ALIEN INVASION!!!

Claremont (AP): A party at a private college here was disrupted when uninvited aliens burst through the gates. “Every year, we celebrate Long Tall Penguins,” explained an angry student. “We get together, dress like the stuffiest professors, and chip bits off an iceberg to cool our drinks. This year, just as we were about to chill the mackerel, two strange alien creatures ran into the courtyard, picked everyone up, and took turns stacking us in piles.”

But another student claimed that the aliens were just misunderstood. “They love to play Connect 4, and since we were wearing black and white clothes, they thought we were playing pieces. They stacked us up in a 5-ply lookahead formation. It was fun!”

According to police, no charges will be filed because the aliens are not subject to Solar jurisdiction.
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!

• hw12 + milestone due on Monday, 12/4
• 5 finite-state machines due as part of hw12
Final project state machine

still broken

it's broken

still works

stop adding features + start adding print statements

comment out print statements + start adding more features

store a copy somewhere else!
State-machine *limits*?
Are there limits to what FSMs can do?

MIT's car, *Talos*

they can't necessarily drive safely...

MIT's car, *Talos*
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?
State-machine *limits*?

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

### Rejected
- 011
- 001
- 11100
- 00110

### Accepted
- 000111
- 0011
- 01
- $\lambda$
State-machine *limits*?

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

FSMs "can't count" at least, not arbitrarily high.

- **Accepted**
  - 000111
  - 0011
  - 01
  - \(\lambda\)

- **Rejected**
  - 011
  - 001
  - 11100
  - 00110
FSMs can't count...

So, let's build a better machine!

Turing Machine
a Turing Machine rule:

\[
0 \, \rightarrow \, 1, R
\]

READ  WRITE  MOTION

an accepting state \textbf{always halts} -- then basks in its success!

if a transition is missing, the input FAILS!

try it in JFLAP...
a Turing Machine rule:

\[ 0 ; 1 , R \]

try it in JFLAP...

if a transition is missing, the input FAILS!

Accepted Input!
A Turing Machine rule:

- **the tape**
- **R/W head**
- **input**

- **READ**
- **WRITE**
- **MOTION**

If a transition is missing, the input **fails**!

An accepting state **always halts** -- then basks in its success!

The input is **010**.

**Rejected Input.**

A Turing Machine rule:

- **0 ; 1 , R**

Try it in JFLAP...
a Turing Machine rule:

\[
\begin{array}{c}
0 ; 1 , R
\end{array}
\]

try it in JFLAP...
a Turing Machine rule:

```
0 ; 1 , R
```

try it in JFLAP...

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

"blanks" are everywhere else

if a transition is missing, the input FAILS!
CS 5 spokesperson of the day!

I want a 1950's network!

The Imitation Game (2014)

Your rating: ★★★★★★★★★★

8.4

Ratings: 8.4/10 from 5,470 users  Metascore: 71/100
Reviews: 46 user | 108 critic | 31 from Metacritic.com

Alan Turing
What inputs are accepted in general?

Extra: How could you change this TM to accept palindromes? (thought experiment and ex. cr.)
Can TMs compute everything?

Alan Turing says yes...

Turing called them *Logical Computing Machines*

Turing’s *Intelligent Machines, 1948*

http://www.alanturing.net/turing_archive/archive/l/l32/L32-005.html
*So far*, all known computational devices can compute only what Turing Machines can...

Quantum computation

Molecular computation
http://www.arstechnica.com/reviews/2q00/dna/dna-1.html

Parallel computers

Integrated circuits

Electromechanical computation

Water-based computation

Tinkertoy computation

Turing machine
Alan Turing
1912-1954

Enigma machine ~ The axis's encryption engine

1946

2007
Bletchley Park
Alan Mathison Turing
1912-1954

Father of Computer Science
Mathematician, Logician
Wartime Codebreaker
Victim of Prejudice

“Mathematics, rightly viewed, possesses not only truth but supreme beauty; a beauty cold and austere like that of sculpture.” - Bertrand Russell
2012: Turing Celebration

http://aturingmachine.com/examplesSub.php

https://www.youtube.com/watch?v=aBToqFJLrl4
Turing Test Extra Credit: Convince the examiner that he's a computer.

You know, you make some really good points. I'm ... not even sure who I am anymore.
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, \textit{fast} towel folding…)
There are functions that no computer program can compute even with any amount of memory!

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

```python
def progl1(x):
    return x % 2
```
Unprogrammable functions?

There are many more of these …

functions

mathematical functions

programs

There are even with any amount of memory!

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

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

int - bool

\[ f_A(x) = 1 \]

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

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

These are "int-bool" mathematical functions.

They're also sometimes called "integer predicates."

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

...even if you restrict your functions to be totally discrete!
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!

```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
```

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

Let's match!
functions

\[ f_A(x) = 1 \]

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

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

Worksheet!

\[ f_B(x) = 1 \text{ if } x \text{ is odd} \]
\[ 0 \text{ if } x \text{ is even} \]

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?

How - or why - is the set of all functions larger than the set of all programs?

Worksheet!
There are three different functions on the left side -- how many different programs are in the right side?

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?

How - or why - is the set of all functions larger than the set of all programs?
Uncomputable functions?

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

Why?

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

R vs. N
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.

For each integer, there is a "program."

\[
\begin{align*}
\text{This is true:} \\
\text{but how?}
\end{align*}
\]

at least a string! some have syntax errors...
Programs are integers (and vice-versa)

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

Every program is a **string**.

Every string is just a sequence of **bits**

Every sequence of bits is also an **int**!
from **progs** to **ints** ~ and **back**...

```python
# Python converters from int to program (and back)
#

def prog( i ):
    """ return the program whose int is i """
    # convert to a string (just a base-128 int!)
    if i <= 0: return ''
    last_char = chr( i%128 )
    return prog( i/128 ) + last_char

def intify( prog ):
    """ return the int whose program is prog """
    # convert to an int (just interpret as base-128!)
    if prog == '': return 0
    last_char = prog[-1]
    return 128*intify( prog[:-1] ) + ord( last_char )

# to run a string: (1) code = compile( p, 'str', 'exec' )
# (2) exec(code)
```