**CS 5 today:** more machines!

**Final ideas...** Turing Machines and the MANY things computers can't compute...!

- This machine doesn't look all-powerful to me!

**Final projects...**
- Final tutoring hrs/labs meet this week + next
- hw12 + milestone due on Monday, 4/29
- 5 finite-state machines due as part of hw12

---

**Final project state machine**

- stop adding features + start adding print statements
- comment out print statements + start adding more features
- store a copy somewhere else!

- still broken
- it's broken
- it works
- still works

---

**An autonomous vehicle's FSM**

Fig. 9. Situational Interpreter State Transition Diagram. All modes are sub-modes of the system RUN mode (Fig 4(b)).
The data driving the state machine...

MIT's car, Talos - and its sensor suite

But are there any binary-string problems that FSMs can't solve?

FSMs can't count...

So, let's build a better machine!

State-machine limits?

Let's build a FSM that accepts strings with any # of 0s followed by the same # of 1s

011
001
11100
00110
000111
0011
01

You don't need three eyes to see some problems here!

So, let's build a better machine!

Turing Machine

Turing Machine rule: 0 ; 1 , R

try it in JFLAP...
a Turing Machine rule: \[ 0 ; 1 , R \]

**Accepted Input!**

- the input: \[ \begin{array}{c} 0 \\ 1 \\ 1 \end{array} \]

- the tape

---

a Turing Machine rule: \[ 0 ; 1 , R \]

**Rejected Input.**

- the input: \[ \begin{array}{c} 0 \\ 1 \end{array} \]

- the tape

---

an accepting state always halts -- then basks in its success!

- the input: \[ \begin{array}{c} 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{array} \]

- the tape

---

Extra: How could you change this TM to accept palindromes?

try it in JFLAP...
Can TMs compute *everything*?

Alan Turing says *yes*...

Turing called them *Logical Computing Machines*

Turing’s *Intelligent Machines*, 1948

---

**So far**, all known computational devices can compute only what Turing Machines can...

- Quantum computation
- Molecular computation
- Parallel computers
- Integrated circuits
- Electromechanical computation
- Water-based computation
- Tinkertoy computation
- *Turing machine*

---

Can TMs compute *everything*?

Alan Turing says *No!*

**ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM**

*By A. M. Turing.*

[Received 28 May, 1936.—Read 12 November, 1936.]

The “computable” numbers may be described briefly as the real numbers whose expressions as a decimal are calculable by finite means. Although the subject of this paper is ostensibly the computable numbers, it is almost equally easy to define and investigate computable functions of an integral variable or a real or computable variable, computable predicates, and so forth. The fundamental problems involved are, however, the same in each case, and I have chosen the computable numbers for explicit treatment as involving the least cumbersome technique. I hope...
There are many problems computers can't solve at all!

Perhaps this is not that surprising...

- rising sea levels
- disbelief in aliens
- losing to your own Connect4 (at 0 ply!)
- towel folding! (well, fast towel folding...)

Unprogrammable functions?

There are well-defined mathematical functions that no computer program can compute even with any amount of memory!

Functions programs

\[ f(x) = \begin{cases} 
1 & \text{if } x \text{ is odd} \\
0 & \text{if } x \text{ is even}
\end{cases} \]

\[ \text{def } \text{prog1(x): return x%2} \]

functions programs

\[ \text{int - bool programs} \]

Example programs
- Input is one integer, \( x \geq 0 \)
- Output is 0/1 (boolean or bit)

If programs look different they are different – even if they compute the same function!

\[ \text{def prog4(x): return len(str(x+42))>1} \]

...and allow ANY programs at all ~ even syntax errors

Let's match!
functions

\[ f_a(x) = 1 \]

\[ f_b(x) = \begin{cases} 
1 & \text{if } x \text{ is odd} \\
0 & \text{if } x \text{ is even}
\end{cases} \]

\[ f_c(x) = \begin{cases} 
1 & \text{if } x = 0, \ 1, \text{ or } 2 \\
0 & \text{otherwise}
\end{cases} \]

programs

```python
def prog1(x):
    return x % 2

def prog2(x):
    return x < 3

def prog3(x):
    return 1

def prog4(x):
    return len(str(x + 42)) > 1

def prog5(x):
    return x in [0, 1, 2]

def prog6(x):
    if x < 2:
        return x
    else:
        return prog6(x - 2)
```

1. Match each program with the function it computes.
2. There are three different functions on the left side -- how many different programs are in the right side?

Quiz!
Name(s): __________________________

Uncomputable functions?

There are well-defined mathematical functions that no computer program or TM can compute even with any amount of memory!

**Why?**

There are many more of these ... mathematical functions than these!

Programs are "like" integers...

```python
def alien(x):
    if x == 42:
        return True
    else:
        return not alien(x + 1)
```

For each program, there is an integer. This is true: but how?

Programs are "like" integers...

from programs to ints ~ and back...
functions vs. programs!

the Reals from 0 to 1 \( \mathbb{R} \)  
Positive integers \( \mathbb{N} \)

int-bool functions are *real numbers!*
(integer predicates)

For each real #, there is a function.
For each function, there is a real #.

This is true: but how?

functions vs. programs!

for each real # ...
there's a function

any real # from 0-1
one bit at a time...
1 2 3 4 5 6 7 8

\( r_1 = .11111111 \ldots \)
\( r_2 = .10101010 \ldots \)
\( r_3 = .11000000 \ldots \)
\( r_4 = .01101010 \ldots \)

6 and 8 are not prime, so these digits are 0
2 and 3 are prime, so these digits are 1
**To infinity - and beyond**

strictly larger

the Reals from 0 to 1 > Positive integers

**Uncountably infinite** > **Countably infinite**

\[ R > N \]

**Different infinities !?**

There are infinitely many functions + programs...

... but not all infinities are created equal!

Two sets have equal size if their elements have a one-to-one matching

This matching is called a **bijection**.

These sets have the same **cardinality**

**Cantor and "friends"**

Georg Cantor

Leopold Kronecker 1823-1891

Ludwig Wittgenstein 1889-1951

David Hilbert 1862-1943

One-to-one mappings also define equal-sized infinite sets, but the results can be surprising!

Positive evens \( E = \{ 2, 4, 6, 8, 10, \ldots \} \)

Positive integers \( N = \{ 1, 2, 3, 4, 5, \ldots \} \)

ALL integers \( Z = \{ \ldots, -3, -2, -1, 0, 1, 2, 3, \ldots \} \)

\( E \) and \( N \) and \( Z \) all have the same size!
Let's get real

The reals (from 0 to 1)

\{ 0.0 \leq r \leq 1.0 \}

Are these sets equal-sized?

The integers (positive)

\{ 1, 2, 3, 4, 5, \ldots \}

Why is this NOT a valid matching?

(Many) more functions than programs!

Int-bool functions are uncountable

Integer predicates

\text{Uncountably infinite} \quad \text{Countably infinite}

For each real #, there is a function.
For each function, there is a real #.

This is true: How? Why?

Cantor Diagonalization

There are always real #s missing from any list!

The Reals from 0 to 1

Positive integers

\begin{array}{c}
\hline
\text{real #s are always missing} \quad \rightarrow \\
\text{regardless of the matching...} \\
\hline
\end{array}

r_0 =

r_1 = .33333333333... \quad \rightarrow 1

r_2 = .42424242424... \quad \rightarrow 2

r_3 = .31415942653... \quad \rightarrow 3

r_4 = .09090909090... \quad \rightarrow 4

r_5 = ... \quad \rightarrow 5

This margin is too small to hold my marvelous proof!
It overlaps incompletely – *but also goes beyond...!*

**Well-defined mathematical functions**

**R**

**Wow! Things out here are **indescribable**

**N**

**Programs**

**kc**

**hc**

**Human-describable things...**

**Next time!**

So, what is all this stuff out here?!?

Wow! Things out there are indescribable
a Turing Machine rule: \( 0 ; 1 , R \)

If it gets here, it succeeds.

If a transition is missing, the input Fails!

Try it in JFLAP...