Lecture 21 – Turing Machines (TMs)

COSE215: Theory of Computation

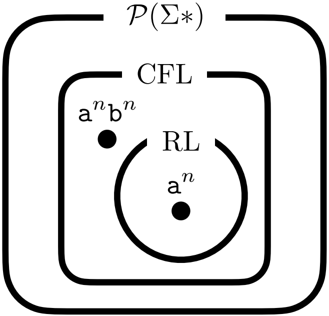
Jihyeok Park

PLRG

2023 Spring

Recall

- A **pushdown automaton (PDA)** is an extension of FA with a **stack**.



$\mathrm{PDA}_{\mathrm{FS}}$
(by final states)

$\|$

$\mathrm{PDA}_{\mathrm{ES}}$
(by empty stacks)

$\|$

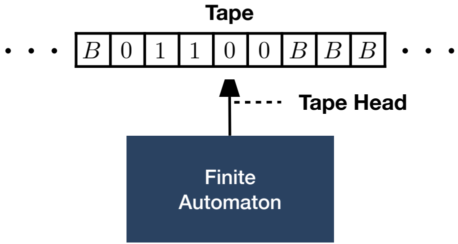
CFG

- Then, how about extensions of finite automata with other structures?
- Do they still represent the class of **context-free languages (CFLs)**?

Contents

1. Turing Machines
   - Definition
   - Turing Machines in Scala
   - Configurations
   - Language of Turing Machines
   - Turing Machines as Computing Machines

Turing Machines



A **Turing machine (TM)** is a finite automaton with a **tape**.
It consists of the following three components:

1. A **finite automaton** with a deterministic transition function.
2. A **tape** is a one-dimensional infinite array of cells.
   - Each cell contains a **tape symbol**.
   - The **blank symbol** B is a special symbol representing an empty cell.
3. A **tape head** is a device that can read and write symbols on the tape.
   - It can move left or right one cell at a time.

Definition of Turing Machines

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 ($\Sigma \subseteq \Gamma$).
- $\delta : Q \times \Gamma \rightharpoonup Q \times \Gamma \times \{L, R\}$ 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**.

Definition of Turing Machines – Example

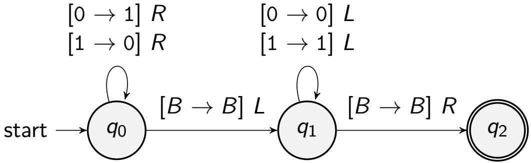
$$M_1 = (\{q_0, q_1, q_2\}, \{0, 1\}, \{0, 1, B\}, \delta, q_0, B, \{q_2\})$$

where

$$\begin{aligned} \delta(q_0, 0) &= (q_0, 1, R) & \qquad \delta(q_1, 0) &= (q_1, 0, L) \\ \delta(q_0, 1) &= (q_0, 0, R) & \qquad \delta(q_1, 1) &= (q_1, 1, L) \\ \delta(q_0, B) &= (q_1, B, L) & \qquad \delta(q_1, B) &= (q_2, B, R) \end{aligned}$$

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

$M_1 =$



Turing Machines in Scala

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

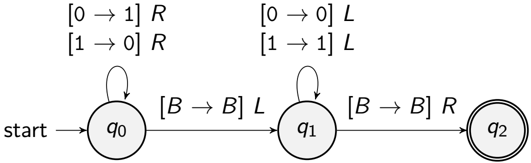
$$\delta : Q \times \Gamma \rightharpoonup Q \times \Gamma \times \{L, R\}$$

```scala
// The type definitions of states, symbols, tape symbols, and head moves
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,
  blankSymbol: TapeSymbol,
  finalStates: Set[State],
)
```

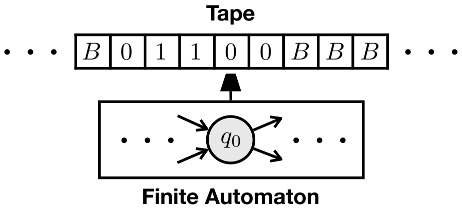
Turing Machines in Scala – Example

$M_1 =$




```scala
// An example of Turing machine
val tm1: TM = TM(
  states      = Set(0, 1, 2),
  symbols     = Set('0', '1'),
  tapeSymbols = Set('0', '1', 'B'),
  trans       = Map(
    (0, '0') -> (0, '1', R),  (1, '0') -> (1, '0', L),
    (0, '1') -> (0, '0', R),  (1, '1') -> (1, '1', L),
    (0, 'B') -> (1, 'B', L),  (1, 'B') -> (2, 'B', R),
  ),
  initState   = 0,
  blankSymbol = 'B',
  finalStates = Set(2),
)
```

Configurations



$\equiv$ 011 q_0 00

Definition (Configurations of Turing Machines)

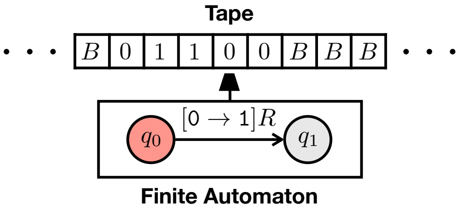
A **configuration** of a Turing machine M is a sequence of the form

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

where

- $q \in Q$ is the **current state**.
- $X_1 \cdots X_n \in \Gamma^*$ is the **portion of the tape** between the leftmost and the rightmost non-blank symbols not under the tape head.
- $X_i \in \Gamma$ is the symbol under the tape head.

One-Step Moves



$011 \ q_0 \ 00$

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.

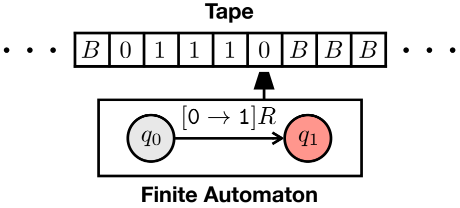
- If $\delta(q, X_i) = (p, Y, L)$,

$$X_1 \cdots X_{i-1} \ q \ X_i X_{i+1} \cdots X_n \vdash X_1 \cdots X_{i-2} \ p \ X_{i-1} Y X_{i+1} \cdots X_n$$

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

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

One-Step Moves



$$011\ q_0\ 00 \vdash 0111\ q_1\ 0$$

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

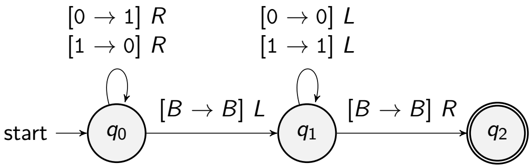
$$X_1 \cdots X_{i-1}\ q\ X_i X_{i+1} \cdots X_n \vdash X_1 \cdots X_{i-2}\ p\ X_{i-1} Y X_{i+1} \cdots X_n$$

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

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

One-Step Moves

$M_1 =$



$$
\begin{array}{lcll}
q_0\ 0110 & \vdash & 1\ q_0\ 110 & (\because \delta(q_0, 0) = (q_0, 1, R)) \\
 & \vdash & 10\ q_0\ 10 & (\because \delta(q_0, 1) = (q_0, 0, R)) \\
 & \vdash & 100\ q_0\ 0 & (\because \delta(q_0, 1) = (q_0, 0, R)) \\
 & \vdash & 1001\ q_0\ B & (\because \delta(q_0, 0) = (q_0, 1, R)) \\
 & \vdash & 100\ q_1\ 1 & (\because \delta(q_0, B) = (q_1, B, L)) \\
 & \vdash^* & q_1\ B1001 & (\because \delta(q_1, x) = (q_1, x, L) \text{ where } x = 0 \text{ or } 1) \\
 & \vdash & q_1\ 1001 & (\because \delta(q_1, B) = (q_2, B, R)) \\
 & \not\vdash & &
\end{array}
$$

We say that M_1 **halts** on input 0110 with output 1001 because it reaches a configuration having no more possible moves.

Language of Turing Machines

PLRG

Definition (Language of Turing Machines)

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

$$L(M) = \{w \in \Sigma^* \mid q_0 w \vdash^* \alpha\ q_f\ \beta \text{ for some } q_f \in F, \alpha, \beta \in \Gamma^*\}$$

Definition (Recursive Enumerable Languages)

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

For example, the language of a Turing machine M_1 is defined as follows:

$$L(M_1) = \{w \mid w \in \{0, 1\}^*\}$$

because M_1 reaches a final state on any input.

Language of Turing Machines

$$L(M) = \{w \in \Sigma^* \mid q_0 w \vdash^* \alpha\ q_f\ \beta \text{ for some } q_f \in F, \alpha, \beta \in \Gamma^*\}$$

```
// The type definitions of words, tapes, and configurations
type Word = String
type Tape = String
case class Config(state: State, prev: Tape, cur: TapeSymbol, next: Tape)
// A one-step move in a Turing machine
def move(tm: TM)(config: Config): Option[Config] = ...
// The initial configuration of a Turing machine
def initConfig(tm: TM)(word: Word): Config = word match
  case a <| x => Config(tm.initState, "", a, x)
  case _      => Config(tm.initState, "", tm.blankSymbol, "")
// The acceptance of a word by a Turing machine
def accept(tm: TM)(word: Word): Boolean =
  def aux(config: Config): Boolean =
    tm.finalStates.contains(config.state) || (move(tm)(config) match
      case None                => false
      case Some(nextConfig) => aux(nextConfig)
    )
  aux(initConfig(tm)(word))
// An example acceptance of a word "0110"
accept(tm1)("0110") // true
```

PLRG

Definition (Turing Computable Functions)

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

$$q_0\ w \vdash^* q_f\ f(w)$$

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

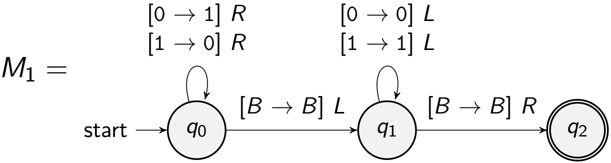
For example, the function $f : \{0,1\}^* \to \{0,1\}^*$ defined as follows is Turing-computable:

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

because there exists a Turing machine M_1 such that $q_0\ w \vdash^* q_f\ f(w)$ for some $q_f \in F$ and all $w \in \{0,1\}^*$. For instance,

$$q_0\ 0110 \vdash^* q_2\ 1001$$
$$q_0\ 1011100 \vdash^* q_2\ 0100011$$

Turing Machines as Computing Machines




```scala
// A multiple moves in a Turing machine
def moves(tm: TM)(config: Config): Config = move(tm)(config) match
  case None              => config
  case Some(nextConfig) => moves(tm)(nextConfig)

// A computation by a Turing machine
def compute(tm: TM)(w: Word): Option[Word] =
  val Config(state, prev, x, next) = moves(tm)(initConfig(tm)(w))
  if (!tm.finalStates.contains(state) || prev != "") None
  else Some(x + next)

// Examples of computations by Turing machines
compute(tm1)("0110")    // Some("1001")
compute(tm1)("1011100") // Some("0100011")
```

Summary

1. Turing Machines
   - Definition
   - Turing Machines in Scala
   - Configurations
   - Language of Turing Machines
   - Turing Machines as Computing Machines

Next Lecture

- Examples of Turing Machines

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