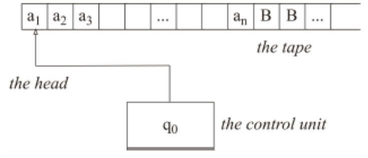
Automata Theory and Formal Languages
Class 7

Marcin Luckner, PhD
mluckner@mini.pw.edu.pl

Version 1.2
18 grudnia 2020

Basic Turing machine

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



Rysunek 1: Basic Turing machine

- Q the finite set of states
- Σ the set of input symbols, a subset of Γ not including B
- Γ the finite set of allowable tape symbols
- δ the transition function, a mapping from $Q \times \Gamma$ to $Q \times \Gamma \times \{L, R\}$
- q_0 the start state
- B the blank symbol $B \in \Gamma$
- F the set of final states $F \subseteq Q$

TM with the stop property

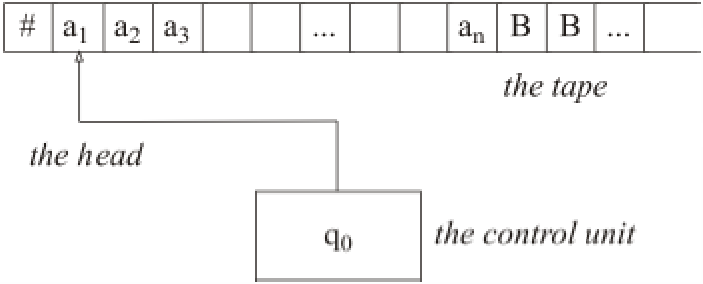
$$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F, R)$$

F includes only one accepting state q_A

R includes a special non–accepting state q_R

TM with guard

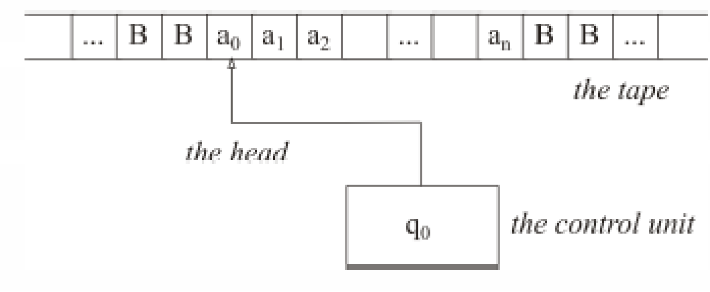
$$M = (Q, \Sigma, \Gamma, \delta, q_0, B, \#, F)$$



Rysunek 2: Turing machine with guard

$\#$ the guard symbol $\# \in \Gamma \wedge \# \notin \Sigma$

Two–way infinite tape



Rysunek 3: Turing machine with two–way infinite tape

Transition table

- A transition table gives the details of computations.
- Each row describes an unique state.
- Each column corresponds to tape symbols.
- The transition results is given in a table cell as a triple: a new state, a new symbol, a movement.
- The calculations are finished when the final state is reached.

δ	B	0	1
q_0	$(q_1, 0, L)$	$(q_0, 1, R)$	$(q_0, 0, R)$
q_1	(q_A, B, R)	$(q_1, 0, L)$	$(q_1, 1, L)$

Task

- Design a Turing machine to compute $f(n) = 2^n$.
- Steps:
  1. Analyse the problem
  2. Select a model
  3. Sketch an algorithm
  4. Select coding
  5. Design calculations

Problem analysis

- The calculations give the following results for the first values:

n	$f(n)$
0	1
1	2
2	4
3	8

- We see that $f(n+1) = 2 * f(n)$, therefore a recursive algorithm can be used.
- We use Turing machine with two–way infinite tape.

Algorithm sketch

1. The input defines the number of iterations $c = n$.
2. The initial output is set as $o = 1$.
3. For each iteration $1 \ldots n$:
   3.1 $c = c - 1$
   3.2 $o = 2o$
4. Remove temporary variables
5. Set o as the result

Binary vs unary coding

- We can code input and output using binary or unary coding.
- If our function is not a translator, the input and output coding should be the same.
- In our case, one coding is better for operation on the result, the second is better for counting.

n	$f(n)$	u(n)	u(f(n))	b(n)	b(f(n))
0	1	B	1	0	1
1	2	1	11	1	10
2	4	11	1111	10	100
3	8	111	11111111	11	1000

- Let's try the unary coding.

Transition table I

δ	B	1	X
q_0	$(q_?, 1, L)$	$(q_1, 1, R)$	–
q_1	(q_2, X, R)	$(q_1, 1, R)$	–
q_2	$(q_3, 1, L)$	–	–
q_3	(q_4, B, R)	$(q_3, 1, L)$	(q_3, X, L)
q_4	–	(q_5, B, R)	$(q_?, B, R)$
q_5	–	$(q_5, 1, R)$	(q_6, X, R)
q_6	–	(q_7, D, R)	–

Transition table II

δ	B	1	X	D
q_0	$(q_?, 1, L)$	$(q_1, 1, R)$	—	—
q_1	(q_2, X, R)	$(q_1, 1, R)$	—	—
q_2	$(q_3, 1, L)$	—	—	—
q_3	(q_4, B, R)	$(q_3, 1, L)$	(q_3, X, L)	—
q_4	—	(q_5, B, R)	$(q_?, B, R)$	—
q_5	—	$(q_5, 1, R)$	(q_6, X, R)	—
q_6	$(q_?, B, L)$	(q_7, D, R)	—	(q_6, D, R)
q_7	(q_8, D, L)	$(q_7, 1, R)$	—	(q_7, D, R)
q_8	—	$(q_8, 1, L)$	(q_6, X, R)	(q_8, D, L)

Transition table III

δ	B	1	X	D
q_0	$(q_?, 1, L)$	$(q_1, 1, R)$	—	—
q_1	(q_2, X, R)	$(q_1, 1, R)$	—	—
q_2	$(q_3, 1, L)$	—	—	—
q_3	(q_4, B, R)	$(q_3, 1, L)$	(q_3, X, L)	—
q_4	—	(q_5, B, R)	$(q_?, B, R)$	—
q_5	—	$(q_5, 1, R)$	(q_6, X, R)	—
q_6	$(\mathbf{q_9}, B, L)$	(q_7, D, R)	—	(q_6, D, R)
q_7	(q_8, D, L)	$(q_7, 1, R)$	—	(q_7, D, R)
q_8	—	$(q_8, 1, L)$	(q_6, X, R)	(q_8, D, L)
q_9	—	—	(q_{10}, X, L)	$(q_9, 1, L)$
q_{10}	(q_4, B, R)	$(q_{10}, 1, L)$	—	—

Transition table IV

δ	B	1	X	D
q_0	$(\mathbf{q_4}, 1, L)$	$(q_1, 1, R)$	—	—
q_1	(q_2, X, R)	$(q_1, 1, R)$	—	—
q_2	$(q_3, 1, L)$	—	—	—
q_3	(q_4, B, R)	$(q_3, 1, L)$	(q_3, X, L)	—
q_4	$(\mathbf{q_A}, \mathbf{B}, \mathbf{R})$	(q_5, B, R)	$(\mathbf{q_A}, B, R)$	—
q_5	—	$(q_5, 1, R)$	(q_6, X, R)	—
q_6	$(\mathbf{q_9}, B, L)$	(q_7, D, R)	—	(q_6, D, R)
q_7	(q_8, D, L)	$(q_7, 1, R)$	—	(q_7, D, R)
q_8	—	$(q_8, 1, L)$	(q_6, X, R)	(q_8, D, L)
q_9	—	—	(q_{10}, X, L)	$(q_9, 1, L)$
q_{10}	(q_4, B, R)	$(q_{10}, 1, L)$	—	—

Final model

$$
\begin{aligned}
M = (Q = & \quad \{q_0, \ldots, q_{10}, q_A\}, \\
\Sigma = & \quad \{1\}, \\
\Gamma = & \quad \{1, B, X, D\}, \\
& \quad \delta, \\
& \quad q_0, \\
& \quad B, \\
F = & \quad \{q_A\})
\end{aligned}
$$

Task

Design Turing machines to recognise the following language
$L = \{a^i b^j c^k : i = j = k\}$ over alphabet $\Sigma = \{a, b, c\}$

Transition table I

δ	a	b	c	B	X
q_0	(q_1, B, R)	q_R	q_R	q_A	—
q_1	(q_1, a, R)	(q_2, X, R)	q_R	q_R	(q_1, X, R)
q_2	q_R	(q_2, b, R)	(q_3, c, R)	q_R	(q_2, X, R)
q_3	q_R	q_R	(q_3, c, R)	(q_4, B, L)	—
q_4	q_R	q_R	(q_5, B, L)	q_R	q_R
q_5	(q_5, a, L)	(q_5, b, L)	(q_5, c, L)	(q_0, B, R)	(q_5, X, L)

Transition table II

δ	a	b	c	B	X
q_0	(q_1, B, R)	q_R	q_R	q_A	$(\mathbf{q_6}, \mathbf{B}, \mathbf{R})$
q_1	(q_1, a, R)	(q_2, X, R)	q_R	q_R	(q_1, X, R)
q_2	q_R	(q_2, b, R)	(q_3, c, R)	q_R	(q_2, X, R)
q_3	q_R	q_R	(q_3, c, R)	(q_4, B, L)	$-$
q_4	q_R	q_R	(q_5, B, L)	q_R	q_R
q_5	(q_5, a, L)	(q_5, b, L)	(q_5, c, L)	(q_0, B, R)	(q_5, X, L)
q_6	q_R	q_R	q_R	q_A	(q_6, B, R)

Final model

$$
\begin{aligned}
M = (Q = & \quad \{q_0, \ldots, q_6, q_A, q_R\}, \\
\Sigma = & \quad \{a, b, c\}, \\
\Gamma = & \quad \{a, b, c, B, X\}, \\
& \quad \delta, \\
& \quad q_0, \\
& \quad B, \\
F = & \quad \{q_A\}, \\
R = & \quad \{q_R\})
\end{aligned}
$$

Assessments I

1. Design Turing machines to compute the following functions in the basic model
   1.1 $f(n) = n!$
   1.2 $f(n, m) = n * m$
   1.3 $f(n, m) = n - m$, function returns 0 when $m > n$
2. Redesign machines from Task 1 into models
   2.1 with the guard
   2.2 with a two–way infinite tape

Assignments II

3. Design Turing machines to recognise the following languages
   - 3.1 The language L over alphabet $\Sigma = \{0, 1\}$ of words with an equal number of 0's and 1's
   - 3.2 $L = \{a^i b^j c^k : k = \max(i, j)\}$ over alphabet $\Sigma = \{a, b, c\}$

Transition table 1.3

- Design basic Turing machine to compute $f(n) = n - m$, function returns 0 when $m > n$
- Input: $1111X111$

δ	X	1	B	S	Z
q_0	(q_1, Z, R)	(q_4, S, R)	–	–	–
q_1	–	(q_1, B, R)	(q_2, B, L)	–	–
q_2	–	$(q_2, 1, L)$	(q_2, B, L)	$(q_3, 1, R)$	(q_3, B, R)
q_3	–	$(q_A, 1, L)$	(q_A, B, L)	–	–
q_4	(q_5, X, R)	$(q_4, 1, R)$	(q_4, B, L)	–	–
q_5	–	$(q_5, 1, R)$	(q_6, B, L)	–	–
q_6	(q_2, B, L)	(q_7, B, L)	–	–	–
q_7	(q_8, X, L)	$(q_7, 1, L)$	–	–	–
q_8	–	(q_4, B, R)	(q_8, B, L)	(q_1, Z, R)	–

Idea 1.1

- Design basic Turing machine to compute $f(n) = n!$
- $f(n) = f(n-1) * n$
  1. $1111 (n=4)
  2. $X111 (skip multiplying n)
  3. $XXY1 (in each iteration skip existing segment)
  4. $xxyjZZZZ (multiply first segment)
  5. $xxyjZZZZZZZZ (multiply first segment n-2 times)
  6. $XXY1YYYYYYYY (restore symbols)
  7. $XXYYYYYYYYYYZZZZZZZZZZZZ (next iteration)
  8. $111111111111111111111111 (result)

Transition table 1.1

δ	1	X	Y	Z	B	j	x	y	\$
q_0	(q_1, X, R)	–	–	–	$(q_2, 1, L)$	–	–	–	–
q_1	(q_3, X, R)	–	–	–	(q_2, B, L)	–	–	–	–
q_2	–	$(q_2, 1, L)$	$(q_2, 1, L)$	$(q_2, 1, L)$	–	–	–	–	$(q_A, \$, R)$
q_3	(q_4, Y, L)	(q_3, X, R)	(q_3, X, R)	(q_3, Z, R)	(q_2, B, L)	–	–	–	–
q_4	–	(q_4, X, L)	(q_4, Y, L)	–	–	–	–	–	$(q_5, \$, R)$
q_5	(q_6, j, R)	(q_6, x, R)	(q_6, y, R)	(q_8, Y, R)	–	(q_5, j, R)	(q_5, x, R)	(q_5, y, R)	–
q_6	$(q_6, 1, R)$	(q_6, X, R)	(q_6, Y, R)	(q_6, Z, R)	(q_7, Z, L)	(q_6, j, R)	(q_6, x, R)	(q_6, y, R)	–
q_7	$(q_7, 1, L)$	(q_7, X, L)	(q_7, Y, L)	(q_7, Z, L)	–	(q_7, j, L)	(q_7, x, L)	(q_7, y, L)	$(q_5, \$, R)$
q_8	–	–	–	(q_8, Y, R)	(q_9, B, L)	–	–	–	–
q_9	$(q_9, 1, L)$	(q_9, X, L)	(q_9, Y, L)	–	–	$(q_9, 1, L)$	(q_9, X, L)	(q_9, Y, L)	$(q_3, \$, R)$