Automata Theory and Formal Languages

Summer Term 2026

17th Lecture

Multi-Tape and Universal Turing Machines

Prof. Dr. Matthias Mnich

Institute for Algorithms and Complexity

Turing Machine

Multi Tape Turing Machine

In practice many problems are difficult to program on a regular turing mashine. In addition, the algorithmic complexity is usually high.

Multi Tape Turing Machine

In practice many problems are difficult to program on a regular turing mashine. In addition, the algorithmic complexity is usually high.

Idea: Extension of the turing machine to several tapes (and heads)

Multi Tape Turing Machine

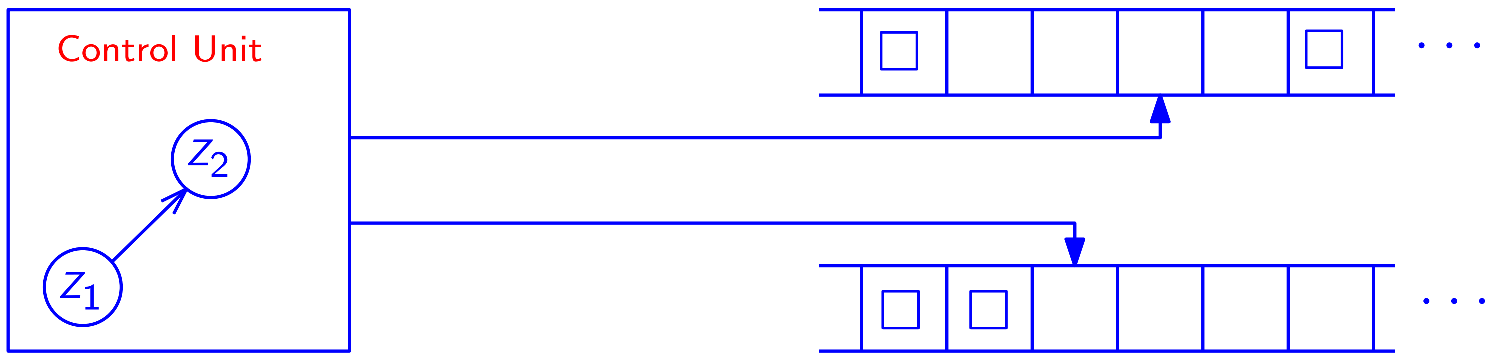
In practice many problems are difficult to program on a regular turing mashine. In addition, the algorithmic complexity is usually high.

Idea: Extension of the turing machine to several tapes (and heads)

k-tape TM:

$$\delta : Z \times \Gamma^k \to Z \times \Gamma^k \times \{L, R, N\}^k$$

$$\delta(z, a_1, \ldots, a_k) = (z', b_1, \ldots, b_k, x_1, \ldots x_k)$$



Multi Tape Turing Machine

Remark

The head positions for each tape do not be the same.

Multi Tape Turing Machine

Remark

The head positions for each tape do not be the same.

Theorem

Any k-tape turing machine can be simulated by a single tape turing machine.

Multi Tape Turing Machine

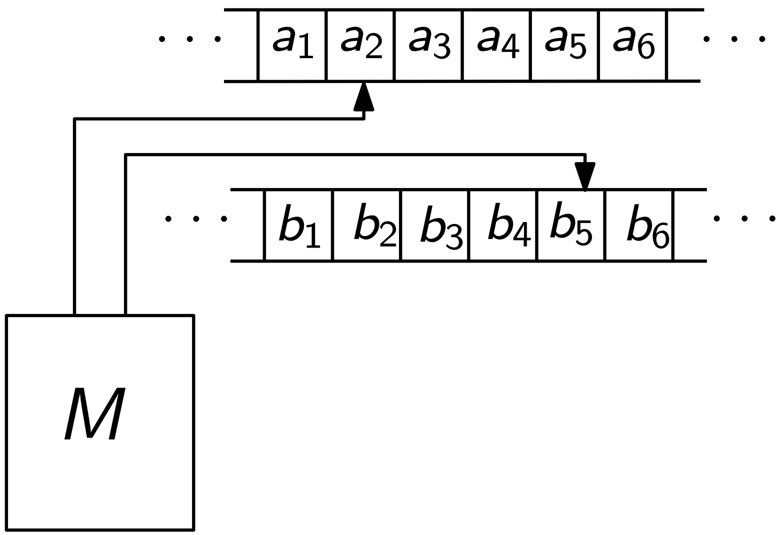
Remark

The head positions for each tape do not be the same.

Theorem

Any k-tape turing machine can be simulated by a single tape turing machine.

Sketch of Proof

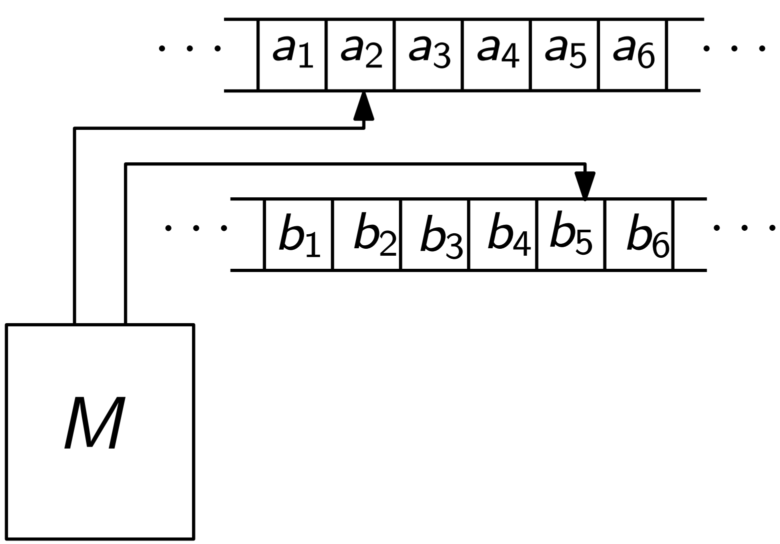


Multi Tape Turing Machine

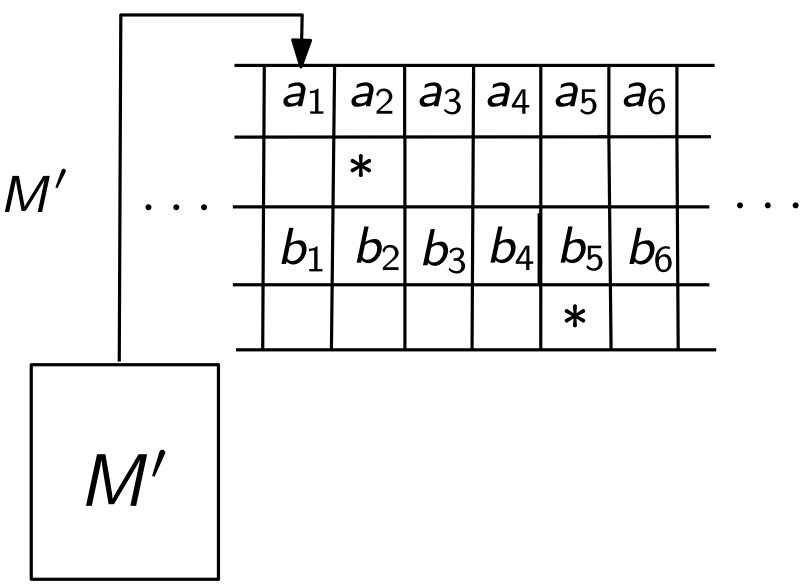
Remark
The head positions for each tape do not be the same.

Theorem
Any k-tape turing machine can be simulated by a single tape turing machine.

Sketch of Proof



M can be simulated by M'

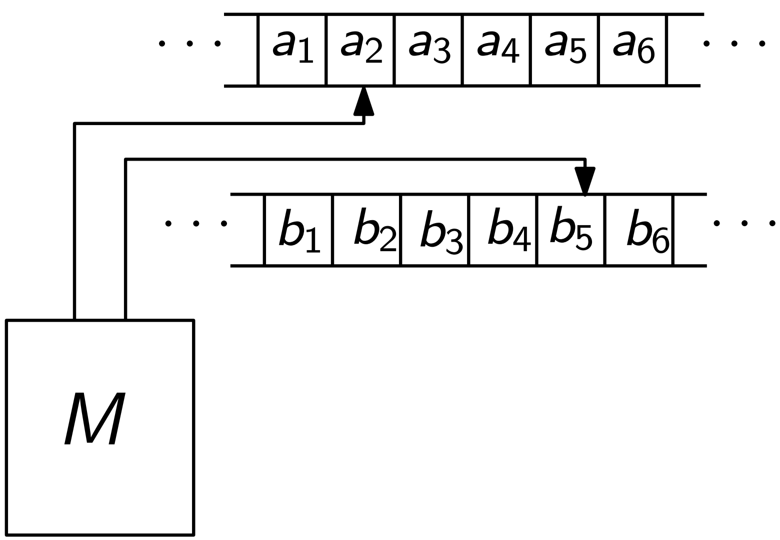


Multi Tape Turing Machine

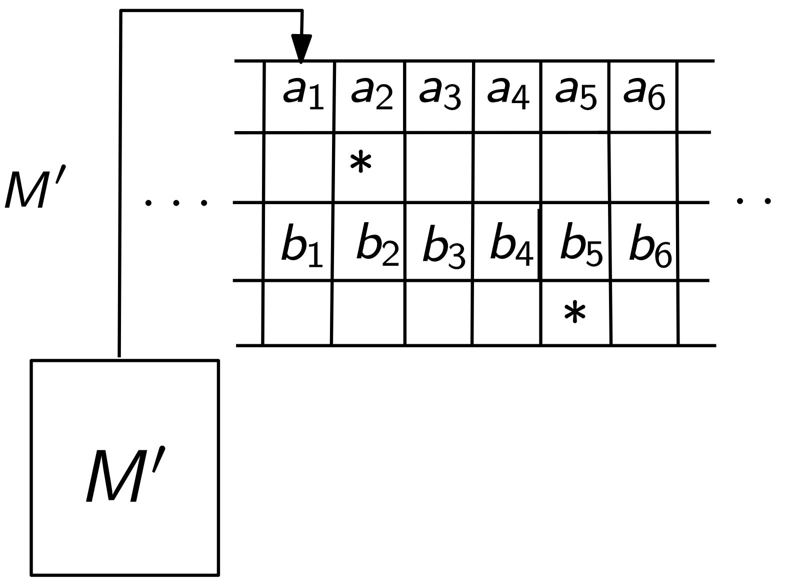
Remark
The head positions for each tape do not be the same.

Theorem
Any k-tape turing machine can be simulated by a single tape turing machine.

Sketch of Proof



M can be simulated by M'



Exercise
2-tape Turing machine for $a^n b^n c^n$ that requires only a single pass.

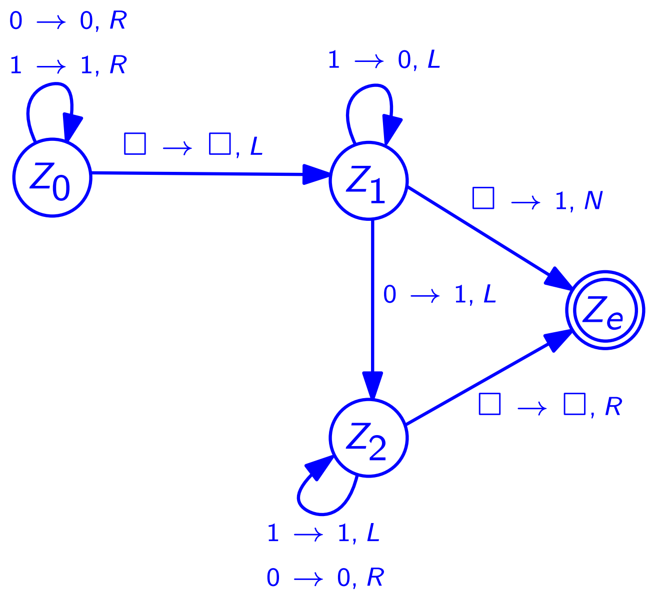
Turing Machine for Calculation of Functions

Example Input: Binary number $x = \{0, 1\}^*$
Output: $x + 1$

Turing Machine for Calculation of Functions

Example Input: Binary number $x = \{0, 1\}^*$
Output: $x + 1$

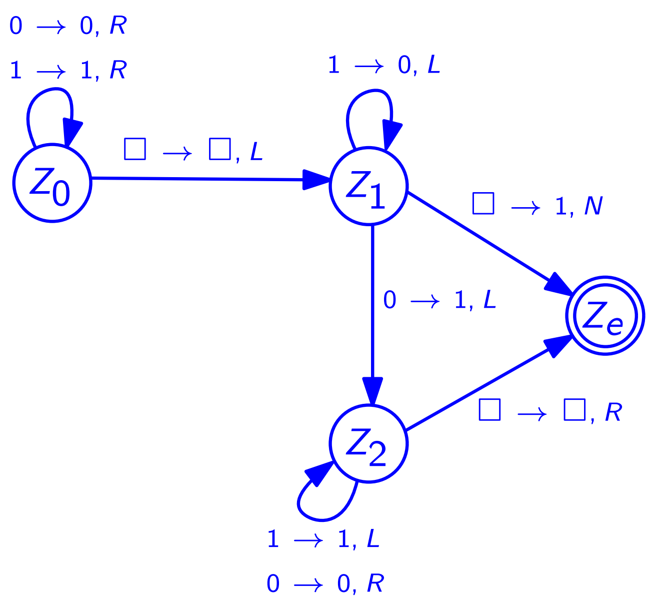
Turing Machine $M = (\{z_0, z_1, z_2, z_e\}, \{0, 1\}, \{0, 1, \square\}, \delta, z_0, \square, \{z_e\})$



Turing Machine for Calculation of Functions

Example Input: Binary number $x = \{0, 1\}^*$
Output: $x + 1$

Turing Machine $M = (\{z_0, z_1, z_2, z_e\}, \{0, 1\}, \{0, 1, \square\}, \delta, z_0, \square, \{z_e\})$



(i) $x = 101$

head position

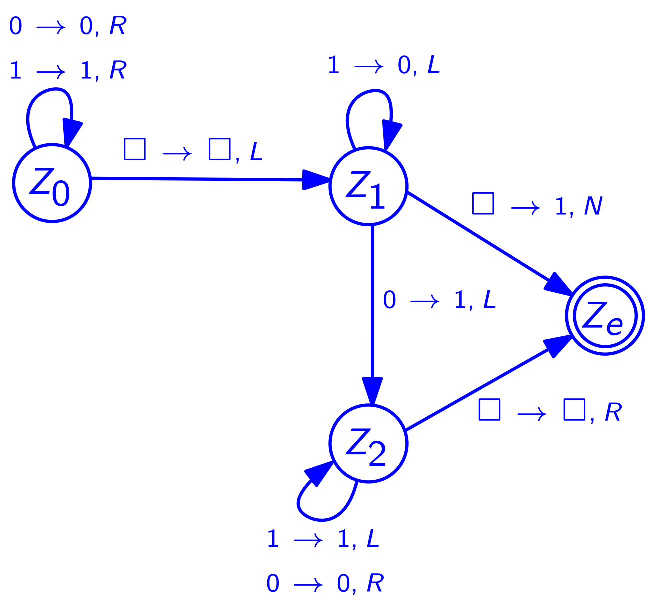
State	Tape
z_0	$\overline{1}01$
z_0	$1\overline{0}1$
z_0	$10\overline{1}$
z_0	$101\overline{\square}$
z_1	$10\overline{1}$
z_1	$1\overline{0}0$
z_2	$\overline{1}10$
z_2	$\overline{\square}110$
z_e	$\overline{1}01$

$z_0 101 \vdash 1 z_0 01 \vdash 10 z_0 1 \vdash$
$101 z_0 \square \vdash 10 z_1 1 \vdash 1 z_1 00 \vdash$
$z_2 110 \vdash z_2 \square 110 \vdash z_e 110$

Turing Machine for Calculation of Functions

Example Input: Binary number $x = \{0,1\}^*$
Output: $x + 1$

Turing Machine $M = (\{z_0, z_1, z_2, z_e\}, \{0,1\}, \{0,1,\square\}, \delta, z_0, \square, \{z_e\})$



(i) $x = 101$

head position

State	Tape
z_0	$\overline{1}01$
z_0	$1\overline{0}1$
z_0	$10\overline{1}$
z_0	$101\overline{\square}$
z_1	$10\overline{1}$
z_1	$1\overline{0}0$
z_2	$\overline{1}10$
z_2	$\overline{\square}110$
z_e	$\overline{1}01$

$z_0101 \vdash 1z_001 \vdash 10z_01 \vdash$
$101z_0\square \vdash 10z_11 \vdash 1z_100 \vdash$
$z_2110 \vdash z_2\square110 \vdash z_e110$

(ii) $x = 11$

State	Tape
z_0	$\overline{1}1$
z_0	$1\overline{1}$
z_0	$11\overline{\square}$
z_1	$1\overline{1}$
z_1	$\overline{1}0$
z_1	$\overline{\square}00$
z_e	$\overline{1}00$

$z_011 \vdash 1z_01 \vdash 11z_0\square \vdash$
$1z_11 \vdash z_110 \vdash z_1\square00 \vdash$
z_e100

Universal TM

The program of a turing machine is realized *in hardware*. This is like building the circuits of a CPU for a specific assembler program

$$\text{1 problem} \quad \leftrightarrow \quad \text{1 turing machine}$$

Universal TM

The program of a turing machine is realized *in hardware*. This is like building the circuits of a CPU for a specific assembler program

$$1 \text{ problem} \quad \leftrightarrow \quad 1 \text{ turing machine}$$

Turing already saw this restriction and developed a universal turing machine.

Universal TM

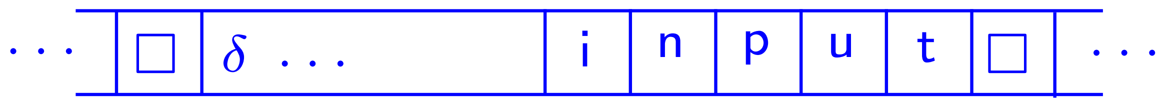
The program of a turing machine is realized *in hardware*. This is like building the circuits of a CPU for a specific assembler program

$$1 \text{ problem} \quad \leftrightarrow \quad 1 \text{ turing machine}$$

Turing already saw this restriction and developed a universal turing machine.

Idea

Put the δ description on the tape followed by the input



Universal TM

The universal turing machine simulates the turing machine that lays on the tape. This simulation is pretty inefficient.

Universal TM

The universal turing machine simulates the turing machine that lays on the tape. This simulation is pretty inefficient.

The turing machine can be described in different ways, either one puts the full tuple onto the tape, or one summarizes the tuple into an integer, the so-called Gödel number. Here, a bijection between $\mathbb{N}$ and the tuple description is exploited.

Universal TM

The universal turing machine simulates the turing machine that lays on the tape. This simulation is pretty inefficient.

The turing machine can be described in different ways, either one puts the full tuple onto the tape, or one summarizes the tuple into an integer, the so-called Gödel number. Here, a bijection between $\mathbb{N}$ and the tuple description is exploited.

The UTM can be used to prove the undecidability of the halting problem.

$$H = \{ < M > x \mid M \text{ is a deterministic turing machine that terminates on input } x \}$$

Example Type 1 Language that is not Type 2

The following language is a type 1 language, which is not type 2

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

Example Type 1 Language that is not Type 2

The following language is a type 1 language, which is not type 2

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

Grammar:

$$S \xrightarrow{(1)} LaR$$

$$L \xrightarrow{(2)/(3)} LD \mid \varepsilon$$

$$Da \xrightarrow{(4)} aaD$$

$$DR \xrightarrow{(5)} R$$

$$R \xrightarrow{(6)} \varepsilon$$

Example Type 1 Language that is not Type 2

The following language is a type 1 language, which is not type 2

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

Grammar:

$$S \overset{(1)}{\rightarrow} LaR$$

$$L \overset{(2)/(3)}{\rightarrow} LD \mid \varepsilon$$

$$Da \overset{(4)}{\rightarrow} aaD$$

$$DR \overset{(5)}{\rightarrow} R$$

$$R \overset{(6)}{\rightarrow} \varepsilon$$

Example derivation:

$$S \overset{(1)}{\Rightarrow} LaR \overset{(2)}{\Rightarrow}{}^3 LDDDaR \overset{(3)}{\Rightarrow} DDDaR$$

$$\overset{(4)}{\Rightarrow} DDaaDR \overset{(4)}{\Rightarrow} DaaDaDR \overset{(4)}{\Rightarrow} aaDaDaDR$$

$$\overset{(4)}{\Rightarrow} aaaaDDaDR \overset{(4)}{\Rightarrow}{}^* aaaaaaaaDDDR$$

$$\overset{(5)}{\Rightarrow} aaaaaaaaDDR \overset{(5)}{\Rightarrow}{}^* aaaaaaaaDR$$

$$\overset{(5)}{\Rightarrow} aaaaaaaaR \overset{(6)}{\Rightarrow}{}^* aaaaaaaa = a^8 = a^{2^3}$$