What does Discrete mean?

The standard definition is “separate and distinct”. The opposite would be “continuous”. Discrete things can often be characterized by integers, whereas continuous things generally require the real numbers.

What is Discrete Mathematics?

There are numerous branches of mathematics. In general, you want to use the type that fits your task. If you are modeling a cannonball’s flight, that might be calculus. If you are modeling vision, that might be linear algebra.

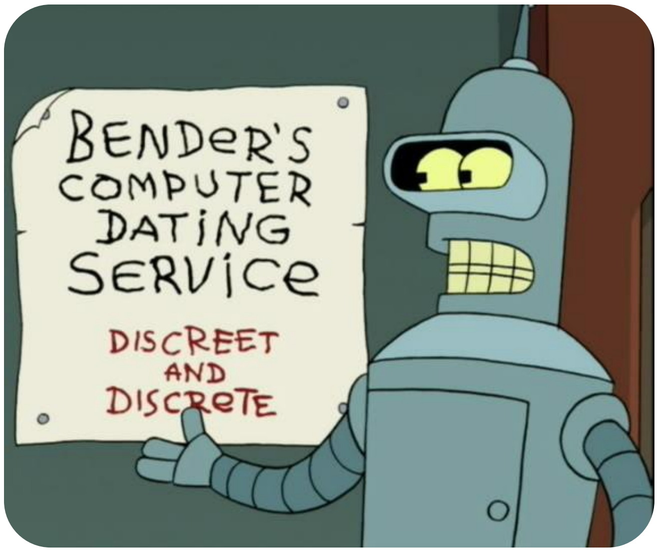


Figure 1: Bender helping us remember Discreet != Discrete

What mathematics should a CS/CE know and use? Well, much of what we do involves discrete numbers. In fact as computers slowly take over the world, things that were formally continuous are now discrete.

Records → **CDs/MP3s**; film → **digital photos**; VHS → **DVDs**.

Computation and a discrete worldview go hand-in-hand. Computer data is discrete (all stored as bits no matter what the data is). Time on a computer occurs in discrete steps (clock ticks), etc. **Because we work almost solely with discrete values, it makes since that we’d need to know discrete mathematics.**

Discrete mathematics is actually a collection of a *large* number of different types of mathematics all used when working with discrete data. Some things we are going to cover in this class include:

- **Logic**[1] (propositional logic, predicate logic, quantified formulae, logical deductions)
  - Architecture (logic gates) ← **My area!**
  - Software engineering (specification and verification)
  - Programming languages (semantics, logic programming)
  - Databases (relational algebra and SQL)
  - Artificial intelligence (automatic theorem proving)
  - Algorithms (complexity and expressiveness)
  - Theory of computation (general notions of computability)
- **Proofs** (including the analysis of algorithms)
  - Software engineering (verification of correctness)
  - Algorithm analysis (showing a task will complete within some time bound (“on time”))
  - Parallel systems (proving a protocol will function correctly in all cases)
- **Asymptotic notation** (“Big Oh” and its friends)
  - Algorithm design and choice (allows us to reasonable compare algorithms rather than implementations)
- **Counting and discrete probability**
  - Architecture (how caches behave, how branch predictors behave) ←**Still my area!**
  - Modeling failure in software and hardware

[1] List from: http://www.cs.rice.edu/~vardi/comp409/, a 400-level class on logic in CS!

So I'll use this material a lot in future classes (or the real world)?

Some of it you will use a lot. I use logic, counting & discrete probability as well as asymptotic notation on a regular basis no matter what classes I teach (daily to weekly). I use proof techniques only very rarely. And honestly, I only use a small fraction of each of these topics. I'd say I use only about 25% of this class material more than once a year. But each person will use different parts depending on what they do.

But more so, discrete math gives us the needed language to discuss and solve problems. Let's consider two examples[2]:

Six Degrees of Kevin Bacon

This is a parlor game wherein movie buffs challenge each other to find the shortest path between an arbitrary actor and prolific Hollywood character actor Kevin Bacon.

Getting to Kevin Bacon's star on the Walk of Fame

Another thing you might want to do is get to Kevin Bacon's star on the Walk of Fame.

Figure 2: Map of the Walk of Fame in Hollywood

To people without some discrete mathematics background, the only two things these two problems would seem to have in common is, well, Kevin Bacon. But a solid CS person would also note that these are both graph theory problems. When solving the Six Degrees of Kevin Bacon or having Google Maps get you to Kevin Bacon's star, the problem is generally described as a graph and the goal is to find the shortest (weighted) path between two vertices in that graph.

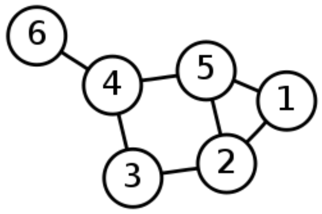


Figure 3: A graph (from pdx.edu)

What's really cool is that we can use the same algorithms to solve either of these two problems! And that's what this class should bring you. A worldview that lets you quickly see ways to address and solve problems you'll encounter as a CS or CE student.

OK, now that's we've got an idea what the class is about, let's address some administrative issues.

[2] Text for Six Degrees of Kevin Bacon comes from the Wikipedia article of the same name.

Structure of the Class

Website: http://www.eecs.umich.edu/courses/eecs203; (includes class schedule)

Piazza: http://piazza.com/umich/summer2016/eecs203

Instructor: Mark Brehob.

- PhD from MSU (go Green!)
- Mainly focused on computer architecture and embedded systems
- Lecturer (full-time teacher) and chief program advisor for computer engineering.
- Office hours: M 10:00-12, Tu 1-2:30, W 1-2:30. 4632 Beyster

GSIs:

- Emily Graetz
    - Friday discussions
    - Office hours:
        - Sunday 2-5 in UGLi Basement (middle tables)
        - Tuesday 4-6, Wednesday 4-6, Friday 2-5 in 1637 Beyster (learning center).
- Jasmine Powell
    - Thursday discussion
    - Office hours:
        - Monday 5-7 in East Hall B723
        - Thursday 3-5 in 1637 Beyster (learning center).

Grades:

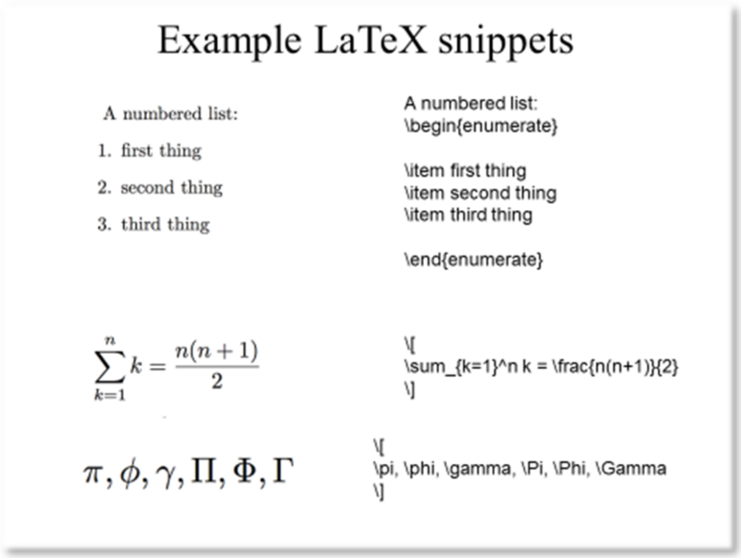
- 10% individual homework (7 assignments, drop 1)
    - HW1 posted, due Thursday at 3pm!
- 10% group homework (6 assignments, drop 1, groups of 1 to 3, hard problems, must use LaTeX!)
    - Groups can change each assignment if you choose. Only list people that help!
- 16% quizes (5 quizzes, drop 1, start of lecture, no notes, 15-25 minutes)
- 29% midterm (May 31st, class time, one page of notes)
- 35% final (June 24th at 4pm, two pages of notes)

Cooperation:

- You are welcome to study together, discussing ideas, etc.
    - But individual homework assignments are to be done individually!
- For group assignments you should only be working with your group.

\end{administrivia};

\begin{discrete math}



Example LaTeX snippets

A numbered list:

1. first thing
2. second thing
3. third thing

```
A numbered list:
\begin{enumerate}

\item first thing
\item second thing
\item third thing

\end{enumerate}
```

$$\sum_{k=1}^{n} k = \frac{n(n+1)}{2}$$

```
\[
\sum_{k=1}^n k = \frac{n(n+1)}{2}
\]
```

$$\pi, \phi, \gamma, \Pi, \Phi, \Gamma$$

```
\[
\pi, \phi, \gamma, \Pi, \Phi, \Gamma
\]
```

Figure 4: Latex is really handy for doing math.

Propositions (1.1)

We spend a lot time proving things in this course. What is a proof?

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

An axiom is a proposition we assume to be true.

Propositional logic

What is a proposition?

A proposition is a declarative statement that is either true or false.

Statement	**True?**	**Proposition?**
Two non-parallel lines in the plane have exactly one point in common.		
Ann Arbor is the capital of Michigan.		
1+1+1=3		
Go blue!		
$x + 5 \neq 10$		
This statement is false.		
There is life on Mars		

Things to watch for:

- There is a difference between not having enough information (such as the x+5 case) and not knowing the answer (such as the Mars case). It just has to have a truth value to be a proposition, you don't need to know the truth value.
- Paradoxes are cool (and useful) but don't have a truth value so aren't propositions.

Logical operators

Say we live in the rather black and white world where we are dealing with propositions. So if ***S*** is "Mark is going to the Store" and ***C*** is "Mark likes Computer games" then we'll assume that each phrase is either true or false (as opposed only sort of liking computer games). We can then use connectives to combine the variables.

Mark is going to the store AND Mark likes computer games.

The above statement is only true if both phrases are true. Let that sentence be ***X***. We can now draw the "truth table" for ***X*** (we'll use the other tables in a minute). When is X true?

S	C	X
F	F	
F	T	
T	F	
T	T	

AND

S	C	
F	F	
F	T	
T	F	
T	T	

____

S	C	
F	F	
F	T	
T	F	
T	T	

____

C	
F	
T	

____

Representation of logical operators.

Using AND, OR, NOT and XOR gets old. So symbols have been used to represent these notions for quite a while. We'll hit three different representations today:

	Math/Philosophy	**Electrical/Computer Engineering**	**Gate**
p AND q			
p OR q			
NOT p			
p XOR q			

Compound Proposition	**Expression in English**
$\neg p$	"It is not the case that p"
$p \wedge q$	"Both p and q"
$p \vee q$	"p or q (or both)"
$p \oplus q$	"p or q (but not both)"
$p \rightarrow q$	"if p then q" "p implies q"
$p \leftrightarrow q$	"p if and only if q"

Truth tables

English is often too ambiguous. (It doesn't clearly distinguish between $p \vee q$ and $p \oplus q$, for instance.) A ***truth table*** is an unambiguous way to show the meaning of a compound proposition. We will use them a lot. Fill in the following table!

p	q	$p \wedge q$	$p \vee q$	$p \oplus q$	$p \rightarrow q$	$p \leftrightarrow q$
T	T					
T	F					
F	T					
F	F					

Implication

Many people have trouble with $p \rightarrow q$. In English, saying "p implies q" suggests that there is a causal connection between p and q. In logic, $p \rightarrow q$ means the truth table on the right. Thus, $0=1 \rightarrow$ Brehob is POTUS has truth value T!

p	q	$p \rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T

It takes practice to get the right intuitions about $p \rightarrow q$. One useful perspective:

- If p is true, $p \rightarrow q$ says something about q.
- If p is false, $p \rightarrow q$ says nothing about q.

Another one: The only way for $p \rightarrow q$ to be false, is for p to be true, and q to be false.

- Write $p \rightarrow q$ using only $\wedge$ $\vee$ and $\neg$.

____

Examples

#1: Let's How can we express these compound propositions in terms of *p*, *q*, *r*, *s*?

English	**Compound proposition**
If it rains, I'll watch a movie and eat popcorn.	
If I don't eat popcorn, I'll eat chocolate.	
I won't eat both chocolate and popcorn unless it rains.	

r = "it'll rain" w = "I'll watch a movie" p = "I'll eat popcorn" c = "I'll eat chocolate"

#2:

Let p , q , and r be the propositions

p : You get an A on the final exam.

q : You do every exercise in the book.

r : You get an A in this class.

English	**Compound proposition(s)**
You get an A in the class, but you do not do every exercise in the book.	
Getting an A on the final exam and doing every exercise in the book is sufficient for getting an A in this class.	

A bit on logic gates. (1.2)

It is traditional in digital logic to use "1" for "T" and "0" for "F".

A	B	S	M
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

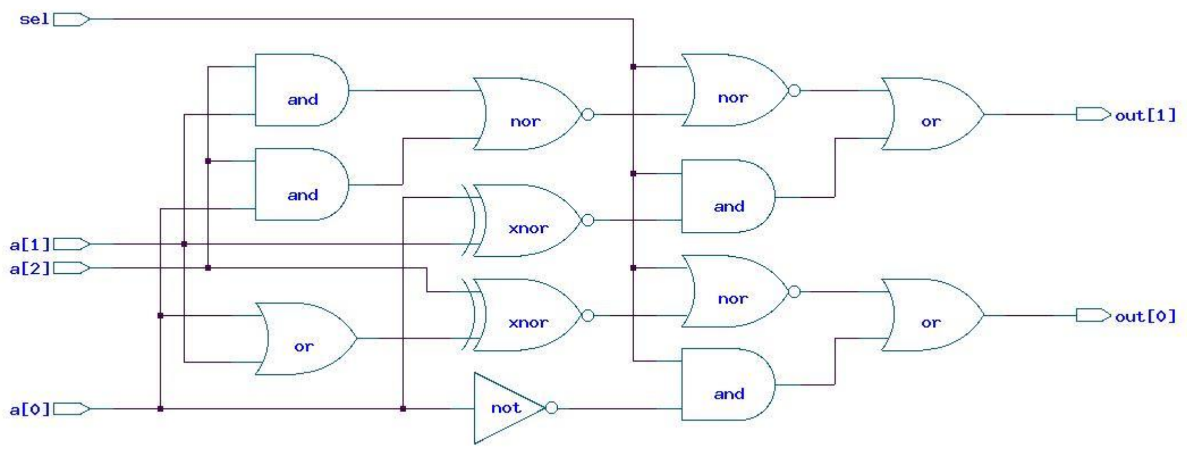
Write a truth table for the following word problem:

- Consider a device with three inputs: A, B and S as well as one output M. M should be equal to A if S=0, else M should be equal to B.

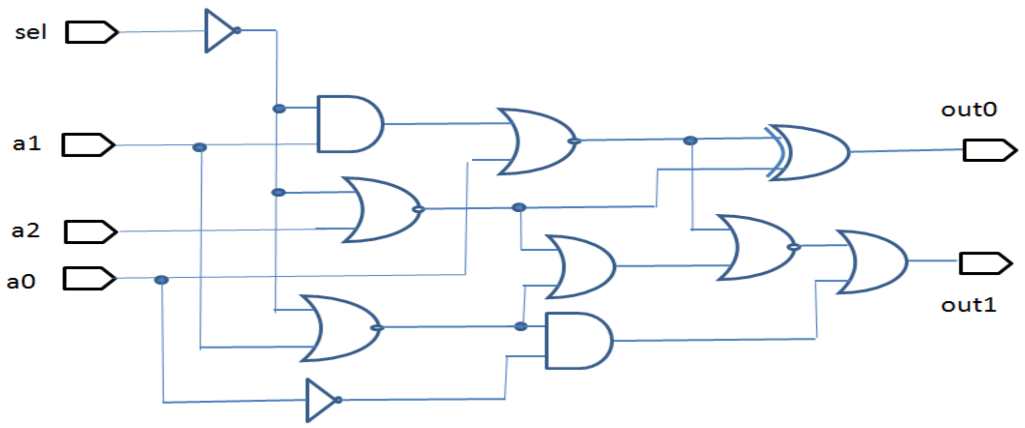
Now, can you draw gates for this? Hint: it can be done with 4 gates (2 AND, 1 OR, 1 NOT).

This device is called a multiplexor (MUX)

Here is an example of an industry supplied logic circuit that can be simplified.



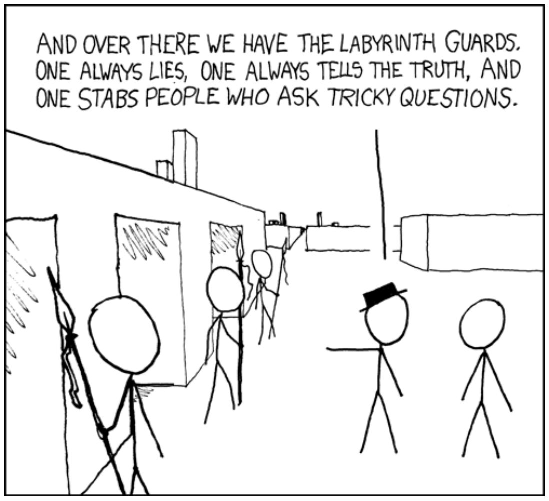
There are a number of ways to simplify this circuit, but one way is using propositional logic equivalencies. After doing so, you can get this:



- The top (unoptimized) circuit has 10 AND/OR/NOR/NAND gates, 2 XNORS and 1 inverter.
- The bottom circuit has 8 AND/OR/NOR/NAND gates, 1 XOR and 2 inverters.

Later (Wednesday…) we'll start looking a bit about how to go about proving that the two circuits are equivalent. But before we jump to that, let's take a short look at "applications" of propositional logic. We've already looked at logic gates a bit (and we'll do more today time allowing). But for now I want to jump to using propositional logic to help solve logic puzzles.

Logic puzzles as an application of propositional logic (1.2)


AND OVER THERE WE HAVE THE LABYRINTH GUARDS. ONE ALWAYS LIES, ONE ALWAYS TELLS THE TRUTH, AND ONE STABS PEOPLE WHO ASK TRICKY QUESTIONS.

Let’s play with some logic problems for a bit, and then see how we can use propositional logic to help with them.

The setup

There is this island in the middle of the ocean where there are two kinds of people: the liars and the truth-tellers. The liars always lie. Any question you ask them will be answered with a lie. The truth-tellers always answer the truth.

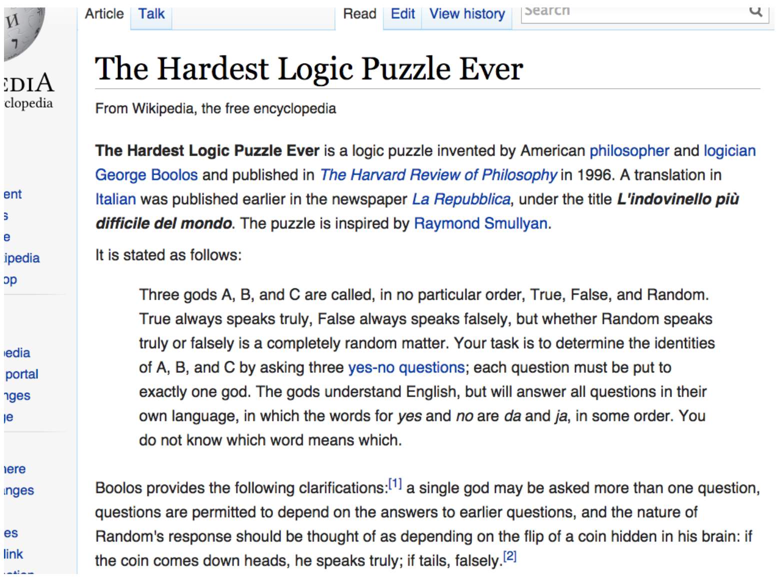
Puzzle #1

1. You're walking in this island and you meet two people A and B, from the island
2. A says “B is a truth teller”
3. B says “We are of different types”
4. Which one(s), if any, are liars?

Puzzle #2

1. Suppose that you meet three people A, B and C from the island.
2. You ask A “Are you a liar?”
3. A answers but his voice is drowned out by a clap of thunder.
4. You ask B “What did A say?”
5. B answers, “A said he is a liar”
6. C exclaims, “Don't believe what B said, he's lying”
7. C then adds “Also, A is a liar”
8. **Which ones are liars?**

We could keep going on these and make them quite hard. In fact there is one that has a solid claim to being “the hardest logic puzzle ever”



The Hardest Logic Puzzle Ever

From Wikipedia, the free encyclopedia

The Hardest Logic Puzzle Ever is a logic puzzle invented by American philosopher and logician George Boolos and published in *The Harvard Review of Philosophy* in 1996. A translation in Italian was published earlier in the newspaper *La Repubblica*, under the title ***L'indovinello più difficile del mondo***. The puzzle is inspired by Raymond Smullyan.

It is stated as follows:

> Three gods A, B, and C are called, in no particular order, True, False, and Random. True always speaks truly, False always speaks falsely, but whether Random speaks truly or falsely is a completely random matter. Your task is to determine the identities of A, B, and C by asking three yes-no questions; each question must be put to exactly one god. The gods understand English, but will answer all questions in their own language, in which the words for *yes* and *no* are *da* and *ja*, in some order. You do not know which word means which.

Boolos provides the following clarifications:[1] a single god may be asked more than one question, questions are permitted to depend on the answers to earlier questions, and the nature of Random's response should be thought of as depending on the flip of a coin hidden in his brain: if the coin comes down heads, he speaks truly; if tails, falsely.[2]

More on digital logic and its applications

Consider the number 123

1 2 3

Each *place* has a value. We normally work in base 10, so each place is 10 times bigger than the last.

In binary we work in base 2. Consider the number 10010_2 (the subscript indicates the base).

1 0 0 1 0

Recall that in digital logic, we treat "T" as "1" and "F" as "0". Consider a device that adds two one-digit binary numbers and outputs a 2 digit binary number. Let the inputs be A and B and the output be R[1:0] (where R[1:0] means R1 concatenated with R0).

```
    A
+   B
====
 R₁R₀
```

Write the truth table for this adder. R1 is to be the most significant digit (farthest to the left in the 2's place in this example) while R0 is to be the least significant digit (farthest to the right, in the 1's place). Then draw the logic gates.

A	B	R_1	R_0
0	0		
0	1		
1	0		
1	1		

The point is that we can use basic logic to do arithmetic. You may say "great, I can add two one-bit numbers". But it turns out we can use this basic idea of using logic states to represent numbers to do all kinds of math. A modern computer can easily do 5-10 billion additions of 64-bit numbers in a second! All based on this basic idea. In fact, all computers are built around this simple idea that we can use logic to do arithmetic. [i]

[i] Material for these notes are taken from previous semester's slides and the text without attribution. Other sources will be cited in-line. XKCD comics (stick figure comics) taken from xkcd.com per CC BY-NC 2.5 license. It should be clear where Wikipedia sources come from. Use of Futurama comic claimed as fair use.