Lecture 10 – Turing Machines, Linear-bounded automata

NTIN071 Automata and Grammars

Jakub Bulín (KTIML MFF UK)

Spring 2024

** 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

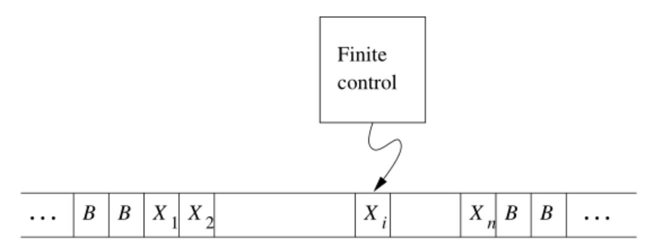
3.1 Turing machine

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, λ-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
- Σ is a finite, nonempty input alphabet
- Γ is a finite, nonempty tape alphabet, $\Gamma \supseteq \Sigma$, $Q \cap \Gamma = \emptyset$
- $\delta\colon (Q \setminus F) \times \Gamma \to Q \times \Gamma \times \{L, R\}$ 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

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

- $q \in Q$ is the current state
- $X_1 \ldots X_n \in \Gamma^*$ 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 $\delta(q, X_i) = (p, Y, L)$:

  $X_1X_2 \dots X_{i-1}qX_iX_{i+1} \dots X_n \vdash_M X_1X_2 \dots X_{i-2}pX_{i-1}\mathbf{Y}X_{i+1} \dots X_n$

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

  $X_1X_2 \dots X_{i-1}qX_iX_{i+1} \dots X_n \vdash_M X_1X_2 \dots X_{i-1}\mathbf{Y}pX_{i+1} \dots X_n$

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

initial configuration: q_0w 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

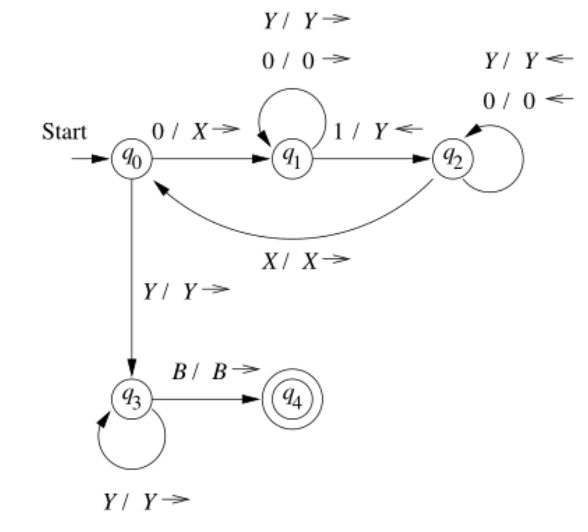
The following TM accepts the language $L = \{0^n 1^n \mid n \geq 1\}$:

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

δ	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	–	–	–	–	–

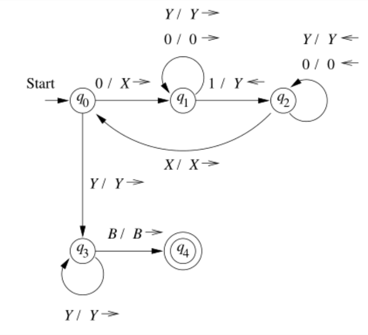
Transition diagram

nodes are states, arcs $q \to p$ 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 $L = \{0^n1^n \mid n > 0\}$.



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 $w = 0010$

$$
\begin{aligned}
&q_0 0011 \vdash \\
&Xq_1 011 \vdash \\
&X0q_1 11 \vdash \\
&Xq_2 0Y1 \vdash \\
&q_2 X0Y1 \vdash \\
&Xq_0 0Y1 \vdash \\
&XXq_1 Y1 \vdash \\
&XXYq_1 1 \vdash \\
&XXq_2 YY \vdash \\
&Xq_2 XYY \vdash \\
&XXq_0 YY \vdash \\
&XXYq_3 Y \vdash \\
&XXYYq_3 B \vdash \\
&XXYYBq_4 B \quad \ldots \text{accepted}
\end{aligned}
$$

$$
\begin{aligned}
&q_0 0010 \vdash \\
&Xq_1 010 \vdash \\
&X0q_1 10 \vdash \\
&Xq_2 0Y0 \vdash \\
&q_2 X0Y0 \vdash \\
&Xq_0 0Y0 \vdash \\
&XXq_1 Y0 \vdash \\
&XXYq_1 0 \vdash \\
&XXY0q_1 B \quad \ldots \text{fail (no instruction)}
\end{aligned}
$$

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

$L = \{a^{2n} \mid n \geq 0\}$ 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

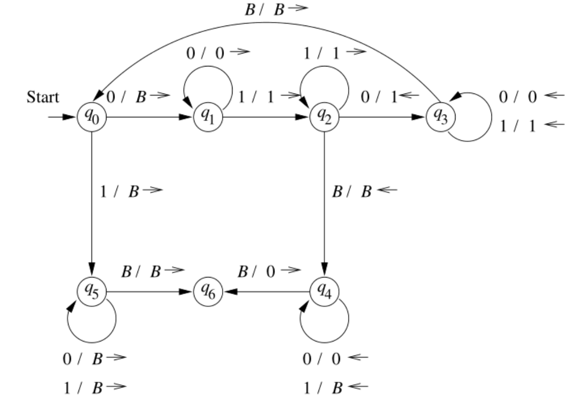
$$f_M : \Sigma^* \to \Sigma^*$$

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 at TM M that recognizes L and *halts on every input* $w \in \Sigma^*$

3.2 Variants of TMs

3.3 TMs and grammars