The Legacy of Alan Turing

Based on slides by team of CMU's 15-251
Sanjeev Arora and Bernard Chazelle

The HELLO assignment

Write a JAVA program to output the words "Hello World!" on the screen and halt.

Space and time are not an issue.

PASS for any working HELLO program, no partial credit.

Grading Program?

A grading program G must be able to take any Java program P and grade it.

G(P)= { Pass, if P prints only the words "Hello World!" and halts.
Fail, otherwise.

How exactly might such a program work?

What kind of program could a student who hated his/her TA hand in?

Nasty Program

```
n:=0;
while (n is not a counter-example
     to the Riemann Hypothesis) {
     n++;
}
print "Hello World!";
```

The Riemann Hypothesis

- Considered by many mathematicians to be the most important unresolved problem in pure mathematics
- Conjecture about the distribution of zeros of the Riemann zeta - function
- 1 Million dollar prize offered by Clay Institute

Nasty Program

```
n:=0;
while (n is not a counter-example
     to the Riemann Hypothesis) {
     n++;
}
print "Hello World!";
```

The nasty program is a PASS if and only if the Riemann Hypothesis is false.

A TA nightmare: Despite the simplicity of the HELLO assignment, there is no program to correctly grade it!

We will develop the ideas needed to prove this.

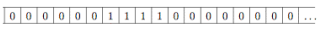
The theory of what can and can't be computed by an ideal computer is called **Computability Theory**

Turing develops a model of computation

- Wanted a model of human calculation.
- Wanted to strip away inessential details.
- What are the important features?

  - Paper (size? shape?)
  - The ability to read or write what's on the paper.
  - The ability to shift attention to a different part of the paper
  - The ability to have what you do next depend on what part of the paper you are looking at and on what your state of mind is
  - Limited number of possible states of mind.

A variant on Turing's model: Turing-Post programs



- 1 dimensional unlimited scratchpad ("infinite")
- Only symbols are 0/1 (tape has a finite number of 1s)
- Can only scan/write one symbol per step
- Legal instructions →

PRINT 0
PRINT 1
GO RIGHT
GO LEFT
GO TO STEP i if 1 SCANNED
GO TO STEP i if 0 SCANNED
STOP

What does this program do?

1. PRINT 0
2. GO LEFT
3. GO TO STEP 2 IF 1 SCANNED
4. PRINT 1
5. GO RIGHT
6. GO TO STEP 5 IF 1 SCANNED
7. PRINT 1
8. GO RIGHT
9. GO TO STEP 1 IF 1 SCANNED
10. STOP

Example: What does this program do?

1. PRINT 0
2. GO RIGHT
3. GO TO STEP 1 if 1 SCANNED
4. GO TO STEP 2 if 0 SCANNED

T-P "programming language" has these instructions

1. PRINT 0
2. PRINT 1
3. GO RIGHT
4. GO LEFT
5. GO TO STEP i if 1 SCANNED
6. GO TO STEP i if 0 SCANNED
7. STOP

What kind of computations can be performed in this model?

Amazing fact about this mickey-mouse model:

It is equivalent to Java!!

In fact, all of the following are equivalent computational models:

- Turing-Post programs
- Turing machines (which we haven't defined precisely)
- Pseudocode (which we haven't defined precisely
- Python
- C++

- Equivalent = If something can be computed in one of these models, it can also be computed in the others.
- = what can be computed on a digital computer (with no bound on memory)

CHURCH-TURING THESIS

This model captures the notion of computation.

Anything "computable" is computable by Turing machine.

Any "reasonable, physically realizable" model of computation can be simulated on a Turing machine "efficiently".

Any well-defined procedure that can be grasped and performed by the human mind and pencil/paper, can be performed on a conventional digital computer with no bound on memory.

Turing's next great insight: duality between programs and data

- A program can be viewed either as
  - a program whose execution does whatever that program is designed to do -- P
  - or as plain data --the code of the program -- code(P).

"Code" for a program

= Binary Representation

Many conventions possible (e.g., ASCII)
One possible convention:

Code	Instruction
000	PRINT 0
001	PRINT 1
010	GO LEFT
011	GO RIGHT
1010...01 (i)	GO TO STEP i IF 0 IS SCANNED
1101...10 (i)	GO TO STEP i IF 1 IS SCANNED
100	STOP

P ⟷ Code (P)

Start with 1, end with 111

P vs code (P)

P:

1. PRINT 0
2. GO RIGHT
3. GO TO STEP 1 if 1 SCANNED
4. GO TO STEP 2 if 0 SCANNED

Code(P)?

Code	Instruction
000	PRINT 0
001	PRINT 1
010	GO LEFT
011	GO RIGHT
1010...01 (i)	GO TO STEP i IF 0 IS SCANNED
1101...10 (i)	GO TO STEP i IF 1 IS SCANNED
100	STOP

Start with 1, end with 111

Turing's next great insight: duality between programs and data

- A program can be viewed either as
  - a program whose execution does whatever that program is designed to do -- P
  - or as plain data --the code of the program -- code(P).
- If code(P) can be viewed as data, it can be the input to another program!

Duality leads to Universality!

- There is a universal Turing machine **U**
  - On input code(P) and an input **x**, **U** outputs the same thing as **P** does on input **x**
  - At each step it decodes which operation **P** would have performed and simulates it.
- One Turing machine, the Universal TM, is enough!
  - Basis for modern stored-program computer
    - Von Neumann studied Turing's UTM design

You be the universal computer

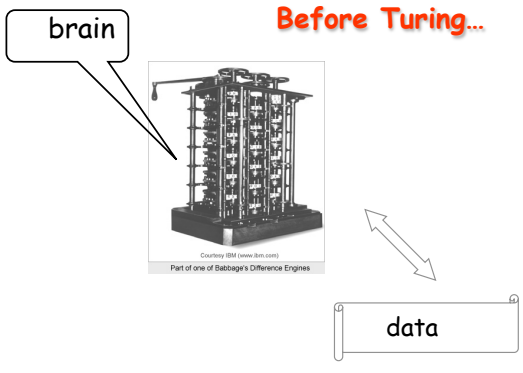
10111101000011001111111 code(P) followed by input

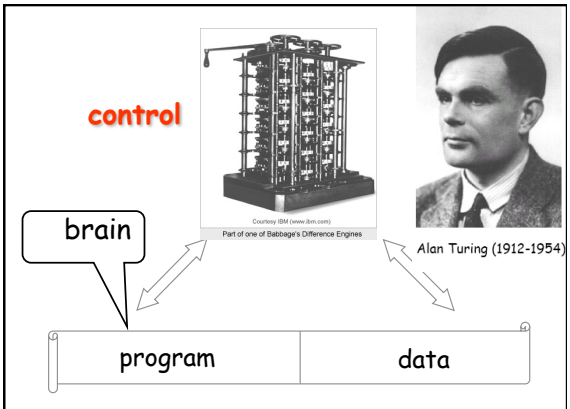
1 011 11010 001 100 111 111

Code	Instruction
000	PRINT 0
001	PRINT 1
010	GO LEFT
011	GO RIGHT
1010...01 (i)	GO TO STEP i IF 0 IS SCANNED
1101...10 (i)	GO TO STEP i IF 1 IS SCANNED
100	STOP

Start with 1, end with 111

Before Turing...





control

brain

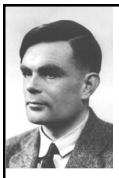
program | data

Print this | Let 'em eat cake

Let 'em eat cake

Fishing ...

Fishing ...



Fishing manual

program | data

control

knows nothing

program | data

control = hardware

knows nothing

program | data

programs = software

Duality leads to Universality!

- There is a universal Turing machine **U**
  - On input code(P) and an input **x**, **U** outputs the same thing as **P** does on input **x**
  - At each step it decodes which operation **P** would have performed and simulates it.
- One Turing machine, the Universal TM, is enough!
  - Basis for modern stored-program computer
    - Von Neumann studied Turing's UTM design

- "existence of software industry lemma" --Scott Aaronson

The Halting Problem (or Universal Termination Detector)

Is there a program HALT such that:

HALT(code(P),x) = True, if P(x) halts
HALT(code(P),x) = False, if P(x) does not halt

1. GO RIGHT
2. GO TO STEP 1 IF 0 IS SCANNED
3. GO TO STEP 1 IF 1 IS SCANNED
4. STOP

THEOREM: There is no program to solve the halting problem (Alan Turing 1937)

We' ll use a "proof by contradiction"

"When something' s not right, it' s wrong."

Bob Dylan

THEOREM: There is no program to solve the halting problem (Alan Turing 1937)

Suppose a program HALT existed that solved the halting problem.

HALT(code(P),x) = True, if P(x) halts
HALT(code(P),x) = False, if P(x) does not halt

We will call HALT as a subroutine in a new program called CONFUSE.

CONFUSE

```
CONFUSE(code(P))
{ if (HALT(code(P),code(P))=True)
      then loop forever;        // i.e., we don' t halt
  else exit;                    // i.e., we halt

}
```

Does CONFUSE(CONFUSE) halt?

CONFUSE <P>=code(P)

```
CONFUSE(code(P))
{ if (HALT(code(P),code(P))=True)
      then loop forever;        // i.e., we dont halt
  else exit;                    // i.e., we halt
}
```

Suppose CONFUSE(<CONFUSE>) halts:
then HALT(<CONFUSE>,<CONFUSE>) = TRUE
⇒ CONFUSE will loop forever on input <CONFUSE>

Suppose CONFUSE(<CONFUSE>) does not halt
then HALT(<CONFUSE>,<CONFUSE>) = FALSE
⇒ CONFUSE will halt on input <CONFUSE>

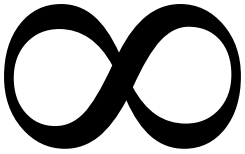
CONTRADICTION

Alan Turing (1912-1954)

Theorem: [1937]

There is no program to solve the halting problem

Similar to the fact that there are different kinds of infinity



Alan Turing (1912-1954)

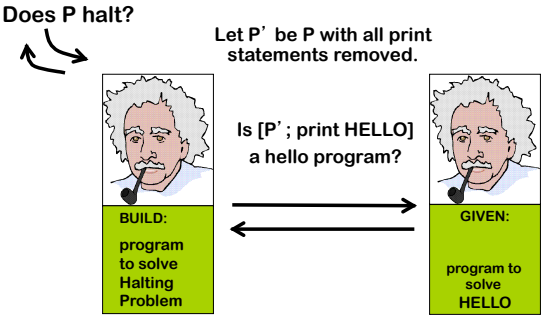
Theorem: [1937]

There is no program to solve the halting problem

"This is a first, fundamental impossibility result for computation -- a natural problem that can't be solved computationally. And starting with this result, impossibility spreads like a shock wave through the space of problems…" Kleinberg/Papadimitriou

- No Hello World Tester

There is no program to grade the HELLO assignment



"This is a first, fundamental impossibility result for computation -- a natural problem that can't be solved computationally. And starting with this result, impossibility spreads like a shock wave through the space of problems…" Kleinberg/Papadimitriou

- No Hello World Tester
- **No automated checking of pretty much any property of software!**