

# Lecture 10 – Turing Machines

NTIN071 Automata and Grammars

---

Jakub Bulín (KTIML MFF UK)

Spring 2025

*\* Adapted from the Czech-lecture slides by Marta Vomlelová with gratitude.  
The translation, some modifications, and all errors are mine.*

## Recap of Lecture 9

- Closure properties of context-free languages (including substitution, homomorphism, inverse homomorphism)
- Also closure properties of deterministic CFLs
- Dyck languages, a characterization of context-free languages

## CHAPTER 3: TURING MACHINES

---

## 3.1 Turing machine

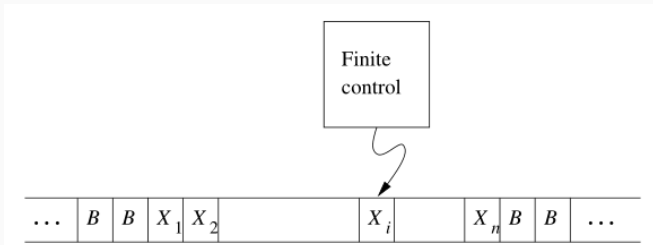
---

# History and motivation

1931–1936 Gödel, Church, Turing, Kleene: formalize ‘algorithms’

**Turing machine:** a general model of any computer

- a two-way infinite **tape** (sequential memory)
- a **head** to read/write, moves in both directions
- a control unit (finite state)



Other formalizations: RAM,  $\lambda$ -calculus, partially recursive functions

**Computability theory:** what problems can[t] computers solve?

# The definition

A **Turing Machine (TM)** is  $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$  where:

- $Q$  is a finite, nonempty set of **states**
- $\Sigma$  is a finite, nonempty **input alphabet**
- $\Gamma$  is a finite, nonempty **tape alphabet**,  $\Gamma \supseteq \Sigma$ ,  $Q \cap \Gamma = \emptyset$
- $\delta: (Q \setminus F) \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$  is the (partial) **transition function**, i.e., one instruction is  $\delta(q, X) = (p, Y, D)$  where:
  - $q \in Q \setminus F$  is the current state [no transitions out of final states]
  - $X \in \Gamma$  is the tape symbol in the current cell
  - $p \in Q$  is the next state to switch to
  - $Y \in \Gamma$  is the tape symbol to rewrite  $X$  with in the current cell
  - $D \in \{L, R\}$  is the **direction** in which the head then moves
- $q_0 \in Q$  is the **start state**
- $B \in \Gamma \setminus \Sigma$  is the **blank symbol**, initially written in all but finitely many cells that hold the input symbols
- $F \subseteq Q$  are the **final** or **accepting** states

# Describing computation: configurations

**Recall** computation graph: vertices=configurations, arcs=moves  $\vdash$

A configuration of a TM is a finite string

$$X_1 X_2 \dots X_{i-1} q X_i X_{i+1} \dots X_n$$

- $q \in Q$  is the current state
- $X_1 \dots X_n \in \Gamma^*$  describe the contents of the relevant portion of the tape, that is, between
  - the first (leftmost) non-blank symbol or head position, and
  - the last (rightmost) non-blank symbol or head position
- the tape head is scanning the  $i$ -th symbol  $X_i \in \Gamma$

## Describing computation: moves

For **moves** of a TM  $M$ , use same notation as for PDA:  $\vdash_M, \vdash_M^*, \vdash^*$

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

$$X_1 X_2 \dots X_{i-1} q X_i X_{i+1} \dots X_n \vdash_M X_1 X_2 \dots X_{i-2} p X_{i-1} \mathbf{Y} X_{i+1} \dots X_n$$

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

$$X_1 X_2 \dots X_{i-1} q X_i X_{i+1} \dots X_n \vdash_M X_1 X_2 \dots X_{i-1} \mathbf{Y} p X_{i+1} \dots X_n$$

And  $\vdash_M^*$  is a reflexive, transitive closure of  $\vdash_M$  (oriented **path** in the computation graph).

**initial configuration:**  $q_0 w$  for the input word  $w \in \Sigma^*$

**accepting configurations:** those where  $q \in F$ , any tape contents (i.e., in our definition, the TM doesn't need to 'clean' the tape)



# The language, an example

The language **recognized by** a TM  $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$  is:

$$L(M) = \{w \in \Sigma^* \mid q_0 w \vdash_M^* \alpha p \beta, p \in F, \alpha, \beta \in \Gamma^*\}$$

A language is **recursively enumerable** if it is recognized by some TM

## Example

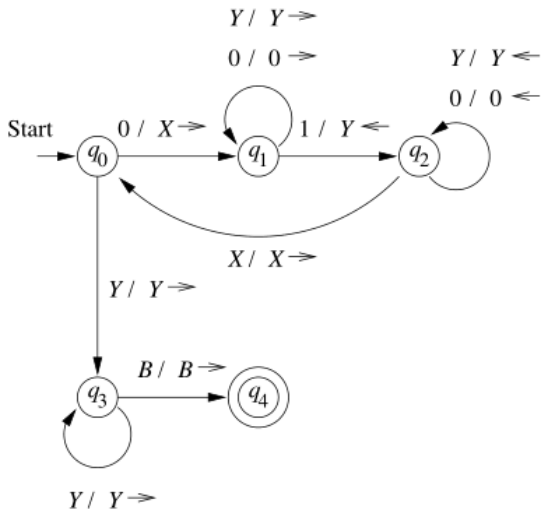
The following TM accepts the language  $L = \{0^n 1^n \mid n \geq 1\}$ :

$$M = (\{q_0, q_1, q_2, q_3, q_4\}, \{0, 1\}, \{0, 1, X, Y, B\}, \delta, q_0, B, \{q_4\})$$

$\delta$	0	1	X	Y	B
$q_0$	$(q_1, X, R)$	—	—	$(q_3, Y, R)$	—
$q_1$	$(q_1, 0, R)$	$(q_2, Y, L)$	—	$(q_1, Y, R)$	—
$q_2$	$(q_2, 0, L)$	—	$(q_0, X, R)$	$(q_2, Y, L)$	—
$q_3$	—	—	—	$(q_3, Y, R)$	$(q_4, B, R)$
$q_4$	—	—	—	—	—

# Transition diagram

**nodes** are states, **arcs**  $q \rightarrow p$  are labeled by  $X/YD$  for all  $\delta(q, X) = (p, Y, D)$  (use  $D \in \{\leftarrow, \rightarrow\}$  instead of  $\{L, R\}$ )



# The program explained

Recognizes  $L = \{0^n 1^n \mid n > 0\}$ .

On tape always  $X^*0^*Y^*1^*$ .

Repeatedly rewrite a 0 to  $X$ ,  
and the corresponding 1 to  $Y$ :

$q_0$ : rewrite 0 to  $X$ , switch to  $q_1$

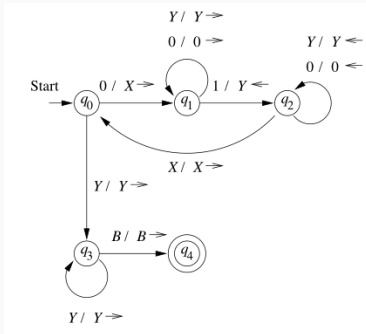
$q_1$ : search forward for the first 1, rewrite to  $Y$ , switch to  $q_2$

$q_2$ : search backward for the last  $X$ , go forward, switch to  $q_0$

If  $q_0$  sees 0, continue as above, if it sees  $Y$ , switch to  $q_3$

$q_3$ : moves to the end to check that there are no remaining 1s

- if  $q_3$  finds  $B$ , switch to  $q_4$ , accept (accepting state)
- if  $q_3$  finds 1, fail (no instruction, not accepting state)



## Computation examples: $w = 0011$ and $w = 0010$

$q_0 0011 \vdash$

$Xq_1 011 \vdash$

$X0q_1 11 \vdash$

$Xq_2 0Y1 \vdash$

$q_2 X0Y1 \vdash$

$Xq_0 0Y1 \vdash$

$XXq_1 Y1 \vdash$

$XXYq_1 1 \vdash$

$XXq_2 YY \vdash$

$Xq_2 XYY \vdash$

$XXq_0 YY \vdash$

$XXYq_3 Y \vdash$

$XXYYq_3 B \vdash$

$XXYYBq_4 B \quad \dots \text{accepted}$

$q_0 0010 \vdash$

$Xq_1 010 \vdash$

$X0q_1 10 \vdash$

$Xq_2 0Y0 \vdash$

$q_2 X0Y0 \vdash$

$Xq_0 0Y0 \vdash$

$XXq_1 Y0 \vdash$

$XXYq_1 0 \vdash$

$XXY0q_1 B \quad \dots \text{fail (no instruction)}$

# Recognizing regular and context-free languages

## Regular languages:

- simulate a DFA, move always right, never write on the tape
- if we see  $B$ , we are at the end of input: if the DFA is in accepting state, switch to a new accepting state  $q_F$
- (note: in a TM, the accepting state  $q_F$  cannot have outgoing transitions; in a DFA it is allowed)

### Example

$L = \{a^{2^n} \mid n \geq 0\}$  recognized by the following TM:

$M = (\{q_0, q_1, q_F\}, \{a\}, \{a, B\}, \delta, q_0, B, \{q_F\})$  with transitions

- $\delta(q_0, B) = (q_F, B, R)$
- $\delta(q_0, a) = (q_1, a, R)$
- $\delta(q_1, a) = (q_0, a, R)$

**Context-free languages:** simulate a PDA, simulate an auxiliary tape to hold the stack contents (how?? later)

# Turing machines with output

Turing Machines can give output, i.e., compute a (partial) function

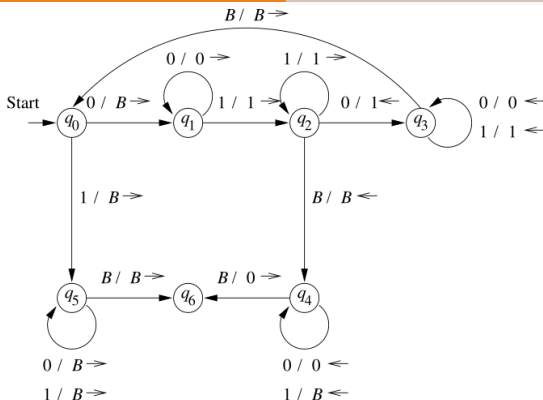
$$f_M : \Sigma^* \rightarrow \Gamma^*$$

where  $f_M(w)$  is defined as follows:

- if  $M$  halts, then  $f_M(w)$  equals the **contents of the tape** at the end of computation (everything between the first and last non-blank symbol, or  $f_M(w) = \epsilon$  if the tape is all blanks)
- if  $M$  does not halt, then  $f_M(w)$  is **undefined**

Note: the set of accepting states  $F$  is ignored, often omitted

## Example: computing **monus** $m \dot{-} n = \max(m - n, 0)$



$m, n$  encoded in unary

at the start:  $0^m 1 0^n$

at the end:  $0^{m \dot{-} n}$

find leftmost 0, delete

search right for a 1

if found, continue

find a 0, rewrite by 1

return left

if no 0 found, either left or right:

right: replace all 1s by B

left ( $m < n$ ): replace all 1s and 0s

by B (leave the tape blank)

# Halting, recursively enumerable and recursive languages

## Definition

A TM **halts** if it enters a state  $q$ , scanning a tape symbol  $X$ , and there is no transition in this situation, i.e.,  $\delta(q, X)$  is undefined.

A TM halts whenever it gets to an accepting state (no outgoing transitions allowed). In general, we cannot require that a TM always halts, even if it does not accept.

(Until a TM halts, we do not know whether it will accept or not.)

## Definition

A language  $L$  is:

- **recursively enumerable** if it is **recognized** by some TM
- **recursive** if there exists a TM  $M$  that recognizes  $L$  and *halts on every input  $w \in \Sigma^*$* ; in that case, we also say  $M$  **decides**  $L$



## We will see that...

context-sensitive  $\subsetneq$  recursive  $\subsetneq$  recursively  
enumerable  $\subsetneq$  all languages

- Every context-sensitive language is recursive.
- Not all recursive languages are context sensitive.
- Every recursive language is recursively enumerable.
- Not all recursively enumerable languages are recursive.
- A language is recursively enumerable, iff it is generated by a Type 0 grammar in the Chomsky hierarchy.
- Not all languages are recursively enumerable.

## 3.2 Variants of TMs

---

## Construction tricks

---

## Construction trick: storage in the FA unit

The following TM recognizes the language  $L = L(01^* + 10^*)$ ; it remembers 0, 1, or  $B$  in its state:

$$M = (\{q_0, q_1\} \times \{0, 1, B\}, \{0, 1\}, \{0, 1, B\}, \delta, [q_0, B], B, \{[q_1, B]\})$$

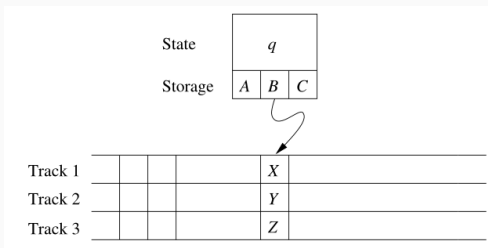
$\delta$	0	1	B
$\rightarrow [q_0, B]$	$([q_1, 0], 0, R)$	$([q_1, 1], 1, R)$	
$[q_1, 0]$		$([q_1, 0], 1, R)$	$([q_1, B], B, R)$
$[q_1, 1]$	$([q_1, 1], 0, R)$		$([q_1, B], B, R)$
$*[q_1, B]$			

In general, we can store a finite number of variables with finitely many possible values (e.g. Boolean, input symbols, etc.): the state is a tuple, entries are values of the variables.

## Construction trick: tape with multiple tracks

To split the tape into two tracks, each of which can hold a tape symbol:  $\Gamma' = \Gamma \cup \{ \overset{X}{\underset{Y}{\mid}} \mid X, Y \in \Gamma \}$ . At the beginning, traverse the input changing  $a$  to  $\overset{B}{\underset{a}{\mid}}$ , then return.

Or say  $\Gamma = \{0, 1, B\}$  and we want to put a mark  $*$  over certain digits. Then  $\Gamma' = \{0, 1, B, \overset{*}{\underset{0}{\mid}}, \overset{*}{\underset{1}{\mid}}\}$ . (We write  $[X, Y]$  for  $\overset{X}{\underset{Y}{\mid}}$ .)



**NB:** This is different from (but will be used to simulate!) multiple tapes with heads moving independently.

**Example:**  $L_{WCW} = \{wCW \mid w \in \{0,1\}^+\}$

Put a mark '\*' over the letter being checked, store it in memory.  
(We skip the preprocessing, assume  $a$  is already  $[B, a]$ .)

$M = (\{q_0, \dots, q_9\} \times \{0, 1, B\}, \{[B, 0], [B, 1], [B, c]\}, \{B, *\} \times \{0, 1, B, c\}, \delta, [q_1, B], [B, B], \{[q_9, B]\})$  where  $\delta$  is ( $a, b \in \Sigma$ ):

- $\delta([q_1, B], [B, a]) = ([q_2, a], [*, a], R)$  pick up the symbol  $a$
- $\delta([q_2, a], [B, b]) = ([q_2, a], [B, b], R)$  move right, search for  $c$
- $\delta([q_2, a], [B, c]) = ([q_3, a], [B, c], R)$  continue right, state changed
- $\delta([q_3, a], [*, b]) = ([q_3, a], [*, b], R)$  continue right
- $\delta([q_3, a], [B, a]) = ([q_4, B], [*, a], L)$  match symbols, clear memory
- $\delta([q_4, B], [*, a]) = ([q_4, B], [*, a], L)$  go left
- $\delta([q_4, B], [B, c]) = ([q_5, B], [B, c], L)$   $c$  found, continue left
- are all symbols left and right checked? branch adequately

## Example continued

- are all symbols left and right checked? branch adequately
- $\delta([q_5, B], [B, a]) = ([q_6, B], [B, a], L)$  left symbol unchecked
- $\delta([q_6, B], [B, a]) = ([q_6, B], [B, a], L)$  proceed left
- $\delta([q_6, B], [*, a]) = ([q_1, B], [*, a], R)$  start again
- $\delta([q_5, B], [*, a]) = ([q_7, B], [*, a], R)$  symbol left from  $c$  checked, go right
- $\delta([q_7, B], [B, c]) = ([q_8, B], [B, c], R)$  proceed right
- $\delta([q_8, B], [*, a]) = ([q_8, B], [*, a], R)$  proceed right
- $\delta([q_8, B], [B, B]) = ([q_8, B], [B, B], R)$  accept

# Multi-tape Turing Machines

---



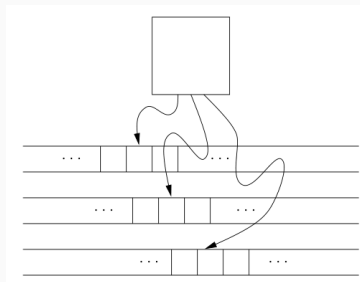
# A multi-tape TM

## Initial configuration:

- input on first tape, others blank
- first head scans the first input letter
- FA unit in the initial state

## One step:

- FA unit switches to the new state
- on each tape rewrite independently
- each head moves independently



**Transition function:**  $\delta: (Q \setminus F) \times \Gamma^n \rightarrow Q \times \Gamma^n \times \{L, R\}^n$

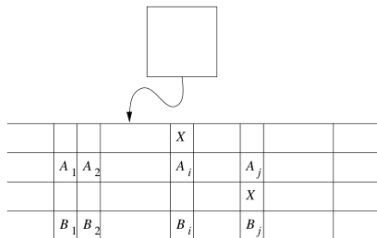
## Theorem

*Any language recognized by a multi-tape TM is also recognized by some (single-tape) Turing machine.*

## Proof: simulate using multiple tracks, mark head positions

Split the single tape into  $2k$  tracks

- odd: mark  $i^{th}$  head position
- even tracks:  $i^{th}$  tape contents



To simulate one step, visit all heads and store in the FA unit:

- the simulated state
- the number of head marks to the left of us
- for every  $i$ , the symbol under  $i^{th}$  head

Then we know enough to simulate one step. We visit all heads again to rewrite and move them. □

Simulating  $n$  steps of a  $k$ -tape TM takes  $O(n^2)$  moves. (One step takes  $4n + 2k$ , heads no further than  $2n$ ; read, write, move marks).

# Nondeterministic Turing Machines

---

# Nondeterministic Turing Machine

## Definition

A **nondeterministic Turing machine** is  $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$  where  $Q, \Sigma, \Gamma, q_0, B, F$  are the same as for a determ. TM, and

$$\delta : (Q - F) \times \Gamma \rightarrow \mathcal{P}(Q \times \Gamma \times \{L, R\})$$

A word  $w \in \Sigma^*$  **is accepted** by the nondeterministic TM  $M$  iff there exists an accepting sequence of moves  $q_0 w \vdash^* \alpha p \beta$ ,  $p \in F$ .

## Theorem

*For every nondeterministic Turing machine  $M_N$  there exist a deterministic TM  $M_D$  such that  $L(M_N) = L(M_D)$ .*

**Proof idea:** Subset construction not possible, the tape is infinite. Instead, BFS of the computation graph. In  $k$  steps, at most  $m^k$  configs for  $m = \max_{q \in Q, x \in \Gamma} |\delta(q, x)|$ . Store on a tape & process.

# Proof

Breadth-first search of all possible computations of  $M_N$   
(configurations reachable from the initial configuration).

Use a two-tape TM:

- **first tape:** maintain a queue of open vertices (configurations) separated by a special symbol
- **second tape:** scratch tape for processing one configuration

To process one configuration:

- copy it to the scratch tape, dequeue (erase)
- read the state and scanned symbol
- if accepting state, accept and halt
- for each possible transition of  $\delta_N$ : perform the move and enqueue the resulting config (write at the end of tape 1)

[The transition function  $\delta_N$  is encoded in  $\delta_D$ .]

# One-way infinite tape

## Theorem

*Every recursively enumerable language is also recognized by a TM whose head never moves left of its initial position.*

$X_0$	$X_1$	$X_2$	$\dots$
*	$X_{-1}$	$X_{-2}$	$\dots$

**Proof idea:** Two-track tape, lower track for negative indices. Remember in state if positive or negative. If negative,  $L$  means moving to the right and vice versa. Special cases around 0.  $\square$

**Exercise:** Consider a variant of Turing Machine with an infinite 2D board instead of a tape, moves  $\{\uparrow, \rightarrow, \downarrow, \leftarrow\}$ . Show how to simulate by the standard TM.

# Summary of Lecture 10

- Turing machine: two-way infinite tape, read, write, move head
- Accept iff in a final state; configurations
- TMs with output, computing a function
- Recursively enumerable vs. recursive languages (always halt).
- Construction tricks:
  - storage in state
  - multiple tracks (on a single tape)
- Variants of TMs:
  - multi-tape (independent heads),
  - nondeterministic (accept iff some choices lead to final state)