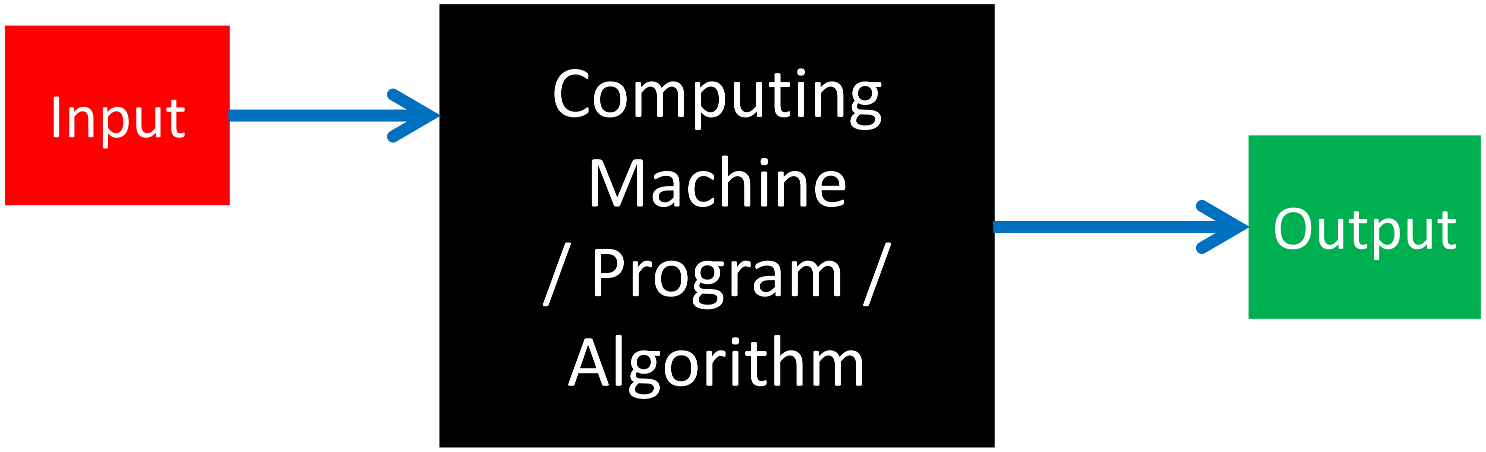
CS3102

September 6

Recall: “Carnot Engine” of computing

- What “type” is the input?
  - String (bit strings)
- What “type” is the output?
  - String
- What have we put in the black box so far?
  - Circuits of logic gates
  - Python/Java/C++
  - X8086
  - NAND-CIRC



What goes in here?

What do we compute on?

- Always strings! (that represent other things)
  - Images
  - DNA
  - Web pages
  - Temperature
- To compute on non-string things, we need a *representation scheme*
  - Function $E: U \to \{0,1\}^*$
    - E should be one-to-one

Model of Computing

- To define a model of computing, we need:
  - A way to give it input
    - Pre-defining the values of the “input” gates
  - A way to get output out of it
    - Pre-defined some gates as “output” gates, and when execution stopped, their values defined the output
  - How do we go from input to output?
    - Execution model: deterministic (non-random)
    - For any gate where all of its incoming wires had defined values, its value becomes the result of applying a particular operation to the values its incoming wires
  - Representation
    - Wires, gates, gates have labels (AON: AND/OR/NOT/INPUT/OUTPUT; NAND:NAND/INPUT/OUTPUT)

Showing AON=NAND

- Anything I can do with AON I can also do with NAND
  - Vice-versa
- Two models of computing are “equivalent” if every function you can implement with one of them you can also implement with the other
- Find a way of converting instances of one model into instances of the other such that we compute the same function
- To go from AON to NAND:
  - Do AND with NANDs
  - Do OR with NANDs
  - NOT with NANDs
- To go from NAND to AON:
  - Do NAND with ANDs/Ors/NOTs

Countable vs. Uncountable

- Countable:
  - The cardinality is less than or equal to that of the natural numbers
  - It's either:
    - Finite
    - It has a bijection with $\mathbb{N}$
- Uncountable:
  - It's bigger than $\mathbb{N}$
  - It's both:
    - Infinite
    - Has no bijection with $\mathbb{N}$

To show uncountable

- Diagonalization
- Show a 1-1 mapping from a known uncountable set to this one
  - S is uncountable
  - $f: S \to T$
  - f is one-to-one
  - $|S| \leq |T|$

Diagonalization

- A special kind of proof by contradiction
- Structure:
  - Observe our set is infinite
  - Toward a contradiction, suppose the set is countable, meaning it has bijection with $\mathbb{N}$
    - This means we can list all of the elements
  - To find a contradiction that the bijection exists we show that no matter how you might try to list all the elements, your list MUST be incomplete
    - Constructing an element we can guarantee is different from everything in the list
      - Following a “diagonal”

$$|\{0,1\}^\infty| > |\mathbb{N}|$$

$\mathbb{N}$

0	0	0	0	0	0	0	0		...
1	1	0	1	0	1	0	1		...
2	0	0	0	0	1	0	1		...
3	1	1	1	0	1	1	1		...
4	1	1	1	0	1	0	1	1	...

$x = 11110$

$|\{infinite\ python\ programs\}| > |\mathbb{N}|$

- Consider the set of a “infinitely-long” python programs
  - An infinitely-long python program is a string where there are an infinite number of prefixes that are valid python programs

```
x=1
x=2
x=3
x=4
x=5
x=6
…
```

```
while(2<3):
   print(“hello”)
print(“hello”)
print(“hello”)
print(“hello”)
print(“hello”)
print(“hello”)
…
```

```
while(2<3):
   print(“hello”)
print(y)
print(“hello”)
print(“hello”)
print(“hello”)
print(“hello”)
…
```

$|\{infinite\ python\ programs\}| > |\mathbb{N}|$

- Consider the set of a “infinitely-long” python programs
  - An infinitely-long python program is a string where there are an infinite number of prefixes that are valid python programs