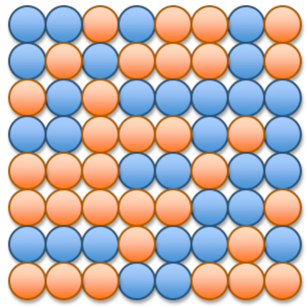


UNIT 14C

The Limits of Computing: Non-computable Functions

(and the Future of Computing)

Problem Classifications

- Tractable Problems
  - Problems that have reasonable, polynomial-time solutions
- Intractable Problems
  - Problems that may have no reasonable, polynomial-time solutions
- Noncomputable Problems
  - Problems that have no algorithms at all to solve them

Program Termination

- Can we determine if a program will terminate given a valid input?
- Example:

```
def mystery1(x):
    while (x != 1):
        x = x - 2
```

  - Does this algorithm terminate when x = 15?
  - Does this algorithm terminate when x = 110?

Another Example

```
def mystery2(x):
    while (x != 1):
        if x % 2 == 0:
            x = x // 2
        else:
            x = 3 * x + 1
```

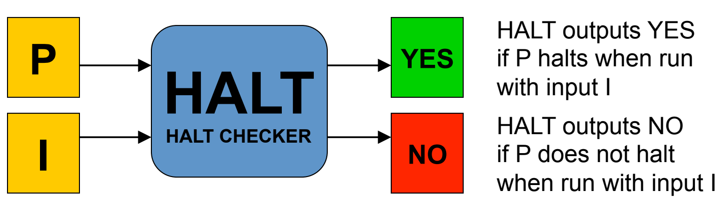
  - Does this algorithm terminate when x = 15?
  - Does this algorithm terminate when x = 110?
  - Does this algorithm terminate for any positive x?

The Halting Problem

- Does a <u>universal</u> program HALT exist that can take <u>any</u> program P and <u>any</u> input I for program P and determine if P terminates/ halts when run with input I?
- Alan Turing showed that such a single, universal program HALT cannot exist.
  - This is known as the Halting Problem.

Proof by Contradiction

- Assume a program HALT exists that requires a program P and an input I.
  - HALT determines if program P will halt when P is executed using input I. No assumptions are made on how it does this. Anything is possible.



  - **We will show that <u>HALT cannot exist</u> by showing that if it did exist we would get a logical contradiction.**

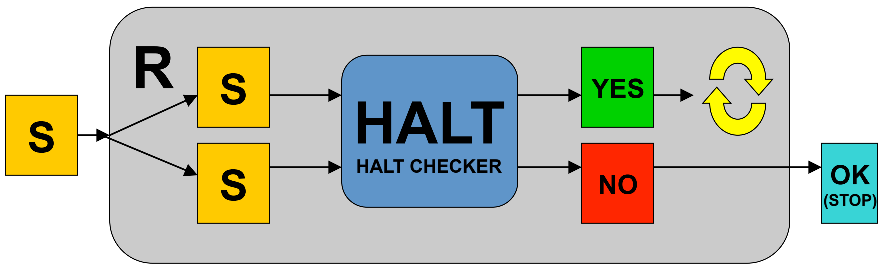
Can a program have as its input a program?

- A compiler is a program that takes as its input a program that needs to be translated from a high-level language (e.g. Python) to a low-level language (e.g. machine language).
  - In general, a program can process any data, so it can have a program as its input to process.
- Deep thought: Could a compiler compile itself since a compiler is a program? (Yes!)

Proof (cont'd)

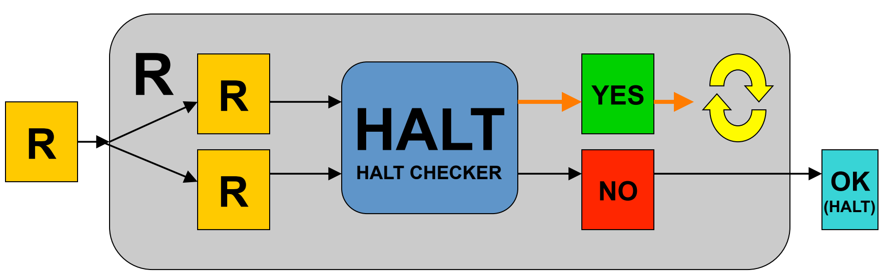
- Let R be a program that takes input S, where S is a program.
- R asks the halt checker HALT what happens if S runs with itself as input?
- If HALT answers that S will halt if it runs with itself as input, then R goes into an infinite loop (and does not halt).
- If HALT answers that S will not halt if it runs with itself as input, then R halts.

How R Works



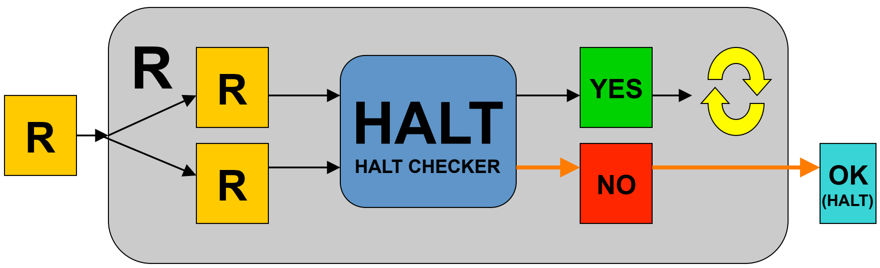
R gets evil

- What happens if R tests <u>itself</u>?
  - If HALT answers yes (R halts), then R goes into an infinite loop and does not halt.



R gets evil (cont'd)

- What happens if R tests itself?
  - If HALT answers no (R does not halt), then R halts.



A Python Way To Look At It

```
def r():
    if halt(r) == True:
        while True: #loop 4ever
            None
    else:
        return None #halt
```

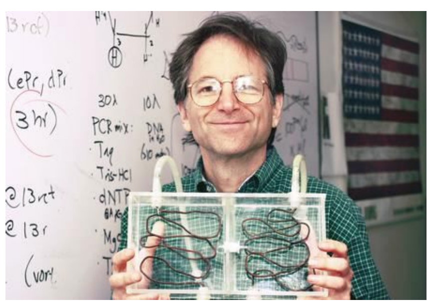
Contradiction!

- No matter what HALT answers about R, R does the opposite, so HALT can never answer the halting problem for the specific program R.
  - Therefore, a universal halting checker HALT cannot exist.
- Conclusion: We can <u>never</u> write a computer program that determines if ANY program halts with ANY input.
  - It doesn't matter how powerful the computer is.
  - It doesn't matter how much time we devote to the computation.

Contradiction in Real Life

The Future of Computing?
DNA Computing

- Use of DNA strands to compute solutions quickly.
- Computing with DNA
  by Leonard Adleman (UC Berkeley)
  - Demonstrated the use of DNA to solve a small instance of the Hamiltonian path problem.
  - DNA sequences consist of the letters A,C,T,G representing the bases adenine, thymine, guanine, and cytosine.
- Adleman demonstrated the use of DNA to solve a Hamiltonian Path problem with 7 cities in 1998.
  - The Hamiltonian Path problem is NP Complete.



Leonard Adleman with his DNA-based computer, which has solved a logic problem that no person could complete by hand, setting a new milestone for this infant technology. (Photo by Irene Fertik)

The Future of Computing?
Quantum Computing

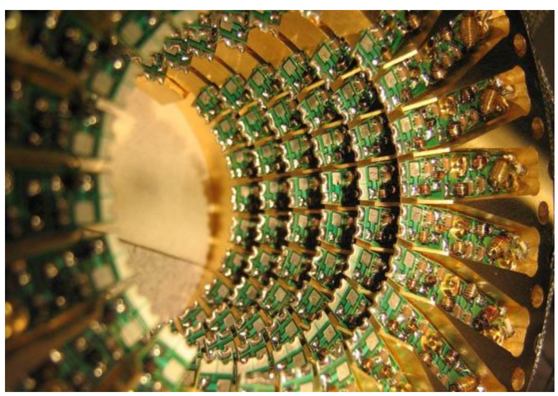
- A subatomic particle has spin (up or down). In quantum physics, particles can be in a state defined by *superposition* (up and down).
  - Using quantum mechanics, a quantum computer can do computations simultaneously since particles can be in two states at once.
  - This only holds as long as we don't interfere or observe these particles. If we do, then the particles will make a random decision and choose one of the two states. (*decoherence*)

Qubits

- In a classic computer, basic information is stored in bit form. A bit can only be in one of two states at any given time.
- In a quantum computer, basic information is stored in a qubit which can be in the states 0 and 1 at the same time (with some probability for each).
- A 4-qubit quantum computer can store 16 separate computations at the same time.
  - This improvement grows exponentially as the size of the quantum computer grows.

Quantum Computing and RSA

- Peter Shor (at AT&T Bell Labs in 1994) described an algorithm that could factor a number that was the product of two prime numbers in polynomial time using a quantum computing model.
  - This algorithm could be used with a quantum computer (once developed) to crack the RSA encryption algorithm.
- In 2001, IBM demonstrated a 7-qubit quantum computer to factor the number 15 into the prime values 3 and 5.
- 2017: IBM makes 20 qubit quantum computing machine available as a cloud service



D-Wave Systems “demonstrated” a 28-qubit quantum computer in November 2007 at a SC07 (a supercomputing conference).

What's Next?

- Will we eventually prove that P = NP or P ≠ NP?
- Will the computers for the next generation be made up of DNA or quantum particles rather than silicon?
- Will robots eventually replace humans as the dominant race due to their superior intelligence?
- Will humans become more and more robotic as they evolve?

PLEASE COMPLETE YOUR FACULTY COURSE EVALUATION (FCE) AND TA EVALUATION!