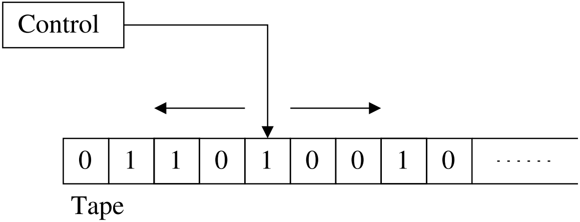
Turing Machine 01

Thumrongsak Kosiyatrakul
tkosiyat@cs.pitt.edu

Human Computer

- Imagine a human computer working with a pencil and paper:
  - The only things written on the paper are symbols from some fixed finite alphabet
  - Each step taken by the computer depends only on the symbol he/she is currently examining and on his "state of mind" at the time
  - His state of mind may change as a result of his examining different symbols, only a finite number of distinct states of mind are possible
- This sounds like a finite-state machine that has its input string written on the paper
- Primitive Steps of Human Computation:
  - Examining an individual symbol on the paper
  - Erasing a symbol or replacing it by another
  - Transferring attention from one symbol to another nearby symbol

- You can think of a **Turing machine** a finite state machine with unlimited amount of memory



- A Turing machine has the following:
  - A control (state diagram/transition functions)
  - An infinite long tape
  - A tape head that can move around on the tape
    - A TM can read input symbols from its tape
    - A TM can write output symbols to its tape

Turing Machines

- At first, tape contains the input string follows by infinite blank (␣) symbols
  - Example: Input: 101

1	0	1	␣	␣	␣	␣	␣	␣	␣	␣	······

  - Example: Input: 011010010

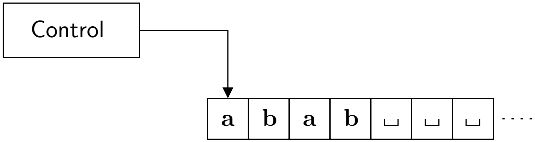
0	1	1	0	1	0	0	1	0	␣	␣	······

  - Example: Input: ε

␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	······

- The machine can store information by writing symbols onto the tape
- The output of a machine is either *accept* or *reject*
  - Output symbols/strings can also be on the tape
- **A Turing machine may run forever**

Turing Machine



1. A Turing machine can read a symbol from the tape under the tape head
2. A Turing machine can write a symbol to the tape under the tape head
3. The tape head can move to the left and to the right one square at a time
4. The tape is infinite
5. The special states for rejecting and accepting take effect immediately (q_{reject} and q_{accept})
    - Unlike DFA that needs to process the last input symbol before accepting or rejecting an input string

Example

- Let $\Sigma = \{0, 1, \#\}$
- Consider the language $B = \{w\#w \mid w \in \{0, 1\}^*\}$
  - Example of strings in B are
    - $01101\#01101$
    - $\#$
    - $01\#01$
  - This language is not regular
    - Need an infinite states to to remember all symbols in w
    - We can use the Pumping lemma to prove that it is not regular
  - If the number of # symbol in a string is not exactly 1, the string is not in the language B

Example

- Let $\Sigma = \{0, 1, \#\}$ and $B = \{w\#w \mid w \in \{0, 1\}^*\}$
- Imaging what would you do if I give you a very long piece of paper and it contains a string of the form $x\#y$ where $x, y \in \{0, 1\}^*$ and they are so long that you cannot remember all its symbols
- One way is to go back and forth across the # and compare symbols at the same position one symbol at a time
- You may need to cross off those that have been compared

$0110110101011010111010100101011001100101010\#0110110101011010111010100101011001100101010$

Formal Definition of a Turing Machine

A **Turing machine** is a 7-tuple, $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$, where Q, Σ, Γ are all finite sets and

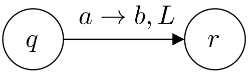
1. Q is a set of states,
2. Σ is the input alphabet not containing the **blank symbol** $\sqcup$,
3. Γ is the tape alphabet, where $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$,
4. $\delta : Q \times \Gamma \to Q \times \Gamma \times \{\mathsf{L}, \mathsf{R}\}$ is the transition function,
5. $q_0 \in Q$ is the start state,
6. $q_{\text{accept}} \in Q$ is the accept state, and
7. $q_{\text{reject}} \in Q$ is the reject state.

Transition Function of a Turing Machine

- The transition function of a TM is defined as:

$$\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{\mathsf{L}, \mathsf{R}\}$$

- An assignment $\delta(q, a) = (r, b, L)$ means if the machine is at state q and the tape head is over a square containing a, the machine write symbol b (replacing a), change its state to r and move the tape head to the left one square
  - This transition can be represented in a state diagram as shown below:



Computation of a Turing Machine

- The input string $w = w_1 w_2 \ldots w_n \in \Sigma^*$ is on the leftmost n square of the tape and the rest are filled with blank symbols
- The tape head starts at the leftmost square of the tape
- The machine processes input according to its transition function
- If the tape head is at the leftmost square and the transition function indicates L, the tape head stays at the same place
- The machine continues until it enter the q_{accept} or q_{reject} state
- **The machine may run forever**

Tracing a Turing Machine

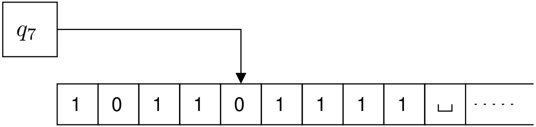
- To trace whether a DFA accept or reject a string is easy
- To trace whether an NFA accept or reject a string is harder
  - An NFA can split into multiple copies
  - Each copy has its own current state
  - We use a computational tree to keep track of all copies
- To trace whether a TM accept or reject a string is even harder
  - We need to keep track or the current state (similar to DFA or NFA)
  - We also need to keep track of the content of the tape
    - The content of the tape is changed over time (TM can write onto the tape)
  - We also need to keep track of the location of its tape head
    - Need to know the symbol under the tape head
    - The tape head needs to move to either left or right direction at every step

Configuration

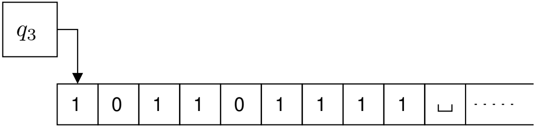
- When a machine process an alphabet, three things change:
  - the state of the machine,
  - the content of the tape, and
  - the location of the tape head
- The above three items can be represented by a configuration
- A **configuration** is in the form of $u\ q\ v$
  - u and v are strings that can be empty
  - q is a state that represents the current state of TM
  - The content of the tape is the string uv
  - The tape head is on the first alphabet of the string v

Configuration (Examples)

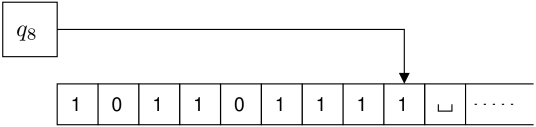
- Example: The configuration $1011q_7 01111$ corresponds to a machine as shown below:



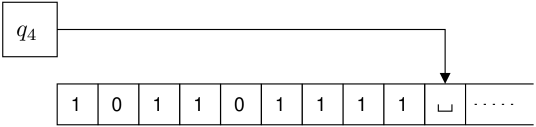
- Example: The configuration $q_3 101101111$ corresponds to a machine as shown below:



- Example: The configuration $10110111q_8 1$ corresponds to a machine as shown below:



- Example: The configuration $101101111q_4$ corresponds to a machine as shown below:



Configuration

- We say that the configuration C_1 **yields** configuration C_2 if the machine can **legally** go from C_1 to C_2 in one step
- For example, suppose a TM has the following transition function:

$$\delta(q_i, 0) = (q_j, 1, L)$$

- Consider the configuration $010010q_i0101$
  - For the above configuration, $u = 010010$, $q = q_i$, and $v = 0101$
    - The current state is q_i
    - The content of the tape is 0100100101
    - The tape head is on top of the symbol 0 (the first symbol of v)
  - According to the above transition function, the next configuration would be

$$01001q_j01101$$

- We says that the $010010q_i0101$ yields $01001q_j01101$

Formally:

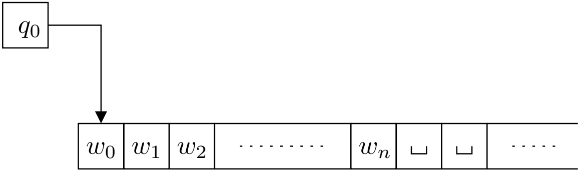
- Let $a, b, c \in \Gamma$ and $u, v \in \Gamma^*$
  - a, b, and c are symbols
  - u and v are strings over Γ
  - By concatenating the symbol a to the end of the string u, we get the string ua
- We say that

$$ua\ q_i\ bv \quad \text{yields} \quad u\ q_j\ acv$$

if $\delta(q_i, b) = (q_j, c, L)$

Starting Configuration

- Given a string $w = w_0 w_1 w_2 \dots w_n$
- Suppose the start state of a TM M is q_0
- When M is about to process the string w:



- The starting configuration of M on input w is $q_0 w$
- Example: The starting configuration of M on input 01101 will be $q_0 01101$

Configuration

- If the state in a configuration is q_{accept}, the configuration is called **accepting configuration**
  - $0101q_{\text{accept}}101$
  - $q_{\text{accept}}1111$
- Similarly, if the state in a configuration is q_{reject}, it is called **rejecting configuration**
  - $q_{\text{reject}}010100$
  - $011q_{\text{reject}}01$
- Once a machine yields either the accepting or rejecting configuration, the machine will not yield any other configuration (**halting configuration**).

Language of a TM

- A machine M accepts a string w if the sequence of configuration $C_1, C_2, \ldots C_k$ exists, where
  1. C_1 is the starting configuration of M on input w,
  2. each C_i yields C_j, and
  3. C_k is an accepting configuration
- The set of all strings A accepted by M is **the language of M**
  - M recognizes A or
  - $L(M) = A$
- **Note**: If $L(M) = A$, it does not mean that M rejects all strings that are not in the language A
  - Given a string $s \notin A$, M may loop indefinitely on input s
  - Unlike a DFA D, if $L(D) = B$, D rejects all strings not in the language D
    - DFA D cannot loop indefinitely on any string

Language of a TM

- A language B is called **Turing-recognizable** if some Turing machine recognize it
  - Sometimes we call B is recognizable
- Given a string w and a TM M there are three possibilities:
  1. M accepts w
  2. M rejects w
  3. M loops indefinitely on input w
- Turing machines that never loop indefinitely are called **deciders**
  - These type of TMs will always halt on all inputs
- A decider that recognizes a language is said to **decide** that language
- A language is called **Turing-decidable** if some Turing machines decide it
- Suppose TM M is a decider and $L(M) = C$, we say that
  - M decides C
  - C is decidable