deals with relationships between categories of things. The study of categorical logic goes all the way back to the Greek philosopher Aristotle (384-322 BCE), and continues all the way up to the current day. While it isn't quite as "powerful" as some of the more modern methods of deductive logic we'll learn later in the class, the rules are much simpler, and it works perfectly well as a tool for analyzing many common types of deductive arguments.

1. What is a **categorical statement**? What does it mean to put categorical statements into **standard form**?
2. What are the component parts of a standard form categorical statement? What are the **quantifier, subject term, copula,** and **predicate term**?
3. What are the four forms of categorical statements? What is the **quality** and **quantity** of each?

4. What does it mean for a term to be **distributed** by a categorical statement? How can you determine whether a particular term has been distributed?

Categorical logic, like other varieties of formal, deductive logic, aims to formulate *rules* for evaluating good (valid) or bad (invalid) arguments. These rules can then be applied in a mechanical fashion, even by people who don't understand what the argument in question is "about," or who disagree on the truth of the premises. In the 20th century, the study of formal logic eventually led to the invention of the computer, which is essentially a machine that does *nothing but* process arguments in formal logic. In order to use formal logic at all, however, we need to begin by constructing a new "language" in which to express our premises and conclusions. This language, unlike ordinary English, shouldn't be open to ambiguity or misinterpretation. With that in mind, we'll now turn to the language of categorical logic.

2.1 CATEGORICAL STATEMENTS: STANDARD FORM

A **categorical statement** (or **categorical proposition**) makes a claim about the relationship between two categories of objects.

1. "Many insects are vegetarians." (The category of insects includes one or more members of the category of vegetarians.)
2. "Reality television shows are not appropriate for children." (No members of the category of reality TV shows are members of the category of TV shows appropriate for children.)
3. "The 2022 Rams are Superbowl winners." (The category of Super-Bowl winning teams includes the category of teams identical to the 2022 Rams.)

Every **standard form categorical statement** has the form <Quantifier><subject term><copula><predicate term>

- The **quantifier** ("All", "No", or "Some") specifies how much of the subject class is being talked about.
- The **subject term** and **predicate term** are plural noun phrases naming the two classes being related.
- The **copula** ("are" or "are not") connects the subject term with the predicate term.

In order to apply the methods of categorical logic that we will be learning in future classes, you will need to express both the premises and conclusion as standard form categorical statements.

2.2 FOUR TYPES OF CATEGORICAL STATEMENTS

Standard form categorical statements come in just FOUR basic types, each of which is abbreviated by a letter:

Statement	Type	Common English Expressions
"All S are P"	Universal Affirmative (A)	"Only Ps are Ss", "Every S is a P", "Any S is a P", "Whatever is S is P", "If something is S, it is P", "Something is a P if it is S"
"No S are P"	Universal Negative (E)	"If S, then not P", "All Ss are non-Ps", "Ss are not Ps", "Nothing S is P"
"Some S are P"	Particular affirmative (I)	"There are Ss that are Ps," "Most Ps are Ss", "A few Ss are Ps", "At least one S is P"
"Some S are not P"	Particular negative (O)	"Not all Ss are Ps," "Many Ss are not Ps," "Ss are not always P", "A few Ss are not P"
"All S are not P"	NOT STANDARD FORM!!!	This is NOT a standard form categorical statement, because it might mean two different things. For example, "all birds are not ravens" means "some birds are not ravens" while "all birds are not mammals" means "no birds are mammals." Because of this

		ambiguity, you CANNOT use statements of this form in categorical logic.

In order to use categorical logic, ALL premises and conclusions must be translated into standard form *A, E, I,* or *O* statements.

2.3 DETERMINING THE FORM: QUANTITY AND QUALITY

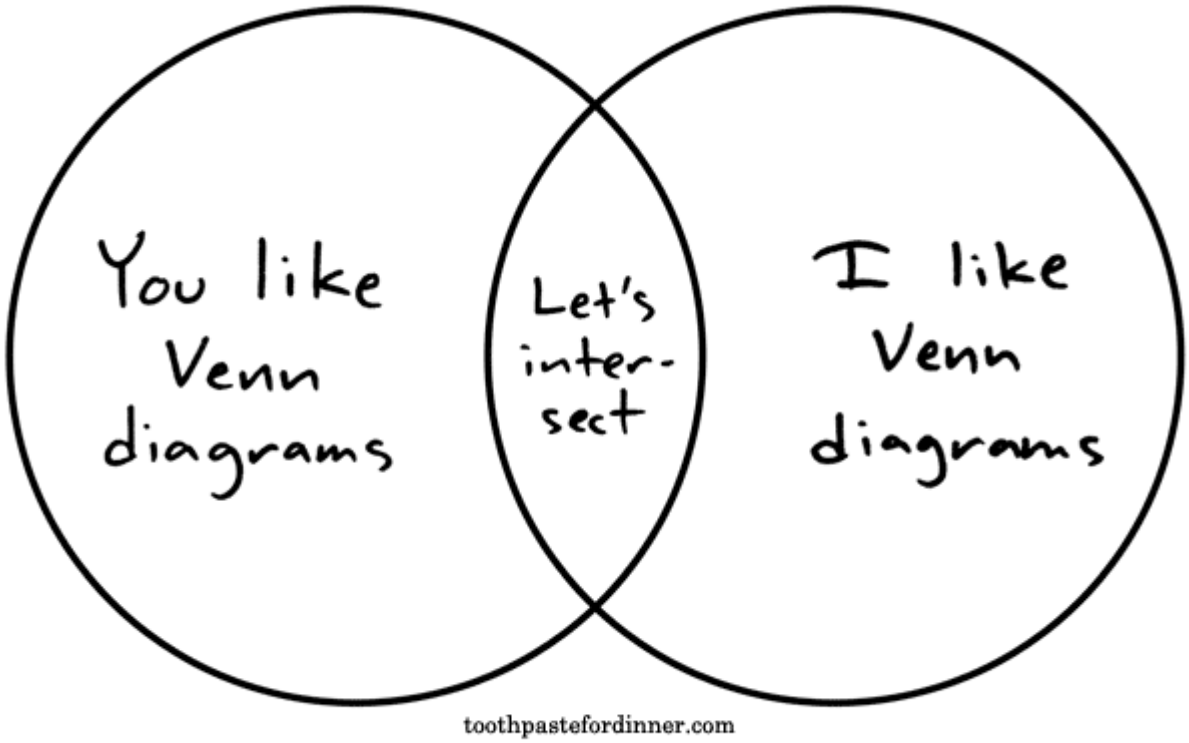


Figure 1 Venn diagrams are often used to represent categorical logic.

The form of a categorical statement is entirely determined by two properties: *quality* and *quantity*. The **quality** of a categorical statement depends on whether it states that something BELONGS to a class, or whether it states that something does NOT BELONG to a class.

- "All S are P" (A) and "Some S are P" (I) have **affirmative** quality.
- "No S are P" (E) and "Some S are not P" (O) have **negative** quality.

The **quantity** of a categorical statement depends on whether it makes a claim about EVERY member of a class, or just SOME (which means "at least one") members of a class.

- "All S are P" (A) and "No S are P" (E) have **universal** quantity.
- "Some S are P" (I) and "Some S are not P" (O) have **particular** quantity.

2.4 WHICH TERMS ARE DISTRIBUTED?

In the categorical statements like "All S are P", "Some S are M", and "No P are M", the **terms** are the individual letters S, P, and M. Logicians say that a term is **distributed** by a statement if and only if that

statement makes a claim about *every* member of the class denoted by the term. What terms are distributed depends on the type of categorical statement:

Statement	Distributes	Example
"All S are P" (A)	S, but not P	"All rubies are gems" makes a claim about all rubies, but not all gems
"No S are P" (E)	Both	"No tires are tulips" makes a claim about every tire AND every tulip.
"Some S are P" (I)	Neither	"Some trees are tall things" doesn't make a claim about every tree, or every tall thing. So, it distributes neither term.
"Some S are not P" (o)	P only	"Some novels are not romances" does NOT make a claim about every novel, but it DOES make a claim about every romance. Why? This statement claims that there is at least one thing in the universe (in this case, at least one book) that is NOT a romance. This tells you something about romance: not *everything* is a romance.

2.5 Summary Table

Statement	Letter name	Quantity	Quality	Terms distributed
All S are P	A	Universal	Affirmative	S
No S are P	E	Universal	Negative	S and P
Some S are P	I	Particular	Affirmative	None
Some S are not P	O	Particular	Negative	P

2.6 Translating English Statements

Many English statements don't have obvious categorical equivalents. Over the years, however, logicians have worked out a number of "tricks" to translating English statements.

Category	Original	Categorical Translation
Active Verbs	Pat ran very quickly.	All persons identical to Pat are persons who ran very quickly.
General	Mammals give live birth.	All mammals are beings that give live birth.
Individuals	Lincoln was a U.S. president.	All persons identical to Lincoln are former U.S. presidents.
Times	It is 12:00 A.M.	All times identical to the present and times identical to 12:00 A.M.
Places	I'll be at the bar soon.	All persons identical to myself are people who will be at the bar soon.
"Always", "Never"	It never rains on the sun.	No places identical to the Sun are places it rains.
"Occasionally"	I eat pizza occasionally.	Some times are times I eat pizza.
Numbers	Two people drank Coke.	Some people are Coke-drinkers. (Note: In categorical logic, we can't say "two." We can only say "at least one").

2.7 Solved Problems

Solved Problem 1: Identify the subject term, predicate term, quantity, and quality of the categorical statement. Then, say which terms are distributed.

Statement	Subject?	Predicate?	Type	Distributes

No Vikings are bears.	Vikings	Bears	Universal Negative (E)	S, P
Some Vikings who sail ships are not bears that eat people.	Vikings who sail ships	Bears that eat people	Particular Negative (O)	P
Some non-bears are non-lions	Non-bears	Non-lions	Particular Affirmative (I)	Neither
All Green Bay Packers are football players.	Green Bay Packers	football players	Universal Affirmative (A)	S

Solved Problem 2: Translate the following into standard form categorical statements.

Statement	Translation
"The 2008 stock market crash was caused by a drop in house prices."	All events identical to the 2008 stock market crash are events caused by drops in house prices.
"Not all changes in stock prices have clear causes."	Some changes in stock prices are not events that have clear causes.
"A few people have made a lot of money by digging up gold in their backyards."	Some people are people who made a lot of money by digging up gold in their backyards.
"Americans don't save enough money."	No Americans are people who save enough money." (Note: Categorical logic has problems with claims like this, since the most plausible interpretation is that we are talking about the *average* American, and not ALL or SOME Americans. These sorts of claims often feature in inductive arguments, which categorical logic is not designed to analyze.)
"If you want to retire at 55, you need to save a lot of money now."	All persons wishing to retire at 55 are persons who should save a lot of money now.
"Only fools would invest all of their money in tulips."	All people who invest all of their money in tulips are fools. (Note: This was really a thing!)
"I own a house, but don't own a car."	All persons identical to me are persons who own a house. No persons identical to me are person who own a car. (Note: In many cases, it is best to use two statements, as we did here.)
"Stock market crashes can happen at any time."	All times are times that stock market crashes can happen.
"Unless you are a genius, you shouldn't try to make money by speculating on foreign currency."	No people who are non-geniuses are people who try to make money… OR All people who try to make money…are geniuses… (Note: In general, "unless" is translated as "if not")

2.8 REVIEW QUESTIONS

Translate the following into standard form categorical statements.

1. Vigorous writing is concise. (W. Strunk)
2. Creativity can solve almost any problem. (G. Lois)
3. Honesty is a good thing, but it is not profitable to its possessor unless it is kept under control. (D. Marquis)
4. College isn't the place to go for ideas. (H. Keller)
5. The time you enjoy wasting is not wasted time. (B. Russell).
6. A wise man will make more opportunities than he finds. (F. Bacon)

3 OVERVIEW OF PROPOSITIONAL LOGIC

In this section, we'll take a look at the basic ideas of propositional logic. **Propositional logic** is a form of deductive logic that studies the relationship between different statements (or "propositions"). Just as categorical logic allows us to prove things about how different *categories* relate to one another, propositional logic will allow us to see how the *truth* or *falsity* of one statement allows to make inferences about the truth or falsity of *another* statement.

1. What is propositional logic? What are the logical operators (or connectives)?
2. What is the difference between a simple statement (or simple proposition) and compound statement (or compound proposition)?
3. What symbols do we use to represent negation, conjunctions, disjunctions, material implication, and material equivalence?
4. What are the antecedent and consequent of a conditional statement? How do these relate to the concepts of sufficient condition and necessary condition?
5. What is the main operator of a compound statement? What does it mean to say that a certain statement is a well-formed formula (WFF)?

Propositional logic, like categorical logic, has a long history going all the way back to Greek philosophers. In particular, the Stoic philosopher **Chrysippus** (279-206 BCE) is often credited with an early formulation. However, serious study of it didn't really start until the 17th-19th centuries, when mathematicians like **Gottfried Leibniz, George Boole,** and **Augustus DeMorgan** began working on formal representations of deductive reasoning. Over the last 150 years or so, logicians expanded the basic ideas of propositional logic to produce **predicate logic** (which combines aspects of both categorical and propositional logic). These ideas provided the basis for the development of computers, and much contemporary work on this sort of logic is done by computer scientists.

3.1 BASIC CONCEPTS IN PROPOSITIONAL LOGIC

Figure 2 Propositional logic was originally developed by "Stoic" philosophers. Stoics advised you to "accept" whichever propositions happened to be true. (From Existential comics).

Propositional logic involves **simple statements** (statements that don't contain any other statements as parts) connected with logical operators, such as AND, OR, NOT, and IF-THEN. A **compound statement** is a statement that *does* contain other statements as parts.

Simple statements are represented by capital letters. "Wilbur is a pig" and "Babe is a pig" are both simple statements. In propositional logic, these statements are represented by upper case letters like W and B.

Compound statements put these letters together. "Both Wilbur and Babe are pigs" is a compound statement (since it contains two simple statements are parts, which it joins with the logical word "and"). You could represent this as "W AND B," where AND here is a logical operator (see below).

The Logical Operators. Propositional logic uses **logical operators** (usually represented by symbols) to connect or modify simple statements: OR, XOR, AND, IF-THEN, and NOT. Most systems don't use XOR; I'm including it here just so you can see the contrast it with OR (and because it is very close to the English word "or")

Operator	Name	Symbols	Truth conditions
P OR Q	Disjunction	$P \vee Q$	True when either P is true, Q is true, or both are true. False only when BOTH P and Q are false. (So, "either Lincoln was a US President or Washington was" is TRUE.)
P XOR Q	Exclusive Disjunction	------	True when either P is true or Q is true. False when both are false or both are true. This corresponds to the English word "or", but is NOT part of most propositional logic systems.
NOT P	Negation	$\sim P$	True whenever P is false. $\sim\sim P$ means the same thing as P. (So, "it is false that the Yankees didn't win" means "The Yankees won.")
IF P THEN Q	Material Implication	$P \supset Q$	False ONLY when the **antecedent** P is true and the **consequent** Q is false, and true in every other case. We say that P is **sufficient** for Q, and that Q is **necessary** for P.
P AND Q	Conjunction	$P \cdot Q$	True when both P is true and Q is true; false in every other case.
P EQUALS Q	Material Equivalence	$P \equiv Q$	True whenever P and Q have the *same* truth value (both are true, or both are false). Importantly, this does NOT mean that P and Q "mean the same thing." It just means they are either both true, or both false.

3.2 Translating Statements into Propositional Logic

In order to use propositional logic to evaluate arguments written in English, we need to first translate them into the appropriate symbols. When doing this, it is important to remember that you're never going to be able to capture *everything* about an English statement in propositional logic. In particular, we can't always capture the **connotation** of sentences. For example, the English statement "I really wanted to eat doughnuts, but I ate salad instead" becomes something like "I wanted to eat doughnuts AND I did not eat doughnuts AND I ate salad." This obviously misses part of the "feel" of the original statement.

Here are some "tips" for translating common expressions:

English Expression	Translation
"It is false that P", "Not P", "It is not the case that P"	~P
"P and Q", "P but Q", "P; however, Q", "Both P and Q", "P; additionally, Q"	P · Q
"P or Q", "P unless Q"	P ∨ Q

"If P then Q", "P only if Q", "Q if P", "Given that P, Q", "P is a sufficient condition for Q", "Q is a necessary condition for P"	P ⊃ Q
"P if and only if Q", "P is equivalent to Q", "P is a necessary and sufficient condition for Q"	P ≡ Q
"Neither P nor Q", "Both P and Q are false"	~P · ~Q or ~(P ∨ Q)

3.3 Main Operators and Well-Formed Formulas

Just as is the case with categorical logic (and just as is the case with ordinary English) statements in propositional logic have their own "grammar" that you'll need to follow. A statement that follows this grammar is called a **well-formed formulas (wff)**. If a statement is NOT a well-formed formula, then it is (quite literally) "meaningless" from the standpoint of propositional logic. The rules for wffs are quite strict, and admit of no exceptions. The reason for this is because we want to avoid any possibility of ambiguity or misinterpretation: we need to make it *perfectly clear* which operators apply to which things.

When translating English statements into propositional logic, you need to remember some simple rules in order to make sure you are producing

1. Negation (tilde) applies only ONE statement. So, ~P is a wff. However, S~P is not. When a tilde is outside a parentheses, as in $\sim(S \cdot P)$, it applies to EVERYTHING inside the parentheses.
2. All of the other operators ($\cdot, \vee, \supset, \equiv$) take exactly two statements (one on each side). So, $S \vee P$ is a wff, but $S \vee$ is not.
3. You need **parentheses** around each pair of statements if you have more than one operator (tildes don't count for this). So $S \vee (P \vee Q)$ is a wff, but $S \vee P \vee Q$ is not. However, $S \vee \sim P$ is fine.

Examples: Well-formed formulas			**Examples: Not well-formed**	
~P	~P ≡ P		P~	P ~≡ P
~~P	P ⊃ (Q · ~R)		~P~	P ⊃ (Q · ~R
P v Q	R ⊃ Q		v(PQ)	RR ⊃ Q
P · ~Q	[R ⊃ (Q ⊃ P)] · Z		P · Q~	[R ⊃ (Q ⊃ P) · Z]
~(P · Q)	(P · Q) v R		~P · Q)	P · Q v R

Where is the main operator? Every compound statement has exactly one **main operator** that determines the truth value of the whole statement. It is the operator whose truth value comes "last" when you are working out the statement's truth value following the "order of operations."

Examples: Finding the Main Operator			
~P	~~P	P · (Q · R)	(P v Q) · (Q ⊃ P)
P ⊃ Q	~P · ~Q	(P ∨ Q) ≡ R	[(P v Q) · (Q ⊃ P)] ⊃ R
~P ⊃ Q	~(P · Q)	~(P ∨ Q) ≡ R	~R ⊃ (P · Q)
~(P ⊃ Q)	~(~P ≡ Q)	~[(P ∨ Q) ≡ R]	~(P ≡ R) ≡ T

3.4 Solved Problems

Problem 1. Suppose that **P = Abraham Lincoln was a U.S. president and W = Abraham Lincoln was a werewolf.** Use this to determine the (1) meaning and (2) truth value of the following compound statements.

Statement	Means	True?

P	Abe Lincoln ("Abe") was a U.S. President	True
W	Abe was a werewolf.	False
P · W	Abe was both president and a werewolf.	False
W~	?	This isn't a WFF
P v W	Abe was either president or a werewolf	True
P ≡ ~W	Abe was president if and only if he wasn't a werewolf	True
P ⊃ W	If Abe was president, he was a werewolf	False
S v ~S	Huh? What does S mean?	True! (This says "Either S is true or it is false." As it turns out, this is always true, regardless of what S means).
W ⊃ P	If Abe was a werewolf, he was president.	True! (This one is weird, but ANY conditional with a false antecedent is true).
W P	?	This isn't a WFF
W · W	Abe was a werewolf and Abe was a werewolf	False. (Another weird one. This actually is a wff; it means the exact same thing as W).
~P · W	Abe wasn't president, but was a werewolf	False.
S · ~S	Here's S again—we don't know what this means.	False (No matter was S means, it can't be true and false at the same time)
~(P · W)	It is false that Abe was BOTH president and a werewolf.	True. (See how parentheses make a difference?)
~(P · W) v P	Either it is false that Abe was BOTH president and a werewolf, or he was president.	True. (Here, we're finally getting to a statement that our brains can't deal with very well. This is where propositional logic starts coming in handy.)
~[~(P · W) v P]	The statement on the line above is false.	False. We know this because the negation in the front (the main operator of this statement) just reverses the truth value of whatever it applies to.
W ≡ ~[~(P · W) v P]	W has the same truth value as the statement on the line above.	True (W is false, and so was the statement on the line above).
~[~P · W v P]	?	Not a WFF

3.5 Review Questions

Translate the following statements into propositional logic. Suppose that S = "Susan will be at the party," P = "Percy will be at the party," and T = "Theo will be at the party." Then, see if you can identify the main operator.

1. It is false that Susan will be at the party.
2. Susan and Percy will both attend the party.
3. Either Susan will be at the party, or Percy will be, or both will be.
4. Susan will be at the party if Percy is.
5. If Susan is not at the party, then Percy won't come either.
6. Neither Susan nor Percy will be at the party.
7. Susan will either be at the party, or she won't be.
8. If Percy and Susan are both at the party, then it is true that either Percy or Susan are at the party.

9. Theo will be at the party if and only either Susan is there or Percy is not.
10. If Theo comes to the party, then Percy will come as well, unless Susan does not come.

4 How Aristotle Created the Computer (by Chris Dixon)[1]

[Brendan's Note: From *The Atlantic.* Provided under Fair Use guidelines for educational use. Please don't distribute outside the class learning management system.]

[Brendan's Note: This article provides a good overview of the ways in which developments in the study of logic—specifically "formal" or "mathematical" logic—have helped shape the modern world.]

The philosophers he influenced set the stage for the technological revolution that remade our world.

The history of computers is often told as a history of objects, from the abacus to the Babbage engine up through the code-breaking machines of World War II. In fact, it is better understood as a history of ideas, mainly ideas that emerged from mathematical logic, an obscure and cult-like discipline that first developed in the 19th century. Mathematical logic was pioneered by philosopher-mathematicians, most notably George Boole and Gottlob Frege, who were themselves inspired by Leibniz's dream of a universal "concept language," and the ancient logical system of Aristotle.

Mathematical logic was initially considered a hopelessly abstract subject with no conceivable applications. As one computer scientist commented: "If, in 1901, a talented and sympathetic outsider had been called upon to survey the sciences and name the branch which would be least fruitful in [the] century ahead, his choice might well have settled upon mathematical logic." And yet, it would provide the foundation for a field that would have more impact on the modern world than any other.

The evolution of computer science from mathematical logic culminated in the 1930s, with two landmark papers: **Claude Shannon**'s "A Symbolic Analysis of Switching and Relay Circuits," and **Alan Turing**'s "On Computable Numbers, With an Application to the *Entscheidungsproblem*." In the history of computer science, Shannon and Turing are towering figures, but the importance of the philosophers and logicians who preceded them is frequently overlooked.

A well-known history of computer science describes Shannon's paper as "possibly the most important, and also the most noted, master's thesis of the century." Shannon wrote it as an electrical engineering student at MIT. His adviser, Vannevar Bush, built a prototype computer known as the Differential Analyzer that could rapidly calculate differential equations. The device was mostly mechanical, with subsystems controlled by electrical relays, which were organized in an ad hoc manner as there was not yet a systematic theory underlying circuit design. Shannon's thesis topic came about when Bush recommended he try to discover such a theory.

Shannon's paper is in many ways a typical electrical-engineering paper, filled with equations and diagrams of electrical circuits. What is unusual is that the primary reference was a 90-year-old work of mathematical philosophy, **George Boole**'s *The Laws of Thought*.

[1] Chris Dixon, "How Aristotle Created the Computer," The Atlantic, March 20, 2017, https://www.theatlantic.com/technology/archive/2017/03/aristotle-computer/518697/.

Today, Boole's name is well known to computer scientists (many programming languages have a basic data type called a Boolean), but in 1938 he was rarely read outside of philosophy departments. Shannon himself encountered Boole's work in an undergraduate philosophy class. "It just happened that no one else was familiar with both fields at the same time," he commented later.

Boole is often described as a mathematician, but he saw himself as a philosopher, following in the footsteps of Aristotle. *The Laws of Thought* begins with a description of his goals, to investigate the fundamental laws of the operation of the human mind:

> *The design of the following treatise is to investigate the fundamental laws of those operations of the mind by which reasoning is performed; to give expression to them in the symbolical language of a Calculus, and upon this foundation to establish the science of Logic ... and, finally, to collect ... some probable intimations concerning the nature and constitution of the human mind.*

He then pays tribute to Aristotle, the inventor of logic, and the primary influence on his own work:

In its ancient and scholastic form, indeed, the subject of Logic stands almost exclusively associated with the great name of Aristotle. As it was presented to ancient Greece in the partly technical, partly metaphysical disquisitions of *The Organon*, such, with scarcely any essential change, it has continued to the present day.

Trying to improve on the logical work of Aristotle was an intellectually daring move. Aristotle's logic, presented in his six-part book *The Organon*, occupied a central place in the scholarly canon for more than 2,000 years. It was widely believed that Aristotle had written almost all there was to say on the topic. The great philosopher Immanuel Kant commented that, since Aristotle, logic had been "unable to take a single step forward, and therefore seems to all appearance to be finished and complete."

[Brendan: In general, it's tough to overstate just how influential Aristotle was. His ideas about physics, astronomy, logic, and many other areas were basically treated as the *only* theories for thousands of years (until people like Copernicus, Darwin, etc. Why do think this was?]

Aristotle's central observation was that arguments were valid or not based on their logical structure, independent of the non-logical words involved. The most famous argument schema he discussed is known as the syllogism:

- All men are mortal.
- Socrates is a man.
- Therefore, Socrates is mortal.

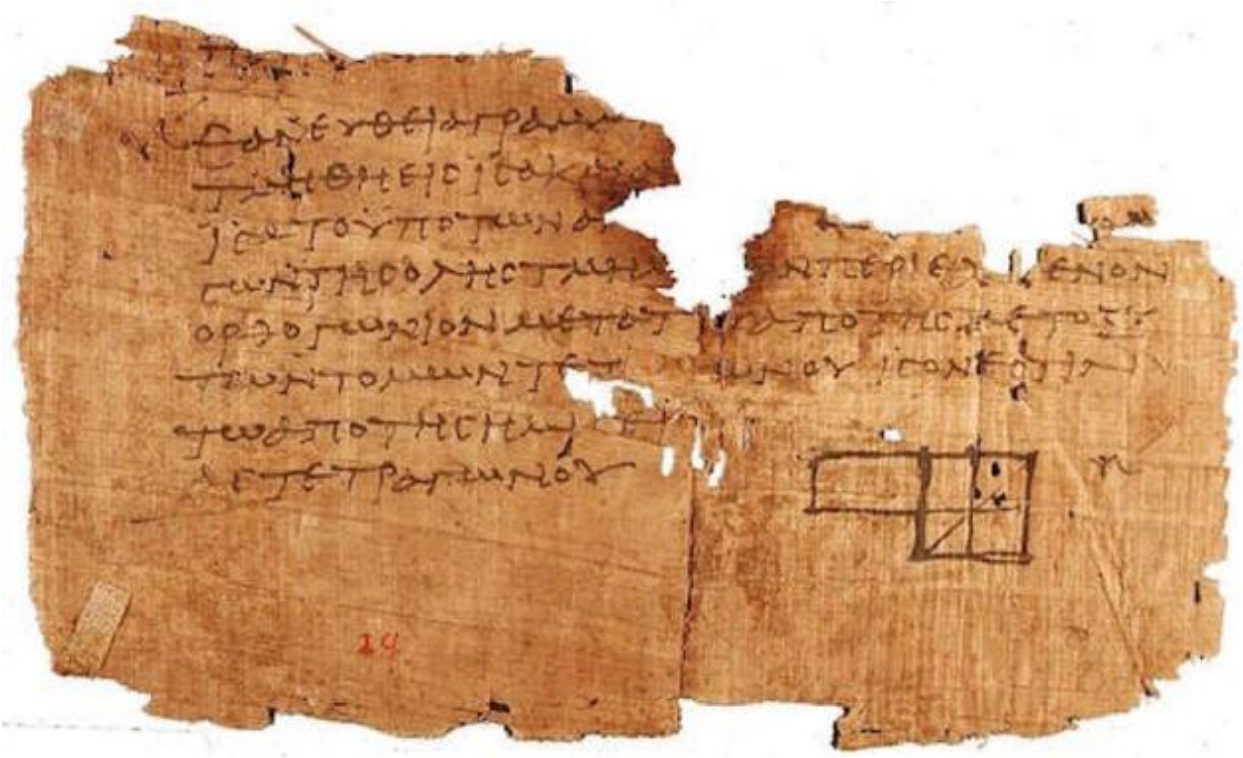
You can replace "Socrates" with any other object, and "mortal" with any other predicate, and the argument remains valid. The validity of the argument is determined solely by the logical structure. The logical words—"all," "is," are," and "therefore"—are doing all the work.

Aristotle also defined a set of basic axioms from which he derived the rest of his logical system:

- An object is what it is (Law of Identity)
- No statement can be both true and false (Law of Non-contradiction)
- Every statement is either true or false (Law of the Excluded Middle)

These axioms weren't meant to describe how people actually think (that would be the realm of psychology), but how an idealized, perfectly rational person ought to think.

Aristotle's axiomatic method influenced an even more famous book, Euclid's *Elements*, which is estimated to be second only to the Bible in the number of editions printed.



A fragment of the *Elements* (Wikimedia Commons)

Although ostensibly about geometry, the *Elements* became a standard textbook for teaching rigorous deductive reasoning. (Abraham Lincoln once said that he learned sound legal argumentation from studying Euclid.) In Euclid's system, geometric ideas were represented as spatial diagrams. Geometry continued to be practiced this way until **René Descartes**, in the 1630s, showed that geometry could instead be represented as formulas. His *Discourse on Method* was the first mathematics text in the West to popularize what is now standard algebraic notation—x, y, z for variables, a, b, c for known quantities, and so on.

Descartes's algebra allowed mathematicians to move beyond spatial intuitions to manipulate symbols using precisely defined formal rules. This shifted the dominant mode of mathematics from diagrams to formulas, leading to, among other things, the development of calculus, invented roughly 30 years after Descartes by, independently, Isaac Newton and Gottfried Leibniz.

[Brendan: You can see here how important "philosophers" have been to the development of math. Descartes brings us modern algebra, Leibniz calculus, etc. It's sometimes tough for us to understand just how revolutionary these ideas were when they were first proposed.]

Boole's goal was to do for Aristotelean logic what Descartes had done for Euclidean geometry: free it from the limits of human intuition by giving it a precise algebraic notation. To give a simple example, when Aristotle wrote:

> *All men are mortal.*

Boole replaced the words "men" and "mortal" with variables, and the logical words "all" and "are" with arithmetical operators:

$$x = x * y$$

Which could be interpreted as "Everything in the set x is also in the set y."

The *Laws of Thought* created a new scholarly field—mathematical logic—which in the following years became one of the most active areas of research for mathematicians and philosophers. Bertrand Russell called the *Laws of Thought* "the work in which pure mathematics was discovered."

Shannon's insight was that Boole's system could be mapped directly onto electrical circuits. At the time, electrical circuits had no systematic theory governing their design. Shannon realized that the right theory would be "exactly analogous to the calculus of propositions used in the symbolic study of logic."

He showed the correspondence between electrical circuits and Boolean operations in a simple chart:

Table I. Analogue Between the Calculus of Propositions and the Symbolic Relay Analysis

Symbol	Interpretation in Relay Circuits	Interpretation in the Calculus of Propositions
X	The circuit X	The proposition X
0	The circuit is closed	The proposition is false
1	The circuit is open	The proposition is true
$X + Y$	The series connection of circuits X and Y	The proposition which is true if either X or Y is true
$X\ Y$	The parallel connection of circuits X and Y	The proposition which is true if both X and Y are true
X'	The circuit which is open when X is closed and closed when X is open	The contradictory of proposition X
=	The circuits open and close simultaneously	Each proposition implies the other

Figure 3 Shannon's mapping from electrical circuits to symbolic logic (University of Virginia)

This correspondence allowed computer scientists to import decades of work in logic and mathematics by Boole and subsequent logicians. In the second half of his paper, Shannon showed how Boolean logic could be used to create a circuit for adding two binary digits.

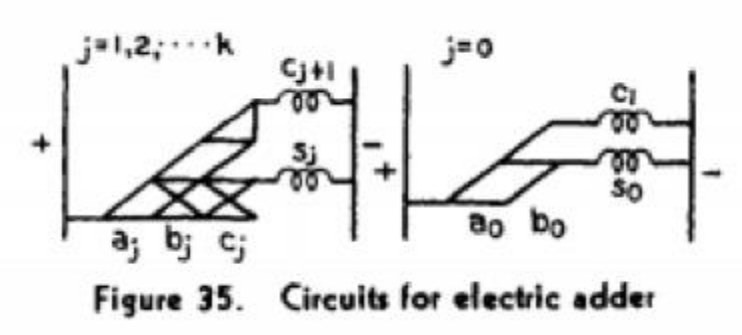


Figure 35. Circuits for electric adder

Figure 4 Shannon's adder circuit (University of Virginia)

By stringing these adder circuits together, arbitrarily complex arithmetical operations could be constructed. These circuits would become the basic building blocks of what are now known as arithmetical logic units, a key component in modern computers.

[Brendan: At the most fundamental level, computer processors are "built" to do logic (and nothing else!). This helps explain why logic is so important in higher-level "programming languages" that humans use to interact with computers.]

Another way to characterize Shannon's achievement is that he was first to distinguish between the *logical* and the *physical* layer of computers. (This distinction has become so fundamental to computer science that it might seem surprising to modern readers how insightful it was at the time—a reminder of the adage that "the philosophy of one century is the common sense of the next.")

Since Shannon's paper, a vast amount of progress has been made on the physical layer of computers, including the invention of the transistor in 1947 by William Shockley and his colleagues at Bell Labs. Transistors are dramatically improved versions of Shannon's electrical relays—the best known way to physically encode Boolean operations. Over the next 70 years, the semiconductor industry packed more and more transistors into smaller spaces. A 2016 iPhone has about 3.3 billion transistors, each one a "relay switch" like those pictured in Shannon's diagrams.

While Shannon showed how to map logic onto the physical world, Turing showed how to design computers in the language of mathematical logic. When Turing wrote his paper, in 1936, he was trying to solve "the decision problem," first identified by the mathematician David Hilbert, who asked whether there was an algorithm that could determine whether an arbitrary mathematical statement is true or false. In contrast to Shannon's paper, Turing's paper is highly technical. Its primary historical significance lies not in its answer to the decision problem, but in the template for computer design it provided along the way.

Turing was working in a tradition stretching back to Gottfried Leibniz, the philosophical giant who developed calculus independently of Newton. Among Leibniz's many contributions to modern thought, one of the most intriguing was the idea of a new language he called the "universal characteristic" that, he imagined, could represent all possible mathematical and scientific knowledge. Inspired in part by the 13th-century religious philosopher Ramon Llull, Leibniz postulated that the language would be ideographic like Egyptian hieroglyphics, except characters would correspond to "atomic" concepts of math and science. He argued this language would give humankind an "instrument" that could enhance human reason "to a far greater extent than optical instruments" like the microscope and telescope.

He also imagined a machine that could process the language, which he called the **calculus ratiocinator.**

If controversies were to arise, there would be no more need of disputation between two philosophers than between two accountants. For it would suffice to take their pencils in their hands, and say to each other: *Calculemus*—Let us calculate.

[Brendan: What do you think of Leibniz's proposed machine? Will we ever reach this point?]

Leibniz didn't get the opportunity to develop his universal language or the corresponding machine (although he did invent a relatively simple calculating machine, the stepped reckoner). The first credible attempt to realize Leibniz's dream came in 1879, when the German philosopher Gottlob Frege published his landmark logic treatise *Begriffsschrift.* Inspired by Boole's attempt to improve Aristotle's logic, Frege developed a much more advanced logical system. The logic taught in philosophy and computer-science classes today—first-order or predicate logic—is only a slight modification of Frege's system.

Frege is generally considered one of the most important philosophers of the 19th century. Among other things, he is credited with catalyzing what noted philosopher Richard Rorty called the "**linguistic turn"** in philosophy. As Enlightenment philosophy was obsessed with questions of knowledge, philosophy after Frege became obsessed with questions of language. His disciples included two of the most important philosophers of the 20th century—Bertrand Russell and Ludwig Wittgenstein.

The major innovation of Frege's logic is that it much more accurately represented the logical structure of ordinary language. Among other things, Frege was the first to use quantifiers ("for every,"

"there exists") and to separate objects from predicates. He was also the first to develop what today are fundamental concepts in computer science like recursive functions and variables with scope and binding.

Frege's formal language—what he called his "concept-script"—is made up of meaningless symbols that are manipulated by well-defined rules. The language is only given meaning by an interpretation, which is specified separately (this distinction would later come to be called **syntax** versus **semantics)**. This turned logic into what the eminent computer scientists Allan Newell and Herbert Simon called "the symbol game," "played with meaningless tokens according to certain purely syntactic rules."

All meaning had been purged. One had a mechanical system about which various things could be proved. Thus progress was first made by walking away from all that seemed relevant to meaning and human symbols.

As Bertrand Russell famously quipped: "Mathematics may be defined as the subject in which we never know what we are talking about, nor whether what we are saying is true."

An unexpected consequence of Frege's work was the discovery of weaknesses in the foundations of mathematics. For example, Euclid's *Elements*—considered the gold standard of logical rigor for thousands of years—turned out to be full of logical mistakes. Because Euclid used ordinary words like "line" and "point," he—and centuries of readers—deceived themselves into making assumptions about sentences that contained those words. To give one relatively simple example, in ordinary usage, the word "line" implies that if you are given three distinct points on a line, one point must be between the other two. But when you define "line" using formal logic, it turns out "between-ness" also needs to be defined—something Euclid overlooked. Formal logic makes gaps like this easy to spot.

[Brendan: Why do you think the ability of Frege's new "logic" to accurately represent language was so influential? It changed the way people did philosophy and mathematics and (eventually, led to computers!).]

This realization created a crisis in the foundation of mathematics. If the *Elements*—the bible of mathematics—contained logical mistakes, what other fields of mathematics did too? What about sciences like physics that were built on top of mathematics?

The good news is that the same logical methods used to uncover these errors could also be used to correct them. Mathematicians started rebuilding the foundations of mathematics from the bottom up. In 1889, Giuseppe Peano developed axioms for arithmetic, and in 1899, **David Hilbert** did the same for geometry. Hilbert also outlined a program to formalize the remainder of mathematics, with specific requirements that any such attempt should satisfy, including:

- *Completeness:* There should be a proof that all true mathematical statements can be proved in the formal system.
- *Decidability:* There should be an algorithm for deciding the truth or falsity of any mathematical statement. (This is the "*Entscheidungsproblem*" or "decision problem" referenced in Turing's paper.)

Rebuilding mathematics in a way that satisfied these requirements became known as Hilbert's program. Up through the 1930s, this was the focus of a core group of logicians including Hilbert, Russell, Kurt Gödel, John Von Neumann, Alonzo Church, and, of course, Alan Turing.

Hilbert's program proceeded on at least two fronts. On the first front, logicians created logical systems that tried to prove Hilbert's requirements either satisfiable or not.

On the second front, mathematicians used logical concepts to rebuild classical mathematics. For example, Peano's system for arithmetic starts with a simple function called the successor function which increases any number by one. He uses the successor function to recursively define addition, uses addition to recursively define multiplication, and so on, until all the operations of number theory are defined. He then uses those definitions, along with formal logic, to prove theorems about arithmetic.

The historian Thomas Kuhn once observed that "in science, novelty emerges only with difficulty." Logic in the era of Hilbert's program was a tumultuous process of creation and destruction. One logician would build up an elaborate system and another would tear it down.

[Brendan: I realize this is a little technical. In your own words, how would you describe "Hilbert's program"?]

The favored tool of destruction was the construction of self-referential, paradoxical statements that showed the axioms from which they were derived to be inconsistent. A simple form of this **"liar's paradox"** is the sentence:

> *This sentence is false.*

If it is true then it is false, and if it is false then it is true, leading to an endless loop of self-contradiction.

Russell made the first notable use of the liar's paradox in mathematical logic. He showed that Frege's system allowed self-contradicting sets to be derived:

> *Let R be the set of all sets that are not members of themselves. If R is not a member of itself, then its definition dictates that it must contain itself, and if it contains itself, then it contradicts its own definition as the set of all sets that are not members of themselves.*

This became known as **Russell's paradox** and was seen as a serious flaw in Frege's achievement. (Frege himself was shocked by this discovery. He replied to Russell: "Your discovery of the contradiction caused me the greatest surprise and, I would almost say, consternation, since it has shaken the basis on which I intended to build my arithmetic.")

[Brendan: Russell let Frege know about this *just after* Frege completed his book (which had taken him years) and published it! Frege was pretty gracious about this, all things considered…]

Russell and his colleague Alfred North Whitehead put forth the most ambitious attempt to complete Hilbert's program with the *Principia Mathematica,* published in three volumes between 1910 and 1913. The *Principia's* method was so detailed that it took over 300 pages to get to the proof that 1+1=2.

Russell and Whitehead tried to resolve Frege's paradox by introducing what they called **type theory.** The idea was to partition formal languages into multiple levels or types. Each level could make reference to levels below, but not to their own or higher levels. This resolved self-referential paradoxes by, in effect, banning self-reference. (This solution was not popular with logicians, but it did influence computer science—most modern computer languages have features inspired by type theory.)

Self-referential paradoxes ultimately showed that Hilbert's program could never be successful. The first blow came in 1931, when Gödel published his now famous incompleteness theorem, which proved that any consistent logical system powerful enough to encompass arithmetic must also contain statements that are true but cannot be proven to be true. **(Gödel's incompleteness theorem** is one of the few logical results that has been broadly popularized, thanks to books like *Gödel, Escher, Bach* and *The Emperor's New Mind).*

[Brendan: Gödel is something like the "Einstein" of modern logic (and, in fact, Einstein and Gödel both moved to US and became friends later in life). His incompleteness theorem shows that there are TRUE statements in mathematics which we can never PROVE to be true. People have spent a lot of time debating over what this "means." There have been some pretty wild claims (it shows that human souls exist! That God is real! That there is no truth!).]

[Brendan: Gödel, like Einstein, fled Nazi persecution. One of his best friends—the logician and philosopher Moritz Schlick—was killed by a Nazi sympathizer radical in the years leading up to war. After this, Gödel became very worried about being poisoned, and would only eat food his wife prepared. Gödel died when his wife became ill, and he refused to eat. Why do you think Gödel's "intellect" couldn't defend him from these beliefs?]

The final blow came when Turing and Alonzo Church independently proved that no algorithm could exist that determined whether an arbitrary mathematical statement was true or false. (Church did this by inventing an entirely different system called the lambda calculus, which would later inspire computer languages like Lisp.) The answer to the decision problem was negative.

Turing's key insight came in the first section of his famous 1936 paper, "On Computable Numbers, With an Application to the *Entscheidungsproblem*." In order to rigorously formulate the **decision problem** (the "*Entscheidungsproblem*"), Turing first created a mathematical model of what it means to be a computer (today, machines that fit this model are known as "**Universal Turing Machines**"). As the logician Martin Davis describes it:

> *Turing knew that an* ***algorithm*** *is typically specified by a list of rules that a person can follow in a precise mechanical manner, like a recipe in a cookbook. He was able to show that such a person could be limited to a few extremely simple basic actions without changing the final outcome of the computation.*
>
> *Then, by proving that no machine performing only those basic actions could determine whether or not a given proposed conclusion follows from given premises using Frege's rules, he was able to conclude that no algorithm for the Entscheidungsproblem exists.*
>
> *As a byproduct, he found a mathematical model of an all-purpose computing machine.*

Next, Turing showed how a program could be stored inside a computer alongside the data upon which it operates. In today's vocabulary, we'd say that he invented the "**stored-program" architecture** that underlies most modern computers:

> *Before Turing, the general supposition was that in dealing with such machines the three categories—machine, program, and data—were entirely separate entities. The machine was a physical object; today we would call it hardware. The program was the plan for doing a computation, perhaps embodied in punched cards or connections of cables in a plugboard. Finally, the data was the numerical input. Turing's universal machine showed that the distinctness of these three categories is an illusion.*

This was the first rigorous demonstration that any computing logic that could be encoded in hardware could also be encoded in software. The architecture Turing described was later dubbed the "**Von Neumann architecture**"—but modern historians generally agree it came from Turing, as, apparently, did Von Neumann himself.

[Brendan: So, Turing does two things. He describes the "logic" behind modern computers (that store programs in memory and "load" them), and then gives a version of Godel's incompleteness theorem, but for programs. Turing's version says that there is no way of constructing a program that can give you the answer to the question "Will this other program ever halt? Or will it run forever?]

[Brendan: On a more personal note, Turing was persecuted for being homosexual, and committed suicide. What else might he have accomplished, had he lived?]

Although, on a technical level, Hilbert's program was a failure, the efforts along the way demonstrated that large swaths of mathematics could be constructed from logic. And after Shannon and Turing's insights—showing the connections between electronics, logic and computing—it was now possible to export this new conceptual machinery over to computer design.

During World War II, this theoretical work was put into practice, when government labs conscripted a number of elite logicians. Von Neumann joined the atomic bomb project at Los Alamos, where he worked on computer design to support physics research. In 1945, he wrote the specification of the EDVAC—the first stored-program, logic-based computer—which is generally considered the definitive source guide for modern computer design.

Turing joined a secret unit at Bletchley Park, northwest of London, where he helped design computers that were instrumental in breaking German codes. His most enduring contribution to practical computer design was his specification of the ACE, or Automatic Computing Engine.

As the first computers to be based on Boolean logic and stored-program architectures, the ACE and the EDVAC were similar in many ways. But they also had interesting differences, some of which foreshadowed modern debates in computer design. Von Neumann's favored designs were similar to modern CISC ("complex") processors, baking rich functionality into hardware. Turing's design was more like modern RISC ("reduced") processors, minimizing hardware complexity and pushing more work to software.

Von Neumann thought computer programming would be a tedious, clerical job. Turing, by contrast, said computer programming "should be very fascinating. There need be no real danger of it ever becoming a drudge, for any processes that are quite mechanical may be turned over to the machine itself."

Since the 1940s, computer programming has become significantly more sophisticated. One thing that hasn't changed is that it still primarily consists of programmers specifying rules for computers to follow. In philosophical terms, we'd say that computer programming has followed in the tradition of **deductive logic,** the branch of logic discussed above, which deals with the manipulation of symbols according to formal rules.

In the past decade or so, programming has started to change with the growing popularity of machine learning, which involves creating frameworks for machines to learn via statistical inference. This has brought programming closer to the other main branch of logic, **inductive logic,** which deals with inferring rules from specific instances.

[Brendan: While this article doesn't discuss inductive logic, Rudolph Carnap (another friend of Godel's) ended up laying the groundwork for modern inductive logic. Like Godel, he built on the work of Boole, Frege, Russell, and others.]

Today's most promising machine learning techniques use neural networks, which were first invented in 1940s by Warren McCulloch and Walter Pitts, whose idea was to develop a calculus for neurons that could, like Boolean logic, be used to construct computer circuits. Neural networks remained esoteric until decades later when they were combined with statistical techniques, which allowed them to improve as they were fed more data. Recently, as computers have become increasingly adept at handling large data sets, these techniques have produced remarkable results. Programming in the future will likely mean exposing neural networks to the world and letting them learn.

This would be a fitting second act to the story of computers. Logic began as a way to understand the laws of thought. It then helped create machines that could reason according to the rules of deductive logic. Today,

deductive and inductive logic are being combined to create machines that both reason and learn. **What began, in Boole's words, with an investigation "concerning the nature and constitution of the human mind," could result in the creation of new minds—artificial minds—that might someday match or even exceed our own.**

[Brendan: What do you think—will logic (and computers) ever allow us to create "artificial minds" that match or exceed human minds?]