# Theory of Computer Science

## B11. Turing Machines II

Gabriele Röger

University of Basel

April 13, 2022

# Theory of Computer Science
April 13, 2022 — B11. Turing Machines II

B11.1 Variants of Turing Machines

B11.2 Multitape Turing Machines

B11.3 Nondeterministic Turing Machines

B11.4 Summary

# B11.1 Variants of Turing Machines

# Reminder: Deterministic Turing Machine

**Definition (Deterministic Turing Machine)**

A (deterministic) Turing machine (DTM) is given by a 7-tuple $M = \langle Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}} \rangle$, where $Q, \Sigma, \Gamma$ are finite and

- $Q$ is the set of states,
- $\Sigma$ is the input alphabet, not containing the blank symbol $\square$,
- $\Gamma$ is the tape alphabet, where $\square \in \Gamma$ and $\Sigma \subseteq \Gamma$,
- $\delta : (Q \setminus \{q_{\text{accept}}, q_{\text{reject}}\}) \times \Gamma \to Q \times \Gamma \times \{\text{L}, \text{R}\}$ is the transition function,
- $q_0 \in Q$ is the start state,
- $q_{\text{accept}} \in Q$ is the accept state,
- $q_{\text{reject}} \in Q$ is the reject state, where $q_{\text{accept}} \neq q_{\text{reject}}$.

Deterministic TM with a single tape that is infinite at one side.

# Turing Machines with Neutral Move

- A DTM only allows head movements to the left or right.
- A DTM with neutral move is a variant with transition function $\delta : (Q \setminus \{q_{\text{accept}}, q_{\text{reject}}\}) \times \Gamma \to Q \times \Gamma \times \{\text{L}, \text{R}, \text{N}\})$, where N means that the R/W-head stays put.

Can this variant recognize languages that standard DTMs cannot?

# Turing Machines with Neutral Move are Equally Powerful

- Obviously, every DTM can be seen as a DTM with neutral move, so the variant is clearly not less powerful.
- Vice versa, every language recognized by a DTM $M$ with neutral moves can be recognized by a (standard) DTM $M'$:
  - For every state $q$ of $M$, $M'$ has an additional state $q'$.
  - If $M$ writes $c$, switches to state $q$ and stays put, $M'$ writes $c$, switches to state $q'$ and moves right. Whenever $M'$ is in one of the primed states, it does not change the tape, switches to the unprimed state $q$ and moves left.
  - For every transition of $M$ with neutral move, $M'$ takes two transitions with the same result.

To show that two models are equivalent,
we can show that we can simulate one by the other.

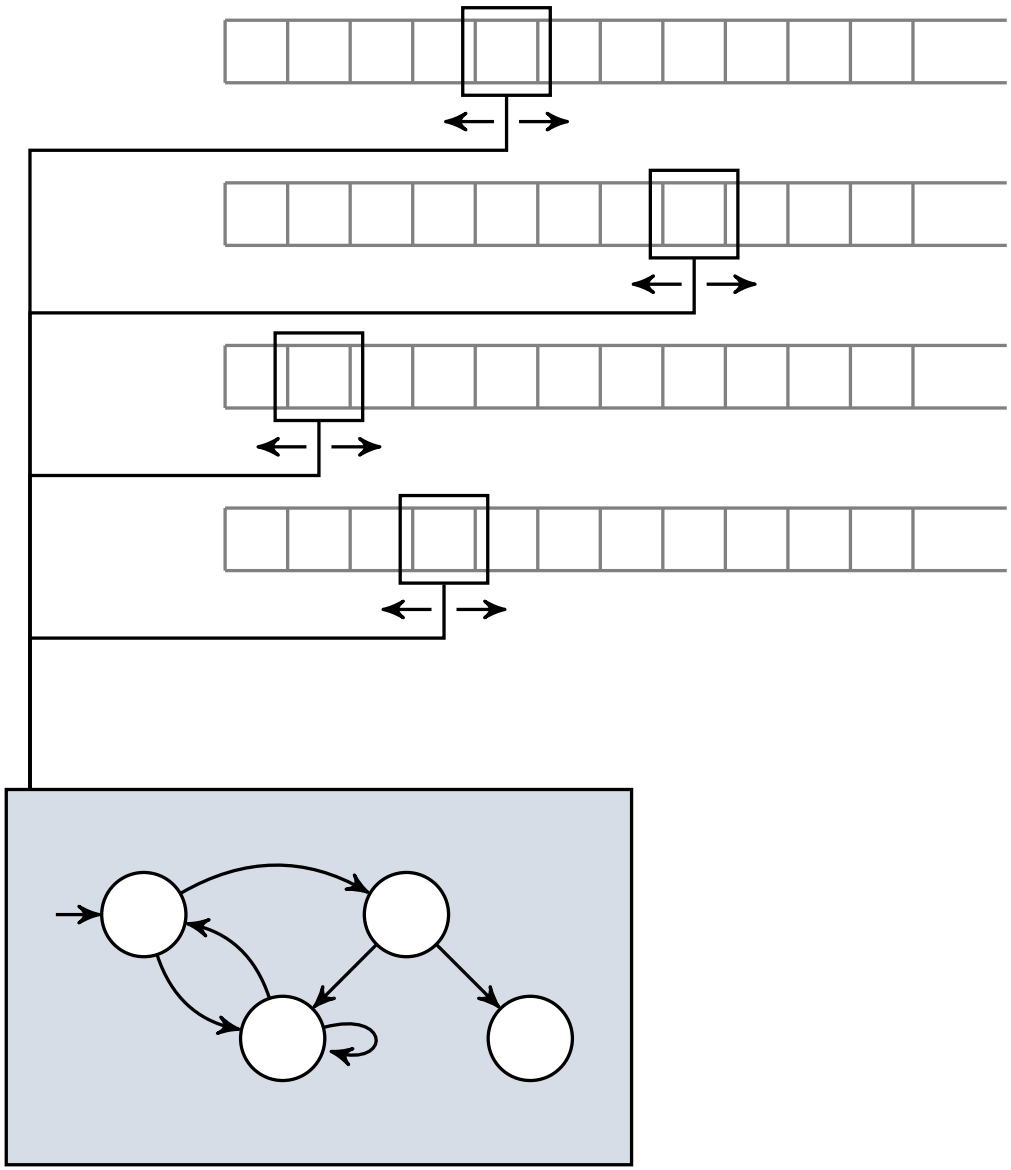
# B11.2 Multitape Turing Machines

# Multitape Turing Machines

A multitape TM is like a DTM (with neutral movement) but with several tapes.

- every tape has its own read-write head,
- the input appears on tape 1,
- all other tapes are initially filled with blank symbols,
- the transition function considers all $k$ tapes simultaneously $\delta : (Q \setminus \{q_{\text{accept}}, q_{\text{reject}}\}) \times \Gamma^k \to Q \times \Gamma^k \times \{\text{L}, \text{R}, \text{N}\}^k$

# Multitape Turing Machines: Conceptually



# Multitape Turing Machine: Transitions

$$\delta(q, a_1, \ldots, a_k) = (q', a'_1, \ldots, a'_k, D_1, \ldots, D_k)$$

- If the TM is in state $q$,
- and on each tape $i$ the head reads symbol $a_i$, then
- the TM switches to state $q'$,
- replaces on each tape $i$ the symbol $a_i$ with $a'_i$, and
- moves the head on each tape $i$ in direction $D_i$ ($D_i \in \{\mathrm{L}, \mathrm{R}, \mathrm{N}\}$)

# Multitape TMs No More Powerful Than Single-Tape TMs

**Theorem**

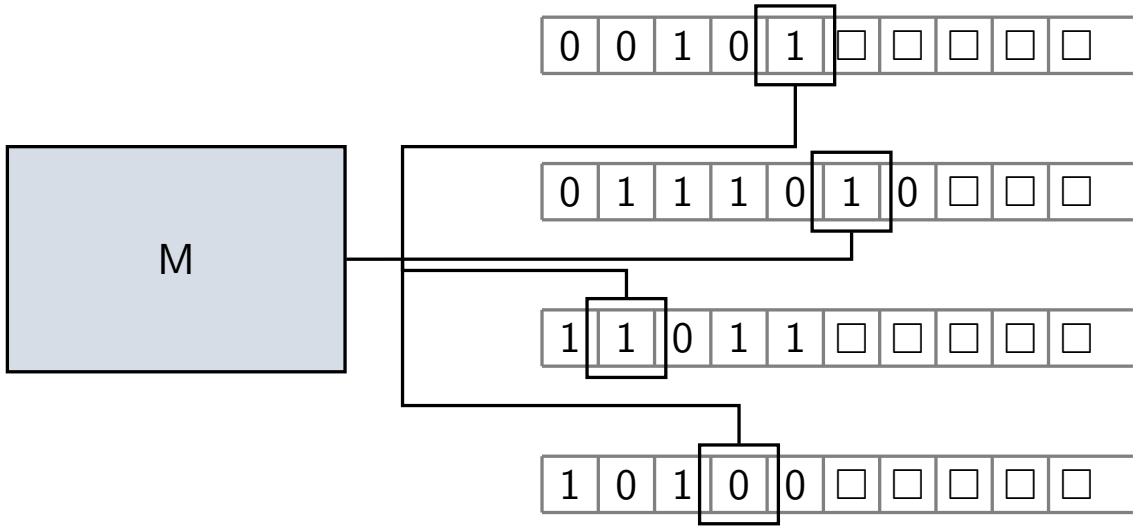
*Every multitape TM has an equivalent single-tape TM.*

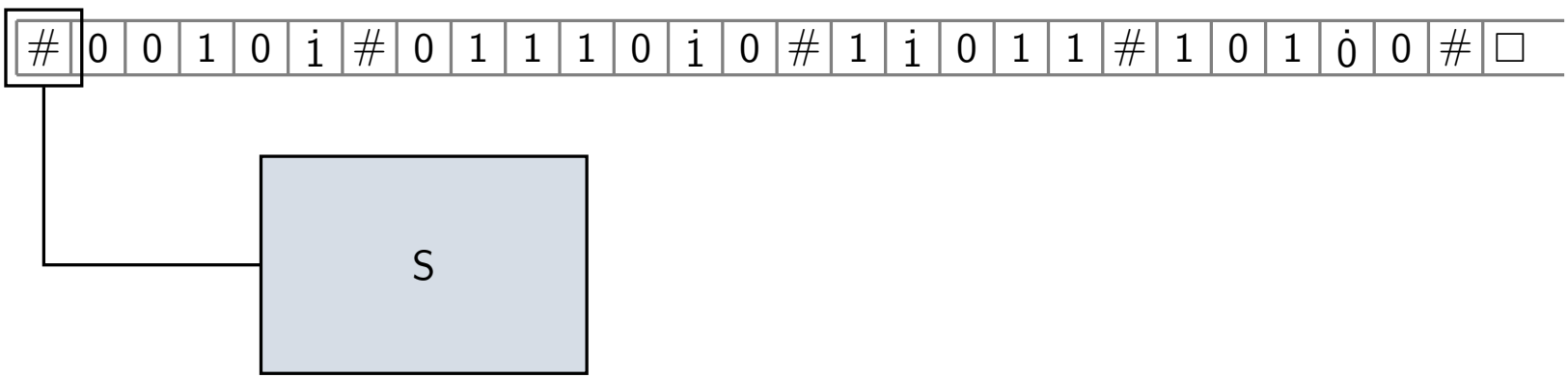
**Proof.**

Let $M$ be a TM with $k$ tapes. We construct a single-tape DTM $S$ that recognizes the same language.

$S$ stores the information of the multiple tapes on its tape, separating the contents of different tapes with a new symbol $\#$.

To keep track of the positions of the heads of $M$, TM $S$ has for each tape symbol $x$ of $M$ a new tape symbol $\dot{x}$ to marks the corresponding positions. ...


0 0 1 0 1 □ □ □ □ □
0 1 1 1 0 1 0 □ □ □
M
1 1 0 1 1 □ □ □ □ □
1 0 1 0 0 □ □ □ □ □


# 0 0 1 0 i # 0 1 1 1 0 i 0 # 1 i 0 1 1 # 1 0 1 o 0 # □
S

# Multitape TMs No More Powerful Than Single-Tape TMs

**Theorem**

*Every multitape TM has an equivalent single-tape TM.*

**Proof (continued).**

On input $w = w_1 \dots w_n$

1. Initialize the tape of $S$ to $\#\dot{w}_1 w_2 \dots w_n \# \dot{\sqcup} \# \dot{\sqcup} \# \dots \#$
2. To simulate a transition of $M$, TM $S$ scans from the leftmost $\#$ to the $k+1$st $\#$ to determine what symbols are under the virtual heads. In a second pass, $S$ updates the tape according to the transition of $M$.
3. If it moves a virtual head on the $\#$ marking the right end of its tape, it frees this position by shifting the tape content from this position on one position to the right and adds a blank into the "new" position.

□

# Details?

Consider the situation where $S$ has done its first pass (back at the left-most position) and has determined that $M$ would take transition

$$\delta(q, x_1, \ldots, x_k) = (q, y_1, \ldots, y_k, D_1, \ldots, D_k).$$

How can you "implement" the second pass of $S$ that updates the tape accordingly? You may assume that it will never move a virtual head from the already represented part of its tape.

First pass and shifting the tape content $\rightsquigarrow$ exercises

# Multitape TMs Equally Powerful as Single-Tape TMs

**Theorem**

*A language is Turing-recognizable iff some multitape Turing machine recognizes it.*

**Proof.**

"$\Rightarrow$": A DTM is a special case of a multitape TM.

"$\Leftarrow$": Previous theorem $\square$

# B11.3 Nondeterministic Turing Machines

# Nondeterministic Turing Machines

A nondeterministic Turing machine (NTM) relates to a DTM as a NFA relates to a DFA.

- The transition function can specify several possibilities: $\delta : (Q \setminus \{q_{\text{accept}}, q_{\text{reject}}\}) \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{\mathrm{L}, \mathrm{R}, \mathrm{N}\})$
- For a given input, we can consider the computation tree whose branches correspond to following different possibilities.
- If some branch leads to the accept state, the NTM accepts the input word.

# Nondeterministic TMs no More Powerful than DTMs

**Theorem**

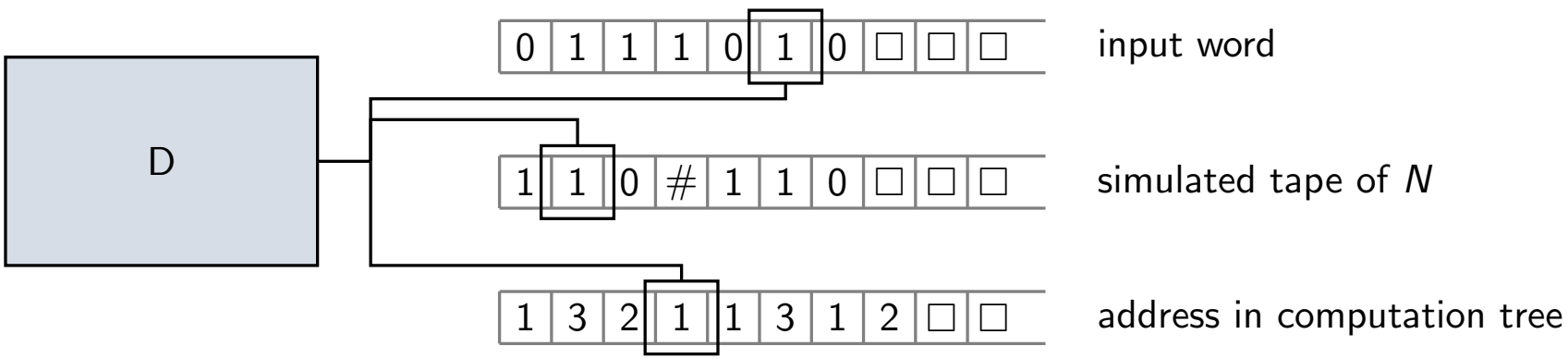
*Every nondeterministic Turing machine has an equivalent deterministic Turing machine.*

# Nondeterministic TMs no More Powerful than DTMs

**Proof.**

Let $N$ be a NTM. We describe a deterministic 3-tape TM $D$ that searches the computation tree of $N$ on input $w$ for an accepting configuration with a breadth-first search. The theorem follows from the equivalence of multitape TMs and DTMs.

The first tape always contains $w$, the second tape corresponds to the content of $N$'s tape on some branch of the computation tree and the third tape tracks the position in $N$'s computation tree. ...



# Nondeterministic TMs no More Powerful than DTMs

What is the "address in the computation tree"?

- Let $b$ be the maximal number of children of a node in the CT (= size of largest set of possibilities in the transition function)
- The address is a string over $\{1, 2, \ldots, b\}$.
- For example, address 312 refers to the node in the CT reached by starting from the root node (= initial configuration)
  - going to the third child node, then
  - going to the first child of the resulting node, and then
  - going to the second child of this child node.
- If a node does not have that many children, the address is invalid.

# Nondeterministic TMs no More Powerful than DTMs

## Proof (continued).

$D$ works on input $w$ as follows:

1. Initially, tape 1 contains $w$, tape 2 and 3 contain only blanks.
2. Copy tape 1 to tape 2.
3. Simulate $N$ on input $w$ following one branch of the computation tree. Before each transition of $N$, determine which choice to make from the next symbol on tape 3. If there is no number left on tape 3, if the choice is invalid or a rejecting configuration is encountered, go to step 4. If an accepting configuration is encountered, accept.
4. Replace the string on tape 3 with the next string (first short strings then longer ones, strings of same length in lexicographic order) and go to step 2.

□

# Nondeterministic TMs no More Powerful than DTMs

Wouldn't it be easier to do a depth-first search for an accepting configuration in the computation tree? Why don't we do this and e.g. first entirely explore the first branch of the tree?

# NTMs and DTMs are Equally Powerful

**Theorem**

*A language is Turing-recognizable iff some nondeterministic Turing machine recognizes it.*

**Proof.**

"$\Rightarrow$": Any DTM can be cast as a NTM.

"$\Leftarrow$": Previous theorem $\square$

# B11.4 Summary

# Summary

We have seen several variants of Turing machines:

- Deterministic TM with head movements left or right
- Deterministic TM with head movements left, right or neutral
- Multitape Turing machines
- Nondeterministic Turing machines

All variants recognize the same languages.