

Lecture 21 – Turing Machines (TMs)

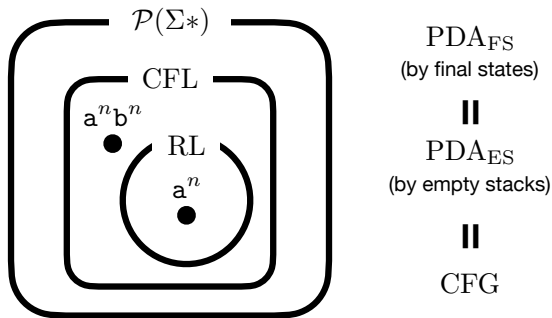
COSE215: Theory of Computation

Jihyeok Park



2026 Spring

- A **context free language** can be recognized by a **context free grammar (CFG)** or a **pushdown automaton (PDA)**.



- Can we increase the expressive power of CFGs or PDAs?

1. Chomsky Hierarchy

2. Turing Machines

- Definition

- Turing Machines in Scala

- Configurations

- One-Step Moves

- Halting of Turing Machines

- Language of Turing Machines

- Turing Machines as Computing Machines

1. Chomsky Hierarchy

2. Turing Machines

Definition

Turing Machines in Scala

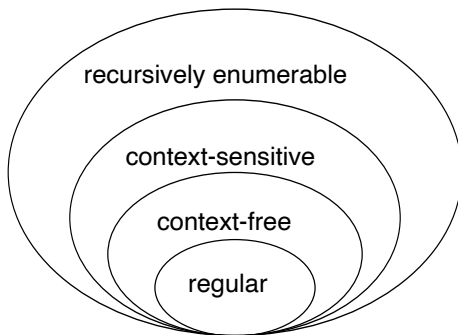
Configurations

One-Step Moves

Halting of Turing Machines

Language of Turing Machines

Turing Machines as Computing Machines



Type	Language	Grammar	Automaton
3	Regular (RL)	Regular	Finite Automaton (FA)
2	Context-Free (CFL)	Context-Free	Pushdown Automaton (PDA)
1	Context-Sensitive (CSL)	Context-Sensitive	Linear-Bounded Automaton (LBA)
0	Recursively Enumerable (REL)	Unrestricted	Turing Machine (TM)

A **Type-3** language is called a **regular language (RL)**.

It can be recognized by a **finite automaton (FA)** or a **regular grammar (RG)** containing production rules of the form:

$$A \rightarrow aB \quad \text{or} \quad A \rightarrow a \quad \text{or} \quad A \rightarrow \epsilon$$

where $A, B \in V$ and $a \in \Sigma$.

For example, the following language is a RL:

$$L = \{a^n \mid n \geq 0\}$$

It can be recognized by the following RG:

$$S \rightarrow aS \mid \epsilon$$

A **Type-2** language is called a **context-free language (CFL)**.

It can be recognized by a **pushdown automaton (PDA)** or a **context-free grammar (CFG)** containing production rules of the form:

$$A \rightarrow \alpha$$

where $A \in V$ and $\alpha \in (V \cup \Sigma)^*$.

For example, the following language is a CFL:

$$L = \{a^n b^n \mid n \geq 0\}$$

It can be recognized by the following CFG:

$$S \rightarrow aSb \mid \epsilon$$

A **Type-1** language is called a **context-sensitive language (CSL)**.

It can be recognized by a **linear-bounded automaton (LBA)** or a **context-sensitive grammar** containing production rules of the form:

$$\alpha A \beta \rightarrow \alpha \gamma \beta \quad \text{or} \quad S \rightarrow \epsilon$$

where $A \in V$, $\alpha, \beta, \gamma \in (V \cup \Sigma)^*$, $|\gamma| \geq 1$, and S is the start variable.

For example, the following language is a CSL:

$$L = \{a^n b^n c^n \mid n \geq 0\}$$

It can be recognized by the following CSG:

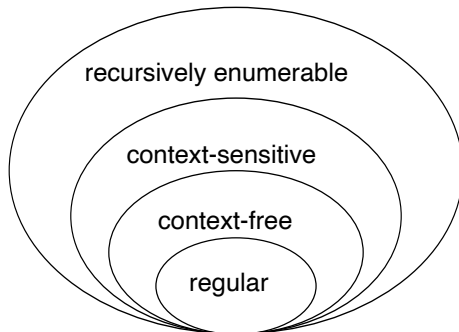
$S \rightarrow aBC$	$CB \rightarrow CZ$	$aB \rightarrow ab$
$S \rightarrow aSBC$	$CZ \rightarrow WZ$	$bB \rightarrow bb$
	$WZ \rightarrow WC$	$bC \rightarrow bc$
	$WC \rightarrow BC$	$cC \rightarrow cc$

A **Type-0** language is called a **recursively enumerable language (REL)**.

It can be recognized by a **Turing machine (TM)** or an **unrestricted grammar (UG)** containing production rules of the form:

$$\alpha \rightarrow \beta$$

where $\alpha, \beta \in (V \cup \Sigma)^*$ and $|\alpha| \geq 1$.



Type	Language	Grammar	Automaton
3	Regular (RL)	Regular	Finite Automaton (FA)
2	Context-Free (CFL)	Context-Free	Pushdown Automaton (PDA)
1	Context-Sensitive (CSL)	Context-Sensitive	Linear-Bounded Automaton (LBA)
0	Recursively Enumerable (REL)	Unrestricted	Turing Machine (TM)

We will not cover details of **Type-1** languages in this course.

Let's focus on **Type-0** languages and **Turing Machines (TMs)**.

1. Chomsky Hierarchy

2. Turing Machines

- Definition

- Turing Machines in Scala

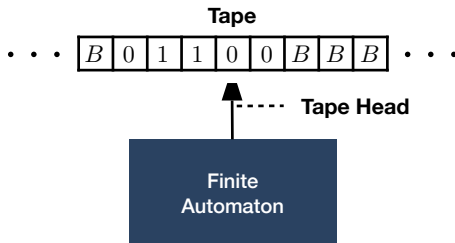
- Configurations

- One-Step Moves

- Halting of Turing Machines

- Language of Turing Machines

- Turing Machines as Computing Machines



A **Turing machine (TM)** is a **deterministic** FA with a **tape**.

- A **tape** is an infinite sequence of cells containing **tape symbols**. (The **blank symbol** B is a special symbol representing an empty cell.)
- A **tape head** points to the current cell.
- A **transition** performs the following operations depending on the current 1) **state** and 2) **tape symbol** pointed by the tape head:
 - **Change** the current **state**.
 - **Replace** the current **tape symbol** pointed by the tape head.
 - **Move** the **tape head** left or right.

Definition (Turing Machines)

A **Turing machine (TM)** is a 7-tuple

$$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$$

where

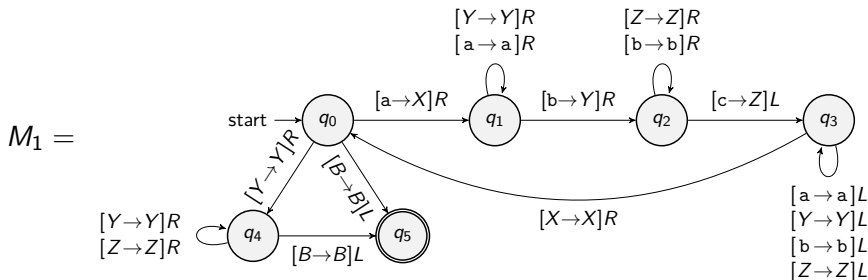
- Q is a finite set of **states**.
- Σ is a finite set of **input symbols**.
- Γ is a finite set of **tape symbols** containing input symbols ($\Sigma \subseteq \Gamma$).
- $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ is a **transition function**.
- $q_0 \in Q$ is the **initial state**.
- $B \in \Gamma \setminus \Sigma$ is the **blank symbol**.
- $F \subseteq Q$ is the set of **final states**.

Note that \rightarrow denotes a **partial function** (i.e., a function that may not be defined for some inputs).

$$M_1 = (\{q_0, q_1, q_2, q_3, q_4, q_5\}, \{a, b, c\}, \{a, b, c, X, Y, Z, B\}, \delta, q_0, B, \{q_5\})$$

$\delta(q_0, a) = (q_1, X, R)$	$\delta(q_0, Y) = (q_4, Y, R)$	$\delta(q_0, B) = (q_5, B, L)$
$\delta(q_1, a) = (q_1, a, R)$	$\delta(q_1, Y) = (q_1, Y, R)$	$\delta(q_1, b) = (q_2, Y, R)$
$\delta(q_2, b) = (q_2, b, R)$	$\delta(q_2, Z) = (q_2, Z, R)$	$\delta(q_2, c) = (q_3, Z, L)$
$\delta(q_3, a) = (q_3, a, L)$	$\delta(q_3, Y) = (q_3, Y, L)$	$\delta(q_3, b) = (q_3, b, L)$
$\delta(q_3, Z) = (q_3, Z, L)$	$\delta(q_3, X) = (q_0, X, R)$	$\delta(q_4, Y) = (q_4, Y, R)$
$\delta(q_4, Z) = (q_4, Z, R)$	$\delta(q_4, B) = (q_5, B, L)$	

The **transition diagram** of M_1 is as follows:

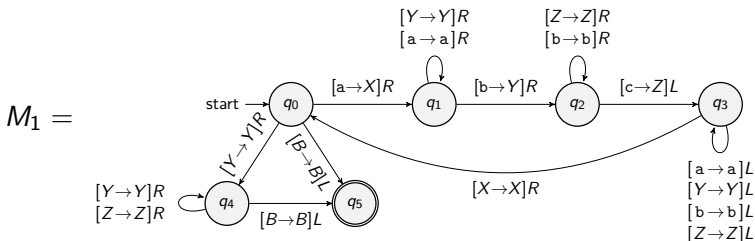


$$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$$

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

```
type State = Int
type Symbol = Char
type TapeSymbol = Char
enum HeadMove { case L, R }
import HeadMove.*

// The definition of Turing machines
case class TM(
  states: Set[State],
  symbols: Set[Symbol],
  tapeSymbols: Set[TapeSymbol],
  trans: Map[(State, TapeSymbol), (State, TapeSymbol, HeadMove)],
  initState: State,
  blank: TapeSymbol,
  finalStates: Set[State],
)
```



```

val tm1: TM = TM(
  states = Set(0, 1, 2, 3, 4, 5), symbols = Set('a', 'b', 'c'),
  tapeSymbols = Set('a', 'b', 'c', 'X', 'Y', 'Z', 'B'),
  trans = Map(
    (0, 'a') -> (1, 'X', R), (0, 'Y') -> (4, 'Y', R), (0, 'B') -> (5, 'B', L),
    (1, 'a') -> (1, 'a', R), (1, 'Y') -> (1, 'Y', R), (1, 'b') -> (2, 'Y', R),
    (2, 'b') -> (2, 'b', R), (2, 'Z') -> (2, 'Z', R), (2, 'c') -> (3, 'Z', L),
    (3, 'a') -> (3, 'a', L), (3, 'b') -> (3, 'b', L), (3, 'Y') -> (3, 'Y', L),
    (3, 'Z') -> (3, 'Z', L), (3, 'X') -> (0, 'X', R), (4, 'Y') -> (4, 'Y', R),
    (4, 'Z') -> (4, 'Z', R), (4, 'B') -> (5, 'B', L),
  ),
  initState = 0, blank = 'B', finalStates = Set(5),
)
    
```

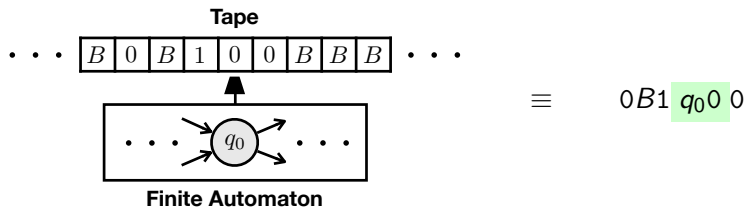
Definition (Configurations of Turing Machines)

A **configuration** of a Turing machine M is in the form of

$$X_1 \cdots X_{i-1} q X_i X_{i+1} \cdots X_n$$

where

- $q \in Q$ is the **current state**.
- $X_1 \cdots X_n \in \Gamma^*$ is the **sub-tape** between the left- and the right-most 1) non-blank symbols or 2) the symbol under the tape head.
- $X_i \in \Gamma$ is the **current tape symbol** under the tape head.



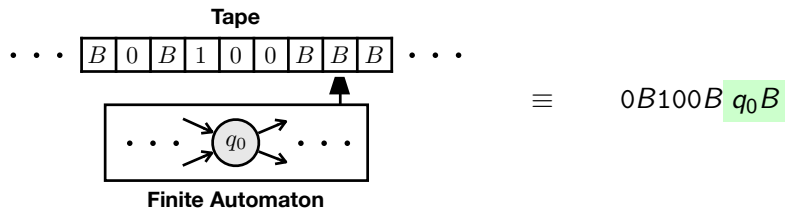
Definition (Configurations of Turing Machines)

A **configuration** of a Turing machine M is in the form of

$$X_1 \cdots X_{i-1} q X_i X_{i+1} \cdots X_n$$

where

- $q \in Q$ is the **current state**.
- $X_1 \cdots X_n \in \Gamma^*$ is the **sub-tape** between the left- and the right-most 1) non-blank symbols or 2) the symbol under the tape head.
- $X_i \in \Gamma$ is the **current tape symbol** under the tape head.



Definition (One-Step Moves of Turing Machines)

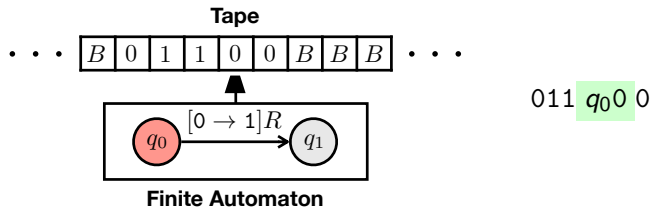
A **one-step move** (\vdash) of a Turing machine M is a transition from a configuration to another configuration.

- If $\delta(q, X_i) = (p, Y, L)$,

$$X_1 \cdots X_{i-1} qX_i X_{i+1} \cdots X_n \vdash X_1 \cdots pX_{i-1} YX_{i+1} \cdots X_n$$

- If $\delta(q, X_i) = (p, Y, R)$,

$$X_1 \cdots X_{i-1} qX_i X_{i+1} \cdots X_n \vdash X_1 \cdots X_{i-1} Y pX_{i+1} \cdots X_n$$



Definition (One-Step Moves of Turing Machines)

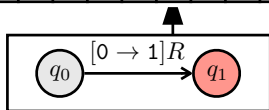
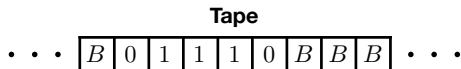
A **one-step move** (\vdash) of a Turing machine M is a transition from a configuration to another configuration.

- If $\delta(q, X_i) = (p, Y, L)$,

$$X_1 \cdots X_{i-1} qX_i X_{i+1} \cdots X_n \vdash X_1 \cdots pX_{i-1} YX_{i+1} \cdots X_n$$

- If $\delta(q, X_i) = (p, Y, R)$,

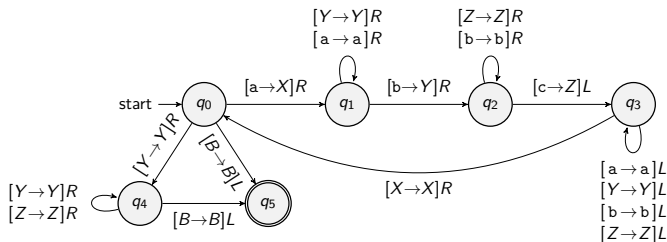
$$X_1 \cdots X_{i-1} qX_i X_{i+1} \cdots X_n \vdash X_1 \cdots X_{i-1} Y pX_{i+1} \cdots X_n$$



Finite Automaton

$$011 q_0 0 \vdash 0111 q_1 0$$

$M_1 =$



$q_0 a b c$	$\vdash X q_1 b c$	$(\because \delta(q_0, a) = (q_1, X, R))$
	$\vdash XY q_2 c$	$(\because \delta(q_1, b) = (q_2, Y, R))$
	$\vdash X q_3 Y Z$	$(\because \delta(q_2, c) = (q_3, Z, L))$
	$\vdash q_3 X Y Z$	$(\because \delta(q_3, Y) = (q_3, Y, L))$
	$\vdash X q_0 Y Z$	$(\because \delta(q_3, X) = (q_0, X, R))$
	$\vdash XY q_4 Z$	$(\because \delta(q_0, Y) = (q_4, Y, R))$
	$\vdash XYZ q_4 B$	$(\because \delta(q_4, Z) = (q_4, Z, R))$
	$\vdash XY q_5 Z$	$(\because \delta(q_4, B) = (q_5, B, L))$
	$\not\vdash$	

Definition (Halting of Turing Machines)

A Turing machine $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ **halts** on input w if there is a sequence of one-step moves from the **initial configuration** $q_0 w$ to a configuration having no more possible moves:

$$q_0 w \vdash^* \alpha q \beta \nmid$$

for some $\alpha, \beta \in \Gamma^*$ and $q \in Q$.

For example, the Turing machine M_1 halts on input abc:

$$q_0 a \text{ bc} \vdash^* XY q_5 Z \nmid$$

Definition (Acceptance by Turing Machines)

For a given Turing machine $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$, M accepts a word $w \in \Sigma^*$ if M **halts** on w with a **final state**:

$$q_0 w \vdash^* \alpha q_f \beta \nmid$$

for some $q_f \in F$ and $\alpha, \beta \in \Gamma^*$.

Definition (Language of Turing Machines)

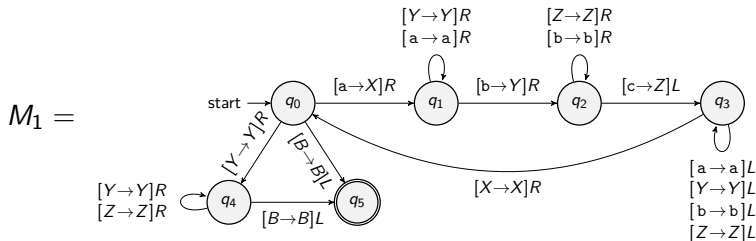
For a given Turing machine $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$, the **language** of M is defined as follows:

$$L(M) = \{w \in \Sigma^* \mid M \text{ accepts } w\}$$

Definition (Recursively Enumerable Languages (RELs))

A language L is **recursively enumerable** if there exists a Turing machine M such that $L = L(M)$.

For example, what is the language of the Turing machine M_1 ?



It accepts the following language. Thus, L is **recursively enumerable**:

$$L(M_1) = L = \{a^n b^n c^n \mid n \geq 0\}$$

```
type Tape = String
case class Config(state: State, tape: Tape, index: Int)

case class TM(...):
  // A one-step move in a Turing machine
  def move(config: Config): Option[Config] = ...

  // The initial configuration of a Turing machine
  def init(word: Word): Config = word match
    case a <| x => Config(initState, word, 0)
    case _      => Config(initState, blank.toString, 0)

  // The configuration at which the TM halts
  final def haltsAt(config: Config): Config = move(config) match
    case None      => config
    case Some(next) => haltsAt(next)

  // The acceptance of a word by TM
  def accept(w: Word): Boolean = finalStates.contains(haltsAt(init(w)).state)

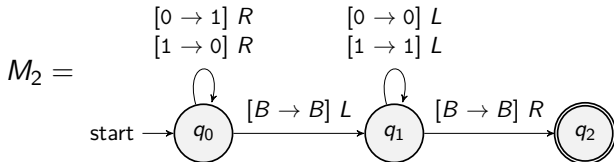
tm1.accept("abc")      // true
tm1.accept("aabbcc")  // true
tm1.accept("abab")    // false
```

Definition (Turing Computable Functions)

A partial function $f : \Sigma^* \rightarrow \Sigma^*$ is **Turing-computable** if there exists a Turing machine $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ such that

$$q_0 w \vdash^* q_f f(w) \not\vdash$$

for some $q_f \in F$ and all $w \in \Sigma^*$, such that $f(w)$ is defined.



For example, TM M_2 defines the following function $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$:

$$f(w) = (\text{the flip of each bit in } w)$$

For example, 0110 is transformed to 1001 by M_2 :

$$q_0 \ 0110 \vdash^* q_2 \ 1001 \not\vdash$$

and 1011100 is transformed to 0100011 by M_2 ¹:

$$q_0 \ 1011100 \vdash^* q_2 \ 0100011 \not\vdash$$

So, f is a **Turing-computable** function.

¹<https://plrg.korea.ac.kr/courses/cose215/materials/tm-flip.pdf>

```
case class TM(...):
  // The computation with a given word by TM
  def compute(word: Word): Option[Word] =
    val Config(state, tape, k) = haltsAt(init(word))
    val (n, x) = (tape.size, tape(k))
    if (k == 0 && finalStates.contains(state)) {
      if (x == blank && n == 1) Some("")
      else if (tape.forall(symbols.contains)) Some(tape.mkString)
      else None
    } else None

val tm2: TM = TM(
  states = Set(0, 1, 2), symbols = Set('0', '1'),
  tapeSymbols = Set('0', '1', 'B'),
  trans = Map(
    (0, '0') -> (0, '1', R), (0, '1') -> (0, '0', R), (0, 'B') -> (1, 'B', L),
    (1, '0') -> (1, '0', L), (1, '1') -> (1, '1', L), (1, 'B') -> (2, 'B', R),
  ),
  initState = 0, blank = 'B', finalStates = Set(2),
)
tm2.compute("0110") // Some("1001")
tm2.compute("1011100") // Some("0100011")
```

1. Chomsky Hierarchy

2. Turing Machines

- Definition

- Turing Machines in Scala

- Configurations

- One-Step Moves

- Halting of Turing Machines

- Language of Turing Machines

- Turing Machines as Computing Machines

- Examples of Turing Machines

Jihyeok Park
jihyeok_park@korea.ac.kr
<https://plrg.korea.ac.kr>