# Turing Machines and the Halting Problem

# Outline and Objectives

**Outline**

- Motivation
- Turing Machine
  - Definition
  - Examples
- Halting Problem

**Learning Objectives**

- Discuss the concept of a Turing Machine and tie to stored program concept and FSM
- Explain the Halting problem *and its result*

# ??

What are the limits of algorithm problem solving and solutions?

Do you think there are formulated problems that cannot be solved algorithmically?

What would a problem like this look like?

# Turing Machine

A Turing Machine is an abstract computational model that …

- operates on an infinite, bi-directional tape
- reads/writes a symbol to the tape
- keeps track of a state and jumps from one finite state to another based on the current state and input

# Turing Machine

Mathematically, a Turing Machine can be defined as a function f

```
f: Q x S -> S x {L,R} x Q
```

where

```
S = {0,1,…} // finite set of symbols
Q = {A,B,C,…H,} // finite set of states
L,R // direction to move tape
```

(Sipser, 140)

# Turing Machine

Or…

A TM reads a value from the tape… transitions to a new state based on the value read and current state… moves the position of the tape… and possibly writes to the tape.

# Turing Machine

Two of the states are of particular interest…

- Initial state (e.g., A) – where the machine starts
- Halt state (e.g., H) – upon entering this state the machine stops execution (i.e., halts)

Initially, the position of the tape is located at the leftmost character and blanks are interpreted as ‘0’.

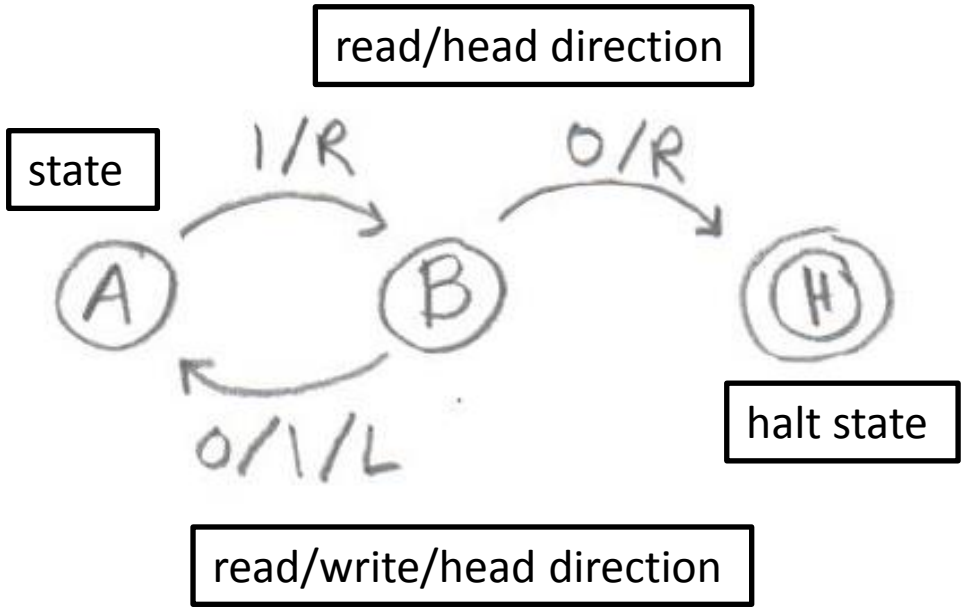
# Turing Machine

What differentiates one Turing Machine from another is primarily …

… the function $f$ (i.e., transition function)

The symbols and states play a role as well.

# TM Notation for Graphs



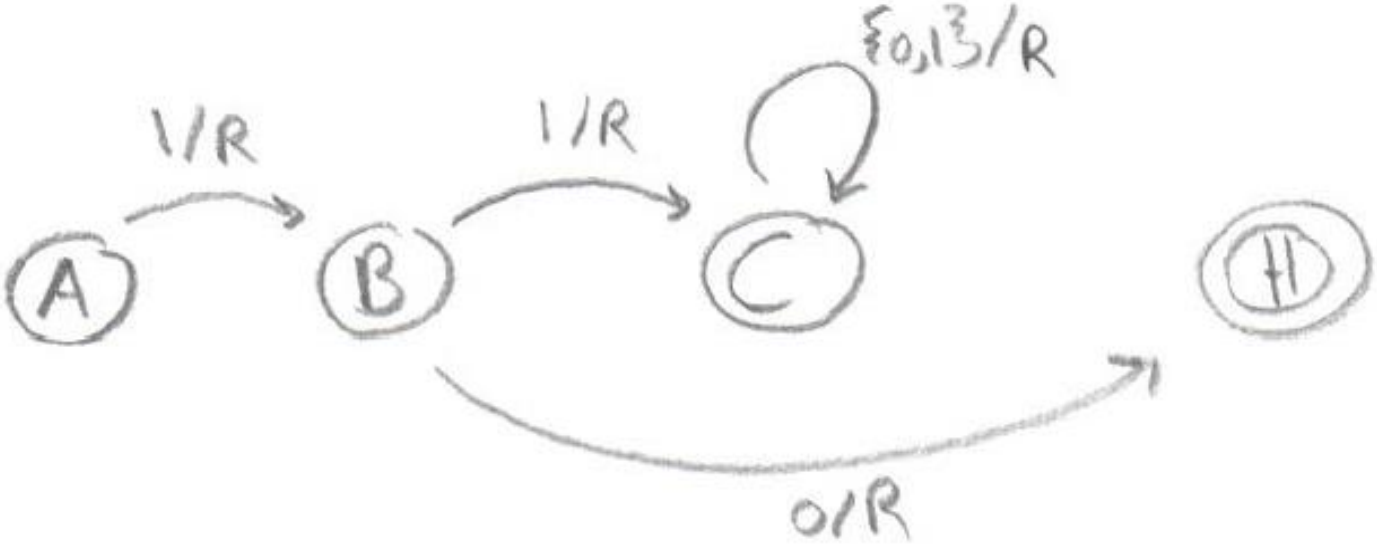
# TRUE/FALSE Example

Assume TRUE = “11” and FALSE = “1”.

We want to create a TM *W* such if the tape contains true when started, the machine does not stop. If the tape contains FALSE, then *W* will halt.

We need to specify S, Q, and f.

# TRUE/FALSE Example



# Specification for TM W

S={A,B,C,H}

Q={0,1}  // A blank is interpreted as 0

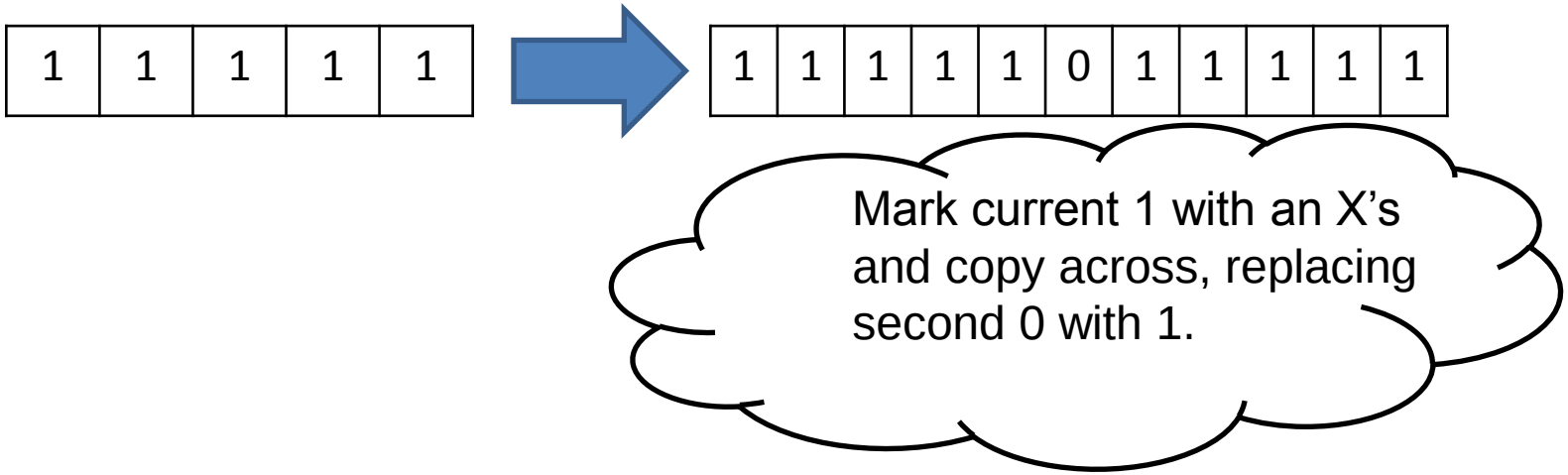
f:

| f(Q,S) | 0 | 1 |
|---|---|---|
| A (start) | --- | **1BR** |
| B | **0HR** | **1CR** |
| C | **0CR** | **1CR** |
| H (halt) | --- | --- |

# COPY Example

Activity: Create a TM to copy a string of 1's on the Tape.

Assume the input is a block of 1's. The TM should make a copy of the 1's separated by a 0 and then halt.



# Specification for TM C

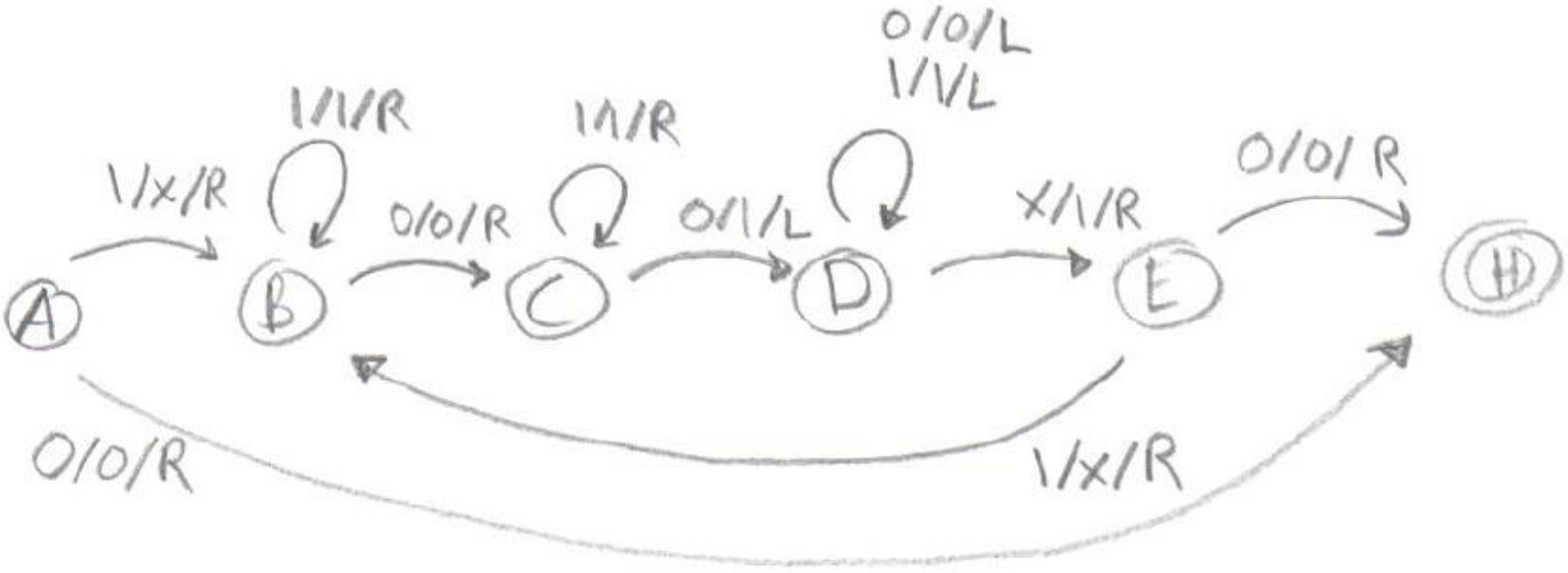
S={A,B,C,D,E,H}

Q={0,1,x}

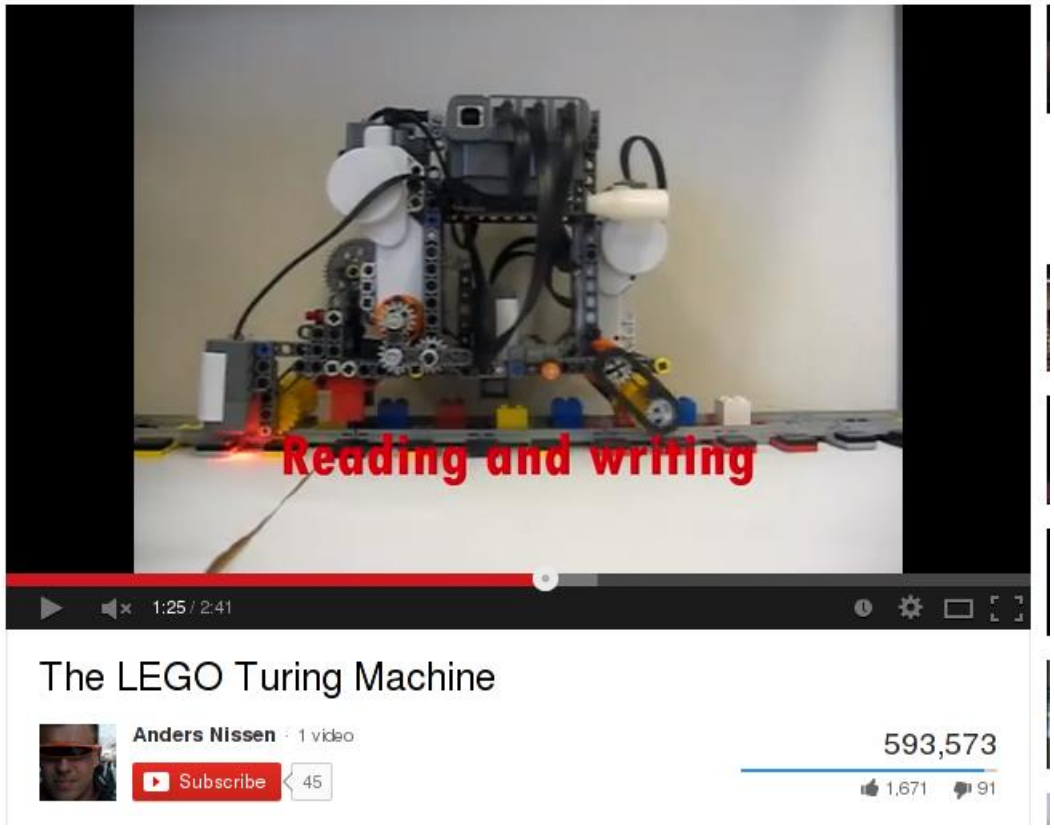
f:

| f(Q,S) | 0 | 1 | X |
|---|---|---|---|
| A (start) | **0HR** | **XBR** | **--** |
| B | **0CR** | **1BR** | **--** |
| C | **1DL** | **1CR** | **--** |
| D | **0DL** | **1DL** | **1ER** |
| E | **0HR** | **XBR** | **--** |
| H (halt) | -- | -- | **--** |

# COPY Example



# Turing Machine




youtube.com/watch?v=cYw2ewoO6c4

# Turing Machine

How to relate a Turing Machine to modern computing?

Tape -> Storage (RAM or Disk)

Function -> Program

Significant in that it was first described by Alan Turing in 1937…

# Universal Turing Machine (UTM)

The idea is to encode a Turing Machine (t) on the tape and follow it by the input (n) to the TM

Then the UTM would read the TM and execute it.

What key concept does this resemble?

STORED PROGRAM CONCEPT

(Sipser, 174)

# Computability

TMs are quite powerful…

computable numbers – a number is TM computable if a TM can calculate a number to arbitrary precision starting from a black tape

computable functions – a function that determines a TRUE/FALSE statement about computable arguments is a TM computable function.

# Halting Problem

It is of interest to know if a TM will halt for a particular input…

We would like a TM which will compute a function *h(t,n)* that which returns

- TRUE if TM *t* halts when started with *n*
- FALSE otherwise

# Halting Problem

THE HALTING PROBLEM IS NOT COMPUTABLE!!!

Proof by Contraction: Assume that we have such a TM H. Join H and C* (the copy TM) such that the halt state of C is the start state of H. This TM will determine if a machine whose encoding is *t*, halts when given *t* as an input.

# Halting Problem

To the halt state of H, add the TM W which goes to into an infinite set of transitions if its input is TRUE and halts if its input is FALSE.

Call the final TM M. M halts if the TM with encoding *t* does not halt on an tape which initially contains *t*. Otherwise it does not halt.

Consider running M on itself. This is a contradiction!

(Sipser, 174)

# Halting Problem

There are two possibilities…

i. M halts on its encoding t (ie, H(t,t) is TRUE), but then M(t) should not halt by definition.

ii. M does not halt on its encoding t (ie, H(t,t) is FALSE), but then M(t) should halt by definition.

Either way there is a contradiction. Our initial assumption about the existence of H is false.

# Consequences of the Halting Problem

Not everything is computable…

we can’t write a program which checks if it will stop

software verification

# References


- Sipser M: *Introduction to the theory of computation*. 2nd ed. Boston: Thomson Course Technology; 2006.