

CS3102 Theory of Computation

www.cs.virginia.edu/~njb2b/cstheory/s2020

Warm up:

How does a Human “compute”?

What are the resources available to a human to do computing?

- Learning/memorizing subroutines
- Brain (is like a cpu)
- Brain also has short term memory
- Senses (input), speech, writing (output)
- Use writing/utensils to remember things
- Pondering (using a “working memory”)

Most important parts (according to Nate)

- CPU
 - Circuits of transistors
 - Brain
- RAM
 - Limited memory
 - Working memory
- HDD/SSD
 - Large memory
 - “plentiful” memory, infinite memory
 - Documents, notes, etc.

Issues and Solutions

- What were the limitations of circuits?
 - ~~Actually infinite inputs (not relevant to us)~~
 - No looking back!
 - Change the machine mid-process
 - Limited storage, bigger inputs require more memory for some functions
 - Larger output space (only 0 or 1)
 - Non-determinism: no communication among parallel paths
 - Outside the scope of this semester
 - Alternation
- How can we overcome those?
 - You can look backwards!
 - Lots of / Plentiful / enough memory: infinite!
 - Make machines that can play the roll of another machine, compute machines (macros)
 - Execution model that allows for long strings to be outputs

Characterizing What's computable

- Things that are computable by FSA (NFA/DFA):
 - Functions that don't need “unbounded memory”
 - Languages expressible as Regular Expressions
- Things that aren't computable by FSA:
 - Things that require more than finitely many states
 - Intuitive example: Majority

Majority with FSA?

- Consider an inputs with lots of 0s

000...0000 111...1111
×49,999 ×50,000

000...0000 111...1111
×50,000 ×50,000

000...0000 111...1111
×50,000 ×50,001

- Recall: we read 1 bit at a time, no going back!
- To count to 50,000, we'll need 50,000 states!

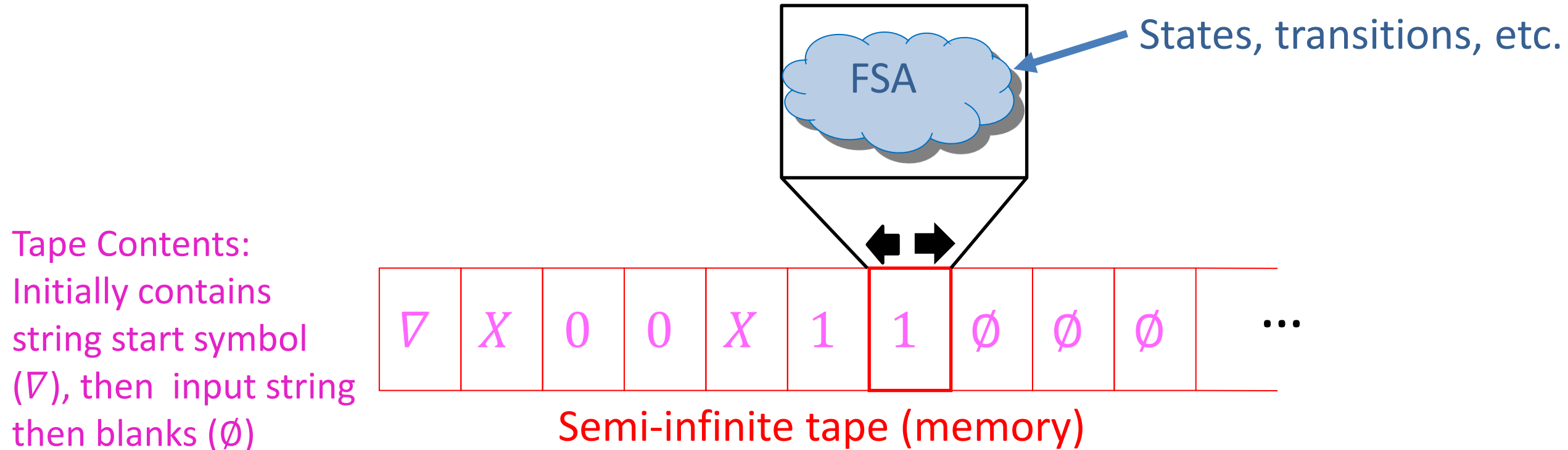
Machine Configurations

- **Configuration:** All the information needed to represent the “status” of an execution
 - Circuit: the circuit itself, all variables and their values
 - Automaton: the implementation of the automaton, currently active state(s)
 - Java program: the values of all the variables, the contents of all the data structures, call stack.
 - Solution: have a model with an unbounded number of potential configurations

Turing Machines

- FSA:
 - Finite number of states,
 - read-once input,
 - transition using input bit and state(s)
- Turing Machine:
 - Finite number of states,
 - ~~– Read-once input,~~
 - Semi-infinite tape (memory)
 - Transition using “current symbol” on tape
 - Can overwrite current symbol, move left/right on tape

Turing Machine



Operation:

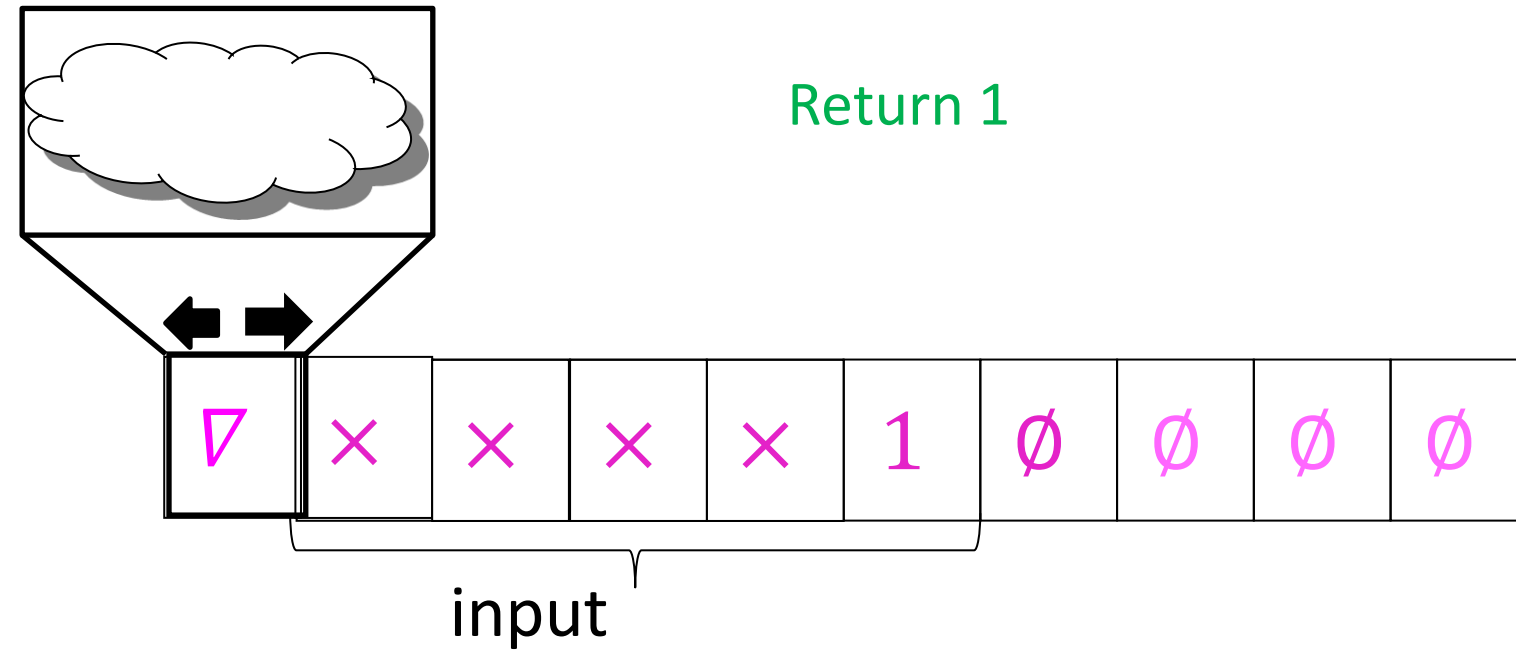
transitions outgoing from each state match on current character on the tape, when transitioning you can overwrite that character and move which cell you're reading

Return 1 if we enter the accept state, Return 0 if we enter the reject state

Example Turing Machine

- For infinite MAJ
- For each “mark off” each 0 with the next 1
- If we can mark all 0s with leftover 1s, accept

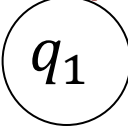
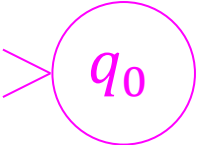
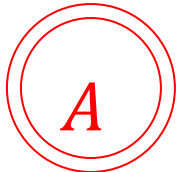
Input: 01101

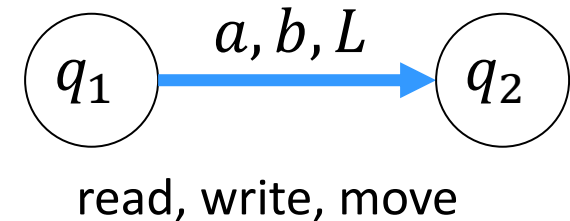


1. Start from beginning, move right while looking for a 0,
2. If you find a 0, mark it, go to the beginning,
3. Move right while looking for a 1
4. If you find a 1, mark it, go to the beginning
5. Repeat 1-4 until no more 0s or no more 1s
6. Go back to beginning, return 1 if we can find more 1s
7. Else return 0

Turing Machine

Basic idea: a **Turing Machine** is a finite state automaton that can optionally read from/write to an infinite **tape**.

- Finite set of states: $Q = \{q_0, q_1, q_2, \dots, q_k\}$ 
- Input alphabet: Σ
- **Tape** alphabet (includes \emptyset, ∇): Γ
- **Transition** function: $\delta: Q \times \overset{\text{Read}}{\Gamma} \rightarrow Q \times \overset{\text{Write}}{\Gamma} \times \overset{\text{Move}}{\{L, R, S, H\}}$
- **Initial** state: $q_0 \in Q$ 
- **Final** states: $F \in Q$ 



Turing Machine is $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$

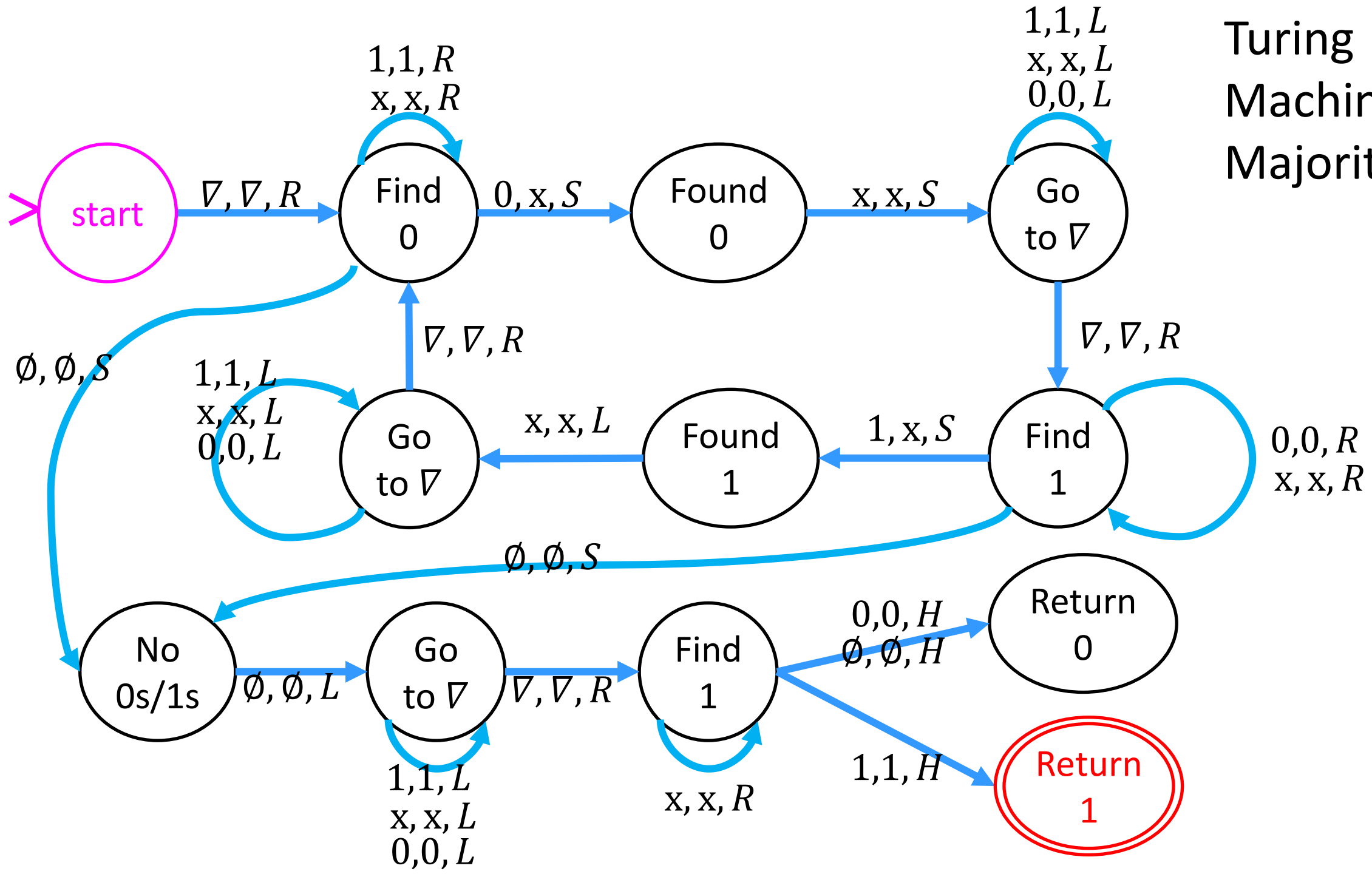
Turing Machine Execution

- Start in the initial state with the “read head” at the start of the tape. The input string follows that, then infinitely many blanks
- Look at the current state and character under the “read head”, then transition to a new state, overwrite that character, and move the “read head”
- Continue until the movement instruction is “halt”
- Output: two models:
 - Decision problems: if you’re in a final state then return 1, else return 0
 - Function: All the contents remaining on the tape (except for ∇ and \emptyset)

Let's Draw One!

1. Start from beginning, move right while looking for a 0,
2. If you find a 0, mark it, go to the beginning,
3. Move right while looking for a 1
4. If you find a 1, mark it, go to the beginning
5. Repeat 1-4 until no more 0s or no more 1s
6. Go back to beginning, return 1 if we can find more 1s
7. Else return 0

Turing Machine for Majority



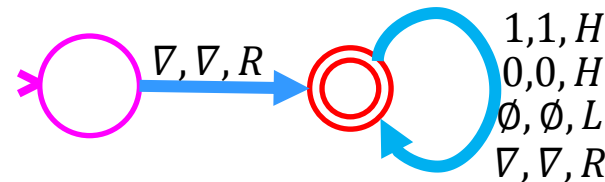
FSA vs TM

- Returns 1 when
 - FSA: you read the last input character and end up in a final state
 - TM: you take a halt transition to a final state
- Returns 0 when
 - FSA: you read the last input character and end up in a non-final state
 - TM: you take a halt transition to a non-final state
- FSA: you'll always eventually read the last character, so they always return something
- TM: You can build a machine that, for some inputs, never takes a halt transition!

Some Turing Machines never return

- In this case they run forever
- 3 behaviors
 - Return 1
 - Return 0
 - Run forever
- This is necessary for computation

```
while(true){  
    twiddle(thumbs);  
}
```



```
while(x != 1){  
    if(x%2 == 0){  
        x = x / 2;  
    }  
    else{  
        x = 3x+1;  
    }  
}
```

Running forever

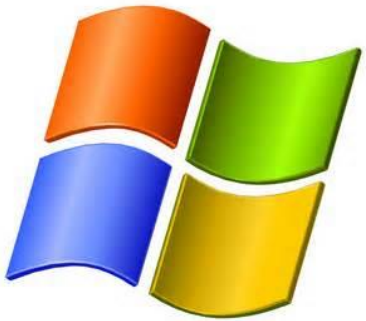
- Is it a bad thing?

Programs we want to halt:



Your 2150
Homework

Programs we want to run forever:



Church Turing Thesis

COMPUTING
DIVISION
COMPUTING
SECTION

Bonus Bureau, Computing Division, 11/24/1924



<https://www.loc.gov/pictures/item/201>

Computing is normally done by writing certain symbols on paper. We may suppose this paper is divided into squares like a child's arithmetic book. In elementary arithmetic the two-dimensional character of the paper is sometimes used. But such a use is always avoidable, and I think that it will be agreed that the two-dimensional character of paper is no essential of computation. I assume then that the computation is carried out on one-dimensional paper, *i.e.* on a tape divided into squares. I shall also

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

The “computable” numbers may be described briefly as the real numbers whose expressions as a decimal are calculable by finite means.

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

will be agreed that the two-dimensional character of paper is no essential of computation. I assume then that the computation is carried out on one-dimensional paper, *i.e.* on a tape divided into squares. I shall also suppose that the number of symbols which may be printed is finite. If we were to allow an infinity of symbols, then there would be symbols differing to an arbitrarily small extent[†]. The effect of this restriction of the number of symbols is not very serious. It is always possible to use sequences of symbols in the place of single symbols. Thus an Arabic numeral such as

17 or 9999999999999999 is normally treated as a single symbol. Similarly in any European language words are treated as single symbols (Chinese, however, attempts to have an enumerable infinity of symbols). The differences from our point of view between the single and compound symbols is that the compound symbols, if they are too lengthy, cannot be observed at one glance. This is in accordance with experience. We cannot tell at a glance whether 9999999999999999 and 9999999999999999 are the same.

will be agreed that the two-dimensional character of paper is no essential of computation. I assume then that the computation is carried out on one-dimensional paper, *i.e.* on a tape divided into squares. I shall also suppose that the number of symbols which may be printed is finite. If we were to allow an infinity of symbols, then there would be symbols differing to an arbitrarily small extent[†]. The effect of this restriction of the number of symbols is not very serious. It is always possible to use sequences of

[†] If we regard a symbol as literally printed on a square we may suppose that the square is $0 \leq x \leq 1, 0 \leq y \leq 1$. The symbol is defined as a set of points in this square, viz. the set occupied by printer's ink. If these sets are restricted to be measurable, we can define the "distance" between two symbols as the cost of transforming one symbol into the other if the cost of moving unit area of printer's ink unit distance is unity, and there is an infinite supply of ink at $x = 2, y = 0$. With this topology the symbols form a conditionally compact space.

a glance whether 9999999999999999 and 9999999999999999 are the same.

Why is the *number of states* finite?

The behaviour of the computer at any moment is determined by the symbols which he is observing, and his “state of mind” at that moment. We may suppose that there is a bound B to the number of symbols or squares which the computer can observe at one moment. If he wishes to observe more, he must use successive observations. We will also suppose that the number of states of mind which need be taken into account is finite. The reasons for this are of the same character as those which restrict the number of symbols. If we admitted an infinity of states of mind, some of them will be “arbitrarily close” and will be confused. Again, the restriction is not one which seriously affects computation, since the use of more complicated states of mind can be avoided by writing more symbols on the tape.