Foundations of Computing

Lecture 13

Arkady Yerukhimovich

February 27, 2025

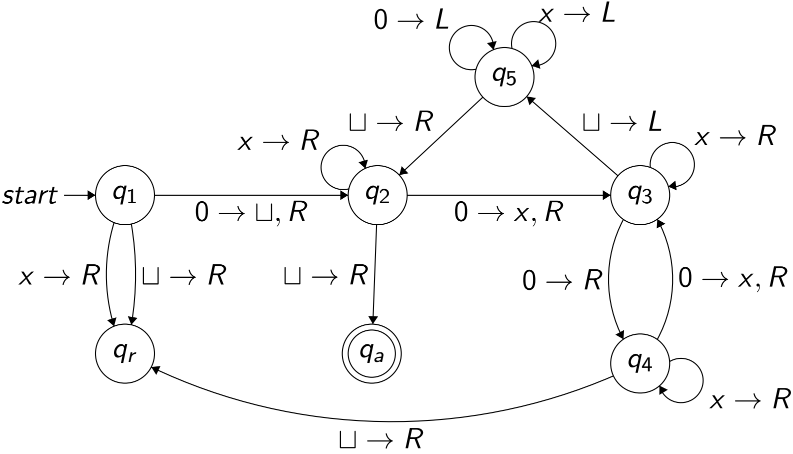
Outline

1. Lecture 12 Review
2. Some More Turing Machines
3. Church-Turing Thesis
4. Turing Machine Variants

Lecture 12 Review

- Turing Machines
  - Definition
  - Examples

Running M on $w = 00$



Outline

Specification of a Turing Machine

There are several levels of detail for specifying a TM

1. Full specification
   - Give full detail of transition function δ
   - This is very tedious

Specification of a Turing Machine

There are several levels of detail for specifying a TM

1. Full specification
   - Give full detail of transition function δ
   - This is very tedious
2. Turing Machine Algorithm specification
   - Explain algorithmically what happens on the tape
   - For example, scan the tape until you find a $\#$, zig-zag on the tape, etc.
   - Don't bother specifying a DFA for the control state

Specification of a Turing Machine

There are several levels of detail for specifying a TM

1. Full specification
   - Give full detail of transition function δ
   - This is very tedious
2. Turing Machine Algorithm specification
   - Explain algorithmically what happens on the tape
   - For example, scan the tape until you find a $\#$, zig-zag on the tape, etc.
   - Don't bother specifying a DFA for the control state
3. Algorithm specification
   - Give algorithm in pseudocode
   - Don't explicitly spell out what happens on the tape

Example 1: $L = \{a^i b^j c^k \mid i \times j = k \text{ and } i, j, k \geq 1\}$

Machine M deciding L

On input string w:

Example 1: $L = \{a^i b^j c^k \mid i \times j = k \text{ and } i, j, k \geq 1\}$

Machine M deciding L

On input string w:

1. Check format of the input – scan input left to right and check that it is a member of $a^* b^* c^*$, reject if it isn't

Example 1: $L = \{a^i b^j c^k \mid i \times j = k \text{ and } i, j, k \geq 1\}$

Machine M deciding L

On input string w:

1. Check format of the input – scan input left to right and check that it is a member of $a^* b^* c^*$, reject if it isn't
2. Return the head back to the beginning of the input

Example 1: $L = \{a^i b^j c^k \mid i \times j = k \text{ and } i, j, k \geq 1\}$

Machine M deciding L

On input string w:

1. Check format of the input – scan input left to right and check that it is a member of $a^*b^*c^*$, reject if it isn't
2. Return the head back to the beginning of the input

Intuition:

- Want to check if $k = i \times j$. Equivalently, $k = \overbrace{j + j + \cdots + j}^{i \text{ times}}$

Example 1: $L = \{a^i b^j c^k \mid i \times j = k \text{ and } i, j, k \geq 1\}$

Machine M deciding L

On input string w:

1. Check format of the input – scan input left to right and check that it is a member of $a^* b^* c^*$, reject if it isn't
2. Return the head back to the beginning of the input

Intuition:

- Want to check if $k = i \times j$. Equivalently, $k = \overbrace{j + j + \cdots + j}^{i \text{ times}}$
- For every a, remove j c's

Example 1: $L = \{a^i b^j c^k \mid i \times j = k \text{ and } i, j, k \geq 1\}$

Machine M deciding L

On input string w:

1. Check format of the input – scan input left to right and check that it is a member of $a^* b^* c^*$, reject if it isn't
2. Return the head back to the beginning of the input

Intuition:

- Want to check if $k = i \times j$. Equivalently, $k = \overbrace{j + j + \cdots + j}^{i \text{ times}}$
- For every a, remove j c's
- If there are no c's left when done then accept

Example 1: $L = \{a^i b^j c^k \mid i \times j = k \text{ and } i, j, k \geq 1\}$

Machine M deciding L

On input string w:

1. Check format of the input – scan input left to right and check that it is a member of $a^* b^* c^*$, reject if it isn't
2. Return the head back to the beginning of the input

Intuition:

- Want to check if $k = i \times j$. Equivalently, $k = \overbrace{j + j + \cdots + j}^{i \text{ times}}$
- For every a, remove j c's
- If there are no c's left when done then accept

3. Cross off an a and scan to the right until you find a b. Zig zag between b's and c's crossing off one of each until all b's are gone.

Example 1: $L = \{a^i b^j c^k \mid i \times j = k \text{ and } i, j, k \geq 1\}$

Machine M deciding L

On input string w:

1. Check format of the input – scan input left to right and check that it is a member of $a^*b^*c^*$, reject if it isn't
2. Return the head back to the beginning of the input

Intuition:

- Want to check if $k = i \times j$. Equivalently, $k = \overbrace{j + j + \cdots + j}^{i \text{ times}}$
- For every a, remove j c's
- If there are no c's left when done then accept

3. Cross off an a and scan to the right until you find a b. Zig zag between b's and c's crossing off one of each until all b's are gone.
4. Restore all the b's, find next uncrossed off a and repeat Step 3.

Example 1: $L = \{a^i b^j c^k \mid i \times j = k \text{ and } i, j, k \geq 1\}$

Machine M deciding L

On input string w:

1. Check format of the input – scan input left to right and check that it is a member of $a^*b^*c^*$, reject if it isn't
2. Return the head back to the beginning of the input

Intuition:

- Want to check if $k = i \times j$. Equivalently, $k = \overbrace{j + j + \cdots + j}^{i \text{ times}}$
- For every a, remove j c's
- If there are no c's left when done then accept

3. Cross off an a and scan to the right until you find a b. Zig zag between b's and c's crossing off one of each until all b's are gone.
4. Restore all the b's, find next uncrossed off a and repeat Step 3.
5. If all a's are crossed off, check if all c's are crossed off. Accept if yes, reject if no.

Example 2 – Build a TM deciding L Below

$$L = \{\#x_1\#x_2\#\cdots\#x_\ell \mid \text{ each } x_i \in \{0,1\}^* \text{ and } x_i \neq x_j \text{ for all } i \neq j\}$$

Example 2 – Build a TM deciding L Below

$$L = \{\#x_1\#x_2\#\cdots\#x_\ell \mid \text{ each } x_i \in \{0,1\}^* \text{ and } x_i \neq x_j \text{ for all } i \neq j\}$$

M deciding L

On input sting w:

1. Look at first symbol, If $\sqcup$, accept. If $\#$ goto step 2. Else, reject

Example 2 – Build a TM deciding L Below

$$L = \{\#x_1\#x_2\#\cdots\#x_\ell \mid \text{ each } x_i \in \{0,1\}^* \text{ and } x_i \neq x_j \text{ for all } i \neq j\}$$

M deciding L

On input sting w:

1. Look at first symbol, If $\sqcup$, accept. If $\#$ goto step 2. Else, reject
2. Place mark on top of first $\#$ and scan to next $\#$ and mark it. If no second $\#$ found, accept.

Example 2 – Build a TM deciding L Below

$$L = \{\#x_1\#x_2\#\cdots\#x_\ell \mid \text{ each } x_i \in \{0,1\}^* \text{ and } x_i \neq x_j \text{ for all } i \neq j\}$$

M deciding L

On input sting w:

1. Look at first symbol, If $\sqcup$, accept. If $\#$ goto step 2. Else, reject
2. Place mark on top of first $\#$ and scan to next $\#$ and mark it. If no second $\#$ found, accept.
3. By zig-zagging compare the two strings to the right of marked $\#$'s. If they are equal, reject

Example 2 – Build a TM deciding L Below

$$L = \{\#x_1\#x_2\#\cdots\#x_\ell \mid \text{ each } x_i \in \{0,1\}^* \text{ and } x_i \neq x_j \text{ for all } i \neq j\}$$

M deciding L

On input sting w:

1. Look at first symbol, If $\sqcup$, accept. If $\#$ goto step 2. Else, reject
2. Place mark on top of first $\#$ and scan to next $\#$ and mark it. If no second $\#$ found, accept.
3. By zig-zagging compare the two strings to the right of marked $\#$'s. If they are equal, reject
4. Move right mark to next $\#$, if there isn't one move left mark one $\#$ to the right and right mark to $\#$ after that (if there isn't one, accept)

Example 2 – Build a TM deciding L Below

$$L = \{\#x_1\#x_2\#\cdots\#x_\ell \mid \text{ each } x_i \in \{0,1\}^* \text{ and } x_i \neq x_j \text{ for all } i \neq j\}$$

M deciding L

On input sting w:

1. Look at first symbol, If $\sqcup$, accept. If $\#$ goto step 2. Else, reject
2. Place mark on top of first $\#$ and scan to next $\#$ and mark it. If no second $\#$ found, accept.
3. By zig-zagging compare the two strings to the right of marked $\#$'s. If they are equal, reject
4. Move right mark to next $\#$, if there isn't one move left mark one $\#$ to the right and right mark to $\#$ after that (if there isn't one, accept)
5. Goto step 3

Outline

Church-Turing Thesis (1936)

Anything that can be computed by an algorithm can be computed by a Turing Machine

Turing Machines and Algorithms

Church-Turing Thesis (1936)

Anything that can be computed by an algorithm can be computed by a Turing Machine

Observations:

- While unproven, all modern computers satisfy Church-Turing thesis

Church-Turing Thesis (1936)

Anything that can be computed by an algorithm can be computed by a Turing Machine

Observations:

- While unproven, all modern computers satisfy Church-Turing thesis
- To prove that some problem cannot be solved by an algorithm, enough to reason about Turing Machines

Turing Machines and Algorithms

Church-Turing Thesis (1936)

Anything that can be computed by an algorithm can be computed by a Turing Machine

Observations:

- While unproven, all modern computers satisfy Church-Turing thesis
- To prove that some problem cannot be solved by an algorithm, enough to reason about Turing Machines
- This means that Turing Machines give an abstraction to capture “feasible computation”

Outline

A Universal Turing Machine

Question

A TM that can run any other TM

$$\hat{M}(M, x) = M(x)$$

A Universal Turing Machine

Question

A TM that can run any other TM

$$\hat{M}(M, x) = M(x)$$

Observations:

- $\hat{M}$ can take any TM M as an input

A Universal Turing Machine

Question

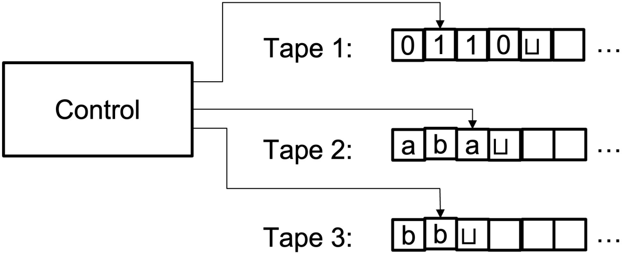
A TM that can run any other TM

$$\hat{M}(M, x) = M(x)$$

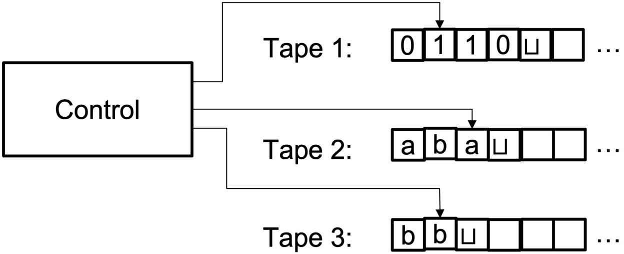
Observations:

- $\hat{M}$ can take any TM M as an input
- In particular $|\hat{M}|$ can be much less than $|M|$

Multi-Tape Turing Machines



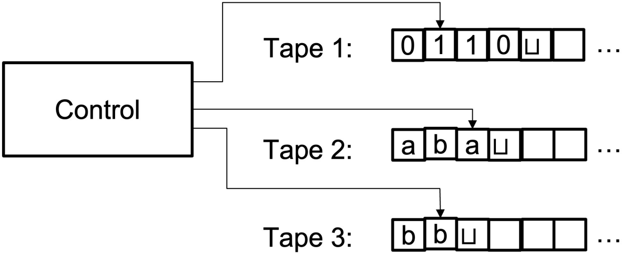
Multi-Tape Turing Machines



In each step:

- M can read each tape
- M can write to each tape
- M can move each tape head Left or Right

Multi-Tape Turing Machines



In each step:

- M can read each tape
- M can write to each tape
- M can move each tape head Left or Right

Formally, for k tapes

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

Multi-Tape Turing Machines

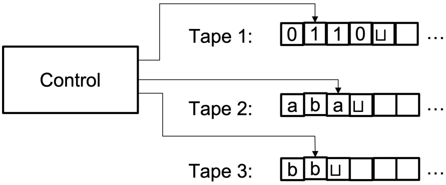
Theorem

Every multi-tape TM has an equivalent single-tape TM

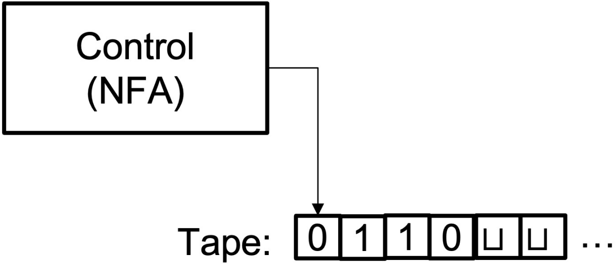
Multi-Tape Turing Machines

Theorem

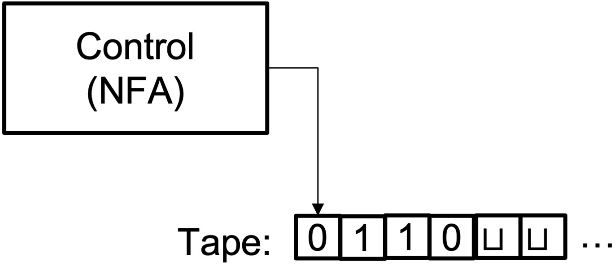
Every multi-tape TM has an equivalent single-tape TM



Nondeterministic Turing Machines



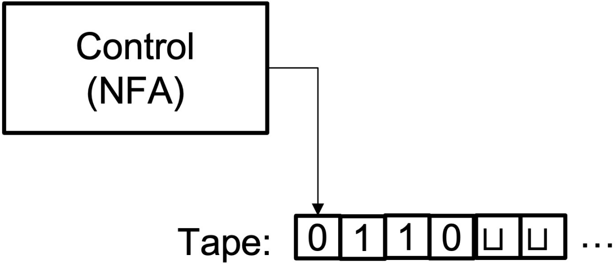
Nondeterministic Turing Machines



Formally,

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

Nondeterministic Turing Machines



Formally,

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

Intuition:

- The control unit is non-deterministic - many transitions possible on each input
- Execution corresponds to a tree of possible executions
- Accept if any of possible execution leads to accept

Nondeterministic Turing Machine

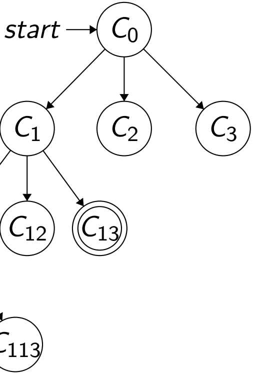
Theorem

Every nondeterministic TM has an equivalent deterministic TM.

Nondeterministic Turing Machine

Theorem

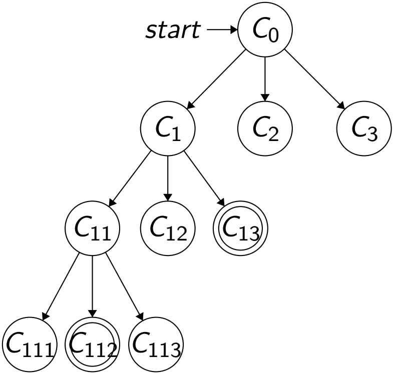
Every nondeterministic TM has an equivalent deterministic TM.



Nondeterministic Turing Machine

Theorem

Every nondeterministic TM has an equivalent deterministic TM.

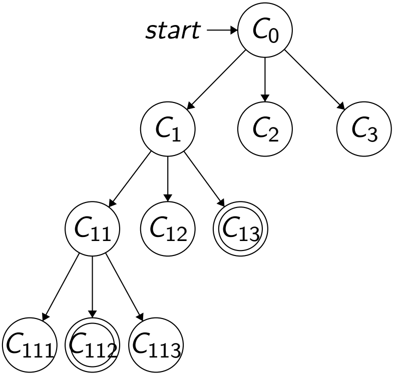


- Recall that an execution of a DTM is a sequence of configurations

Nondeterministic Turing Machine

Theorem

Every nondeterministic TM has an equivalent deterministic TM.

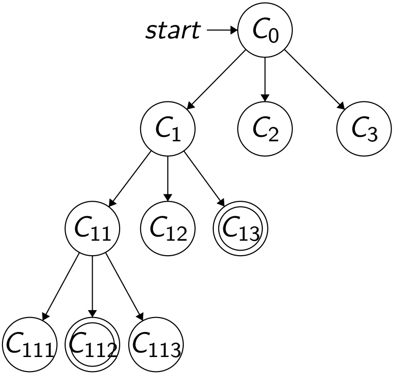


- Recall that an execution of a DTM is a sequence of configurations
- Execution of an NTM is a tree of configurations (branches correspond to non-deterministic choices)

Nondeterministic Turing Machine

Theorem

Every nondeterministic TM has an equivalent deterministic TM.

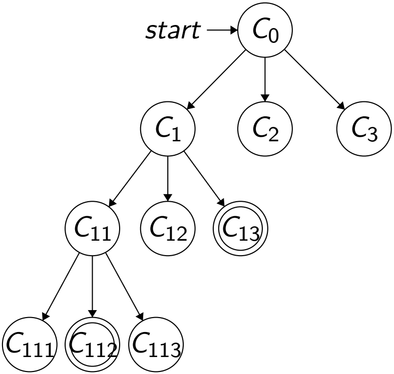


- Recall that an execution of a DTM is a sequence of configurations
- Execution of an NTM is a tree of configurations (branches correspond to non-deterministic choices)
- If any node in the tree is an accept node, the NTM accepts

Nondeterministic Turing Machine

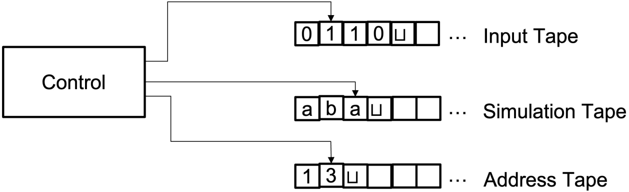
Theorem

Every nondeterministic TM has an equivalent deterministic TM.



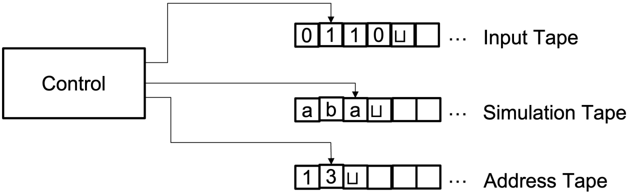
- Recall that an execution of a DTM is a sequence of configurations
- Execution of an NTM is a tree of configurations (branches correspond to non-deterministic choices)
- If any node in the tree is an accept node, the NTM accepts
- To simulate an NTM by a DTM, need to try all configurations in the tree to see if we find an accepting one

Nondeterministic Turing Machines



To simulate an NTM N by a DTM D, we use three tapes:

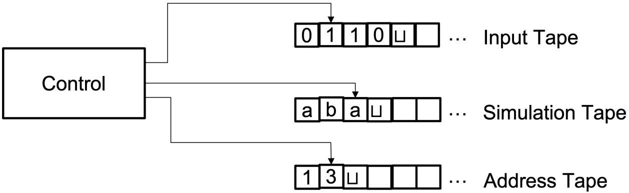
Nondeterministic Turing Machines



To simulate an NTM N by a DTM D, we use three tapes:

1. Input tape – stores the input and doesn't change

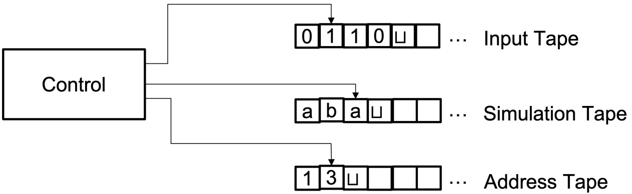
Nondeterministic Turing Machines



To simulate an NTM N by a DTM D, we use three tapes:

1. Input tape – stores the input and doesn't change
2. Simulation tape – work tape for the NTM on one branch of nondeterminism

Nondeterministic Turing Machines



To simulate an NTM N by a DTM D, we use three tapes:

1. Input tape – stores the input and doesn't change
2. Simulation tape – work tape for the NTM on one branch of nondeterminism
3. Address tape – use to store which nondeterministic branch you are on

Nondeterministic Turing Machines

Simulating an NTM N

1. Start with input w on tape 1, and tapes 2,3 empty

Nondeterministic Turing Machines

Simulating an NTM N

1. Start with input w on tape 1, and tapes 2,3 empty
2. Copy w to tape 2

Nondeterministic Turing Machines

Simulating an NTM N

1. Start with input w on tape 1, and tapes 2,3 empty
2. Copy w to tape 2
3. Use tape 2 to simulate a run of N. Whenever it needs to make a non-deterministic choice, see next symbol on tape 3 for which branch to take. If no symbols left, go to step 4

Nondeterministic Turing Machines

Simulating an NTM N

1. Start with input w on tape 1, and tapes 2,3 empty
2. Copy w to tape 2
3. Use tape 2 to simulate a run of N. Whenever it needs to make a non-deterministic choice, see next symbol on tape 3 for which branch to take. If no symbols left, go to step 4
4. Replace string on tape 3 with the lexicographically next one (move onto next non-deterministic branch)

Nondeterministic Turing Machines

Simulating an NTM N

1. Start with input w on tape 1, and tapes 2,3 empty
2. Copy w to tape 2
3. Use tape 2 to simulate a run of N. Whenever it needs to make a non-deterministic choice, see next symbol on tape 3 for which branch to take. If no symbols left, go to step 4
4. Replace string on tape 3 with the lexicographically next one (move onto next non-deterministic branch)
5. If N ever enters an accept state, stop and accept

Nondeterministic Turing Machines

Simulating an NTM N

1. Start with input w on tape 1, and tapes 2,3 empty
2. Copy w to tape 2
3. Use tape 2 to simulate a run of N. Whenever it needs to make a non-deterministic choice, see next symbol on tape 3 for which branch to take. If no symbols left, go to step 4
4. Replace string on tape 3 with the lexicographically next one (move onto next non-deterministic branch)
5. If N ever enters an accept state, stop and accept

Important

Must traverse NTM tree in breadth-first, not depth-first order

Nondeterministic Turing Machines

Simulating an NTM N

1. Start with input w on tape 1, and tapes 2,3 empty
2. Copy w to tape 2
3. Use tape 2 to simulate a run of N. Whenever it needs to make a non-deterministic choice, see next symbol on tape 3 for which branch to take. If no symbols left, go to step 4
4. Replace string on tape 3 with the lexicographically next one (move onto next non-deterministic branch)
5. If N ever enters an accept state, stop and accept

Important

Must traverse NTM tree in breadth-first, not depth-first order

- Depth-first traversal may get stuck in an infinite loop, and not detect terminating branch

Next Week

- Decidable and undecidable languages
- I.e., are there things that no TM can compute?