Oracle Turing machines

Pengxiang Xu

THE UNIVERSITY OF TENNESSEE KNOXVILLE

Outline

- Turing machine
  - Definition
  - Example

- Halting problem
  - Definition

- Oracle turing machine
  - What is Oracle turing machine
  - Application

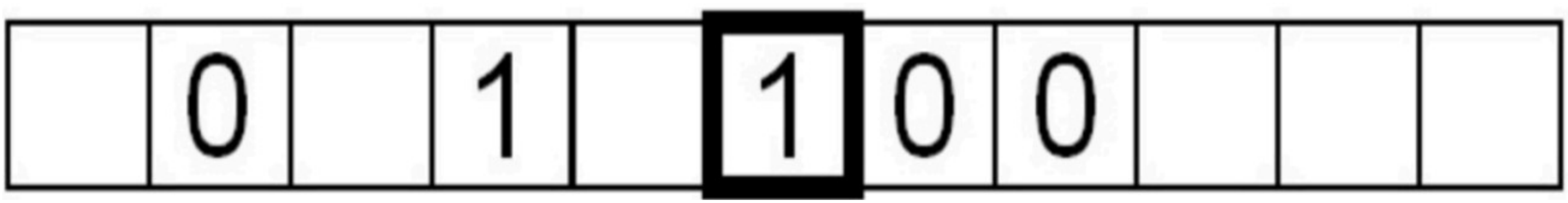
Turing machine

- In 1936, Alan Mathison Turing proposed a computer model that could simulate all computational behaviors—the Turing machine
- A Turing machine is a hypothetical machine.
- Component:
  - Infinite long tape
  - Head
  - Transition rule
  - State

Turing machine

Tape

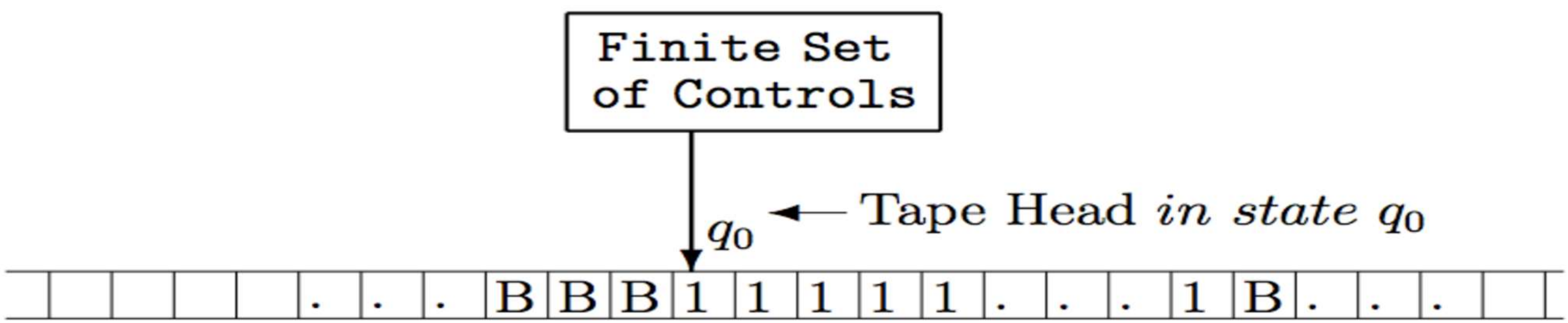
- An infinite tape marked out into squares, each square can be **symbol or Blank**(assume 1,0 and Blank)
- The tape can be moved **left and right** through the machine



Turing machine

Head

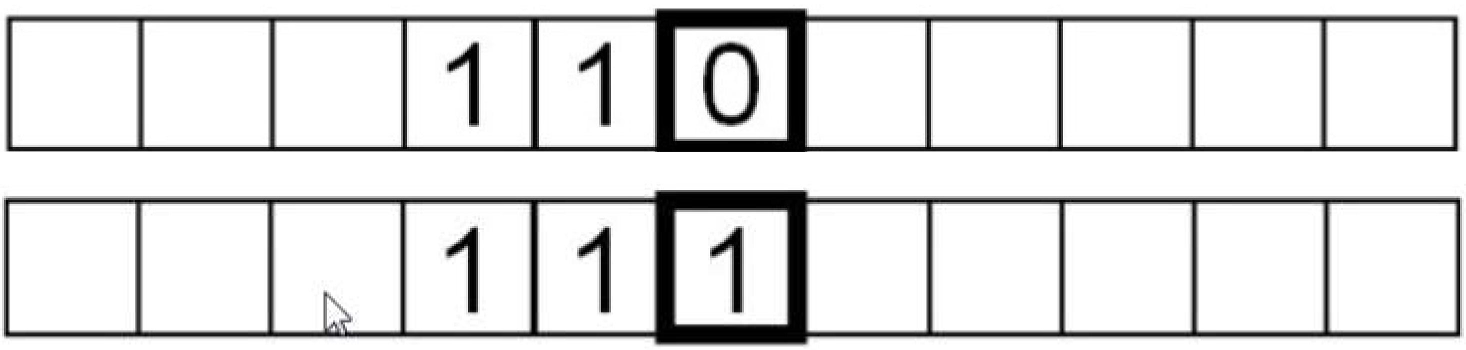
- **Read** the symbol on the square under the head
- **Edit** the symbol by writing a new symbol or erasing it
- **Move** the tape **left or right** by one square



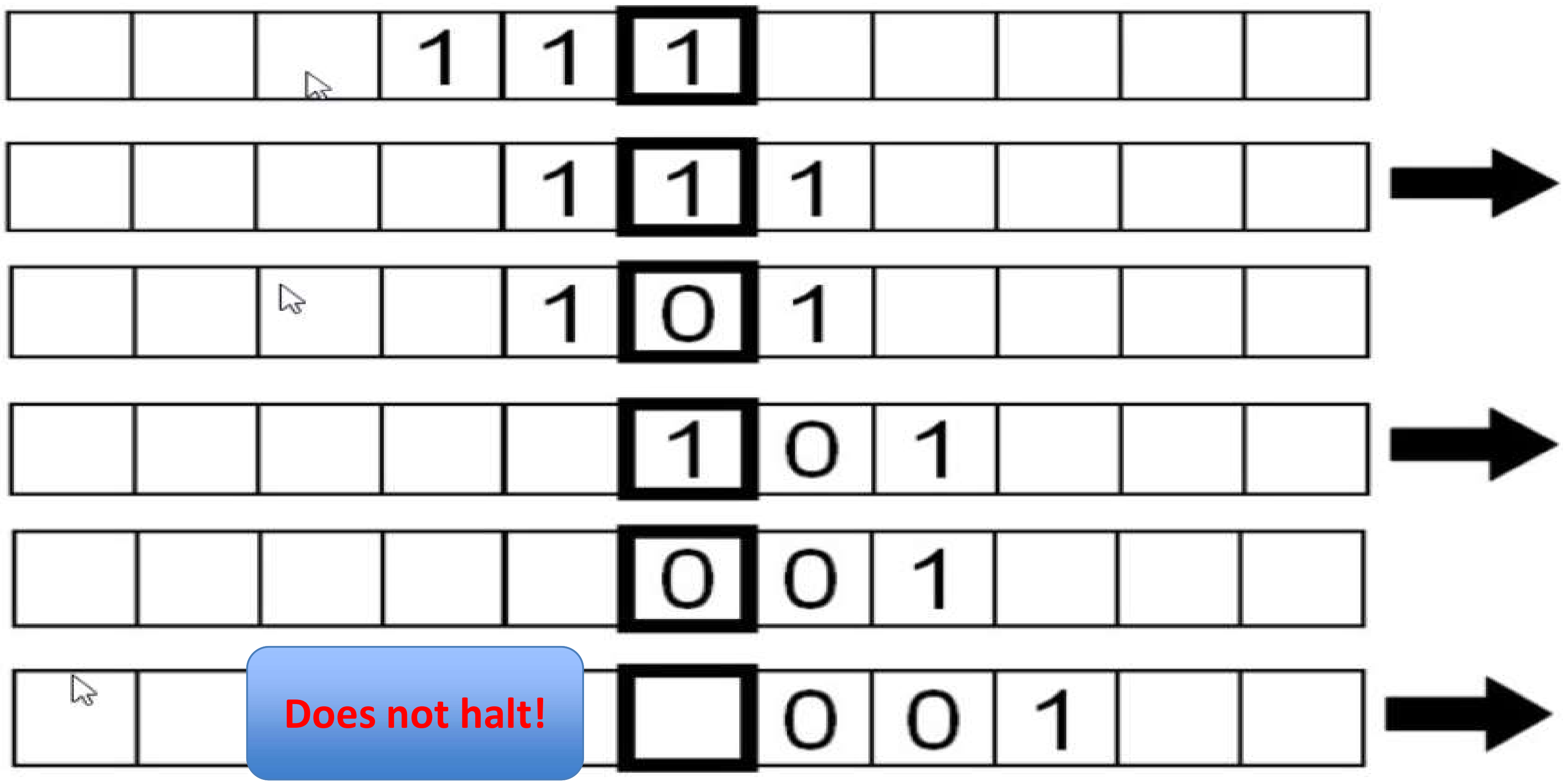
Turing machine

Reverse 0 to 1 and 1 to 0

Symbol read	Write	Move
Blank	None	None
0	1	Right
1	0	Right



Turing machine



Turing machine

State

State	Symbol read	Write	Move	Next State
State 0	Blank	None	None	Stop state
	0	1	Right	State 0
	1	0	Right	State 0

Turing machine

State

State	Symbol read	Write	Move	Next State
State 0	Blank	None	Left	State 1
	0	1	Right	State 0
	1	0	Right	State 0
State 1	Blank	None	Right	Stop state
	0	1	Left	State 1
	1	0	Left	State 1

Halting Problem

The **halting problem** is the problem of determining whether the program will finish running or continue to run forever.

For example,

while (true) continue

This does not halt

Halting Problem

We often want to know if a program halt, but **not possible** to provide one algorithm that answers this for all programs.

Halting Problem

Program H: I can determine all the programs whether they can halt.

Program U: I don't believe that! Given any program, if you think it can halt, I will loop forever; If you think it loop forever, I will halt.(do the opposite)

Program H: So what?

Program U: Could you determine if I can halt?

Halting Problem

```
int H(program)      //return 0 if halt, return 1 if can't halt
int U(program){
    if( H(program)==0 ){
        while(1);
    }else{
        return 0;
    }
}
```

What about U(U)?

Oracle Turing Machine

- It can be visualized as a Turing machine with **a black box**, called an oracle, which is able to decide certain decision problems in a single operation.
- The problem can be of any complexity class. Even **undecidable problems**, like the halting problem, can be used.
- Unrealistic Machine

Oracle Turing Machine

- A machine with an oracle for the halting problem can determine whether particular Turing machines will halt on particular inputs, but they cannot determine whether machines equivalent to themselves will halt.
- This creates a hierarchy of machines, each with a more powerful halting oracle and an even harder halting problem

Oracle Turing Machine

An oracle machine must include a typical Turing machine. AND extra parts:

- An Oracle Tape.
- An Oracle head.
- Two special state: QUERY and RESPONSE

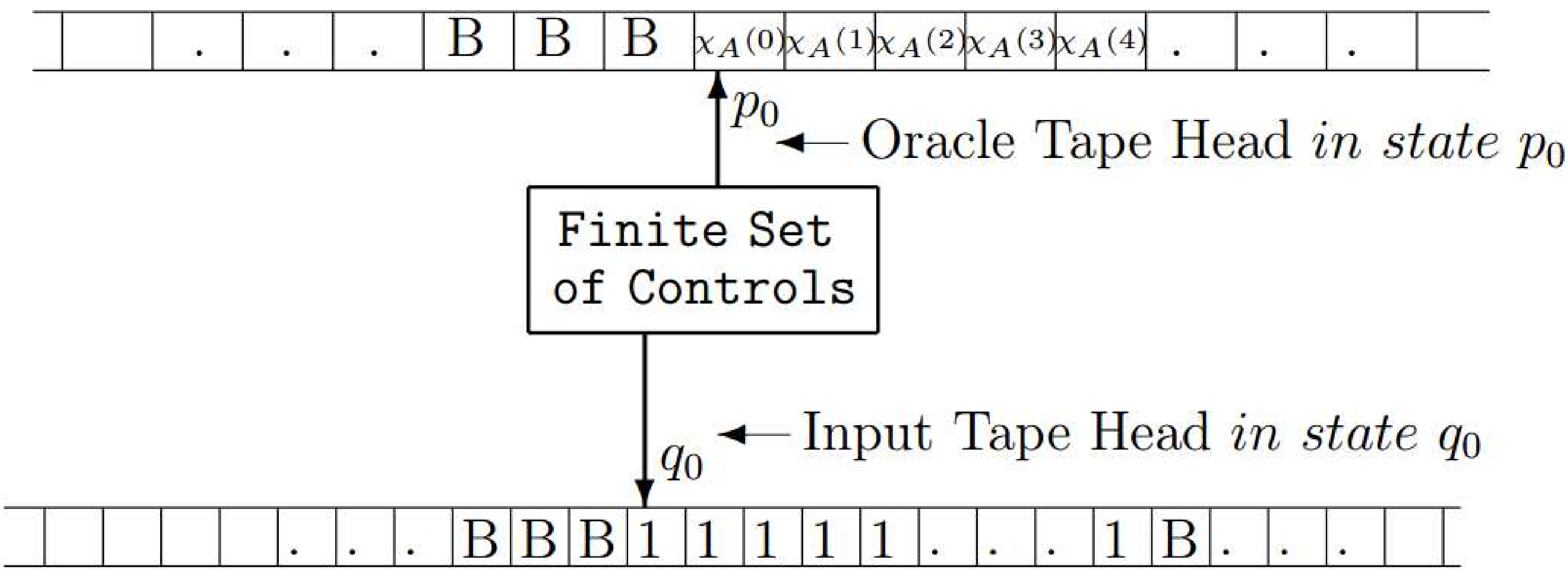
Oracle Turing Machine

From time to time, the oracle machine may enter the QUERY state. When this happens, the following actions are performed in a single computational step:

- Read the contents of the oracle tape
- Viewed oracle and replaced the contents of the oracle tape with the solution to the problem instance
- The state of the oracle machine is changed to RESPONSE either of Qyes or Qno

The effect of changing to the QUERY state is thus to receive, in a single step, a solution to the problem instance that is written on the oracle tape

Oracle Turing Machine



Application

- Turing reductions.
- Identify some barriers to proving results in complexity theory, like proving that P≠NP.
- Cryptography

Turing Reduction

- In computability theory, a Turing reduction from problem A to problem B is a reduction which solves A, assuming the solution to B is already known

Thank you!

Questions?