Lecture 1: Foundations of Logic

What, Why, and How

The overall goal of CprE 310 is to understand how to use mathematical principles in order to solve problems in computer science and engineering.

It is not a stretch to claim that all innovation in mechanical engineering can be distilled down to a (relatively concise) set of laws in *physics*, just as how all innovation in material science can be boiled down to a set of tenets in *chemistry*. Analogously, the behavior of all computing systems can be explained by a few principles in *discrete mathematics*. This will be a somewhat different brand of mathematics from what is typically taught in traditional math courses, but just as important.

A large portion of the course will be devoted to the study of mathematical *proofs*. Proofs can be used to certify that a certain piece of software (or hardware) will **always** work as expected. Such certificates are essential for modern computing systems: with (literally) millions of building blocks in every computer system/computer network/software program, it is vital to ensure that every little block does its job exactly as it should. Proofs enable us to **unambiguously** reason about such systems, detect and correct bugs, and help build faster and better systems.

What is a "proof"? In essence, a "proof" is a method of establishing truth. Now, defining the meaning of 'truth' can be a bit fuzzy, and therefore, the precise meaning of a "proof" differs depending on who you talk to. We will follow the mathematical notion of "proof":

> *A proof of a proposition is a chain of logical deductions leading to the proposition from a set of axioms.*

Quite a mouthful! If we want to give a proof of some proposition, we first need to precisely understand what a "proposition" means, what "axioms" are, and also understand the rules of the game (what types of logical deductions are permitted and what aren't). Today and the next lecture, we will focus on the first bit – propositions.

A bit of backstory: Logical thinking has a rich tradition in *philosophy*, dating back to the Ancient Greeks. Modern logic began to emerge in the 16th century with the work of Descartes and Leibnitz, but became concrete only about 150 years ago when the British mathematicians Boole and De Morgan realized that one can reason about logical statements via *mathematical operations*. This is one of those subtle but remarkable ideas that don't sound like much, but in hindsight, truly changed the course of history. In particular, Lady Ada Lovelace (a student of De Morgan) used these ideas to develop what is now considered the first algorithm that can be executed by a machine, or in other words, the first *computer program*.

Elements of logic

The rest of this lecture will introduce some vocabulary in basic logic which we will use for the remainder of the course.

Propositions

A *proposition* is a declarative sentence (or statement) that is either true or false, but not both.

For instance, the following are propositions:

> Ames is a city located in the state of Iowa.
>
> 2 + 2 = 4.
>
> 2 + 5 < 6.

The first two statements are true, while the last statement is false. However, all three are legit propositions.

The *truth value* of a proposition is denoted as "T" or "1" if a proposition is true, and "F" or "0" if it is false. For example, the truth values of the above 3 propositions are T, T, and F respectively (or if you like: 1, 1, and 0 respectively.)

Not all statements are propositions. For example:

1. What is your name?
2. Be quiet!
3. It will rain tomorrow.
4. $x + 1 = 6$.
5. Booyah.

The sentence (1) is a question (not a statement), and (2) is an imperative (not a declarative). The next two are indeed declarative statements and are a bit subtle. In the case of (3), the reality is that its truth or falsehood depends on chance (and not determined a priori). In the case of (4), its truth or falsehood depends on what x is. (This kind of statement is called a *predicate*; more on this later in the course.) (5) isn't a sentence ^−^.

So, depending on whether they are true or not, propositions can be assigned a truth value – just as how variables in a Java program can be assigned values. Therefore, it is instructive to think of statements as *propositional variables*. For example, the variable *p* can be used to represent the proposition:

> *p*: It is winter season right now in Ames.

and the *value* of *p* will be equal to F. Similarly, if we denote *q* as:

> *q*: The 2020 Tokyo Olympics are yet to happen.

then the value of *q* is T.

Unfortunately, it is not always obvious whether a proposition is true or false. For example, the proposition:

> Every map can be colored with four colors so that adjacent regions on the map have different colors.

did not have a conclusive answer one way or the other for several centuries. It was finally proved to be true in 1976 by Appel and Haken, and is now called the "Four Color Theorem".

Logical connectives

Let us get into the habit of thinking of propositions as mathematical variables. The next step is to define a few basic mathematical *operations* that can manipulate their values, combine different propositions, and so on. We call them *connectives*.

Let *p* and *q* be propositional variables. We begin with the three most fundamental operations:

- *Negation*: Written as $\neg p$, representing "NOT *p*".
- *Conjunction*: Written as $p \wedge q$, representing "*p* AND *q*"
- *Disjunction*: Written as $p \vee q$, representing "*p* OR *q*, OR both."

You may have seen these already in a circuits class; each of these can be realized using a physical device called a "gate".

Let us start with negation, which is the easiest. The operation $\neg p$ simply flips the truth value of *p*. If *p* is true (i.e., its value is T), then its negation is false and vice versa. For example, if *p* represents the proposition "2 + 2 = 5", then the value of *p* is F; therefore, $\neg p$ has a value of T.

Conjunction is next. Given two propositional variables *p* and *q*, the value of $p \wedge q$ is T if both *p* **and** q have a value of T; in all other cases, the value of $p \wedge q$ is F. For example, if *p* represents "John likes apples", *q* represents "John likes oranges", and *r* represents "John likes apples and oranges", then clearly $r = p \wedge q$, since *r* can be true only if both *p* and *q* are also true.

Disjunction is slightly different. Given two propositional variables *p* and *q*, the value of $p \vee q$ is T if **at least one** of *p* and *q* have a value of T; if both *p* and *q* are F, then the value of $p \wedge q$ is F. For example, if *p* represents "John has completed ComS 228", *q* represents "John has enrolled in ComS 228", and *r* represents "John has the necessary pre-reqs for CprE 310", then clearly $r = p \vee q$,

There is a succinct way to characterize the effect of logical connectives: it is called a *truth* table. Basically, a truth table enumerates all possible combinations of value assignments to the propositional variables, and calculates the output of the connectives. Here is the truth table for the above 3 operations; here, *r* represents $\neg p$, *u* represents $p \wedge q$, and *v* represents $p \vee q$.

p	q	r	u	v
F	F	T	F	F
F	T	T	F	T
T	F	F	F	T
T	T	F	T	T

The use of connectives lets us start with a few base propositions, and build complicated statements in a myriad of ways using these propositions as building blocks. We call them "compound propositions". For example, expressions like $p \vee (q \wedge r)$ or $\neg(\neg p \wedge \vee q)$ are compound propositions.

One note about using connectives: be mindful of the *order* in which you evaluate them. For example, $p \vee q \wedge r$ is ambiguous, and may give rise to *completely different* values depending on which connective is evaluated first. It is best to demarcate your preference using parentheses: $(p \vee q) \wedge r$ indicates that the disjunction operation between *p* and *q* is evaluated first, and the result of that is conjuncted with *r*. The preferred order is parenthesis -> negation -> conjunction/disjunction.

English to symbolic logic (and vice versa)

Let's now apply some of the principles of logic that we have just learned to represent ordinary sentences in English.

Suppose we want to write down a logical expression for

It is sunny but not hot.

The first thing to do is identify the propositions in this sentence and model them as propositional variables. One choice (not unique!) is as follows:

p: It is sunny.

q: It is hot.

Therefore, the original sentence can be represented as $p \wedge (\neg q)$.

A sentence like "It is neither sunny nor hot" can be written as $(\neg p) \wedge (\neg q)$. In the opposite direction: if we are given the compound proposition $p \wedge q$, then we can interpret its meaning to represent "It is both sunny and hot".

Let us end with a more interesting example. Suppose we have two propositions:

p: I will go home via bus.

q: I will go home via car.

How should we represent:

r: I will go home either via bus or via car.

At first glance, this looks like $p \vee q$, but there is an implicit "not both" in the sentence! Natural spoken language has several such (unspoken) ambiguities, which is not a problem in daily conversation, but can have serious implications if, for instance, you are trying to formulate ideas precisely.

A compound statement like this is called an *exclusive-OR*, and can be written as:

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

or more concisely:

$p \oplus q$

(Easy homework exercise: can you write down the truth table for $\oplus$?)