Turing Machines and Decidability

Purushothama B R

Department of Computer Science and Engineering
National Institute of Technology Goa ,INDIA

December 7, 2020

I want to Begin With... J K Rowling's Quote

It is impossible to live without failing at something, unless you live so cautiously that you might has well not have lived at all, in which case you have failed by default.

TERMINOLOGY FOR DESCRIBING TURING MACHINES

TERMINOLOGY FOR DESCRIBING TURING MACHINES

- A turning point in the study of the theory of computation.

TERMINOLOGY FOR DESCRIBING TURING MACHINES

- A turning point in the study of the theory of computation.
- We continue to speak of Turing machines, but

TERMINOLOGY FOR DESCRIBING TURING MACHINES

- A turning point in the study of the theory of computation.
- We continue to speak of Turing machines, but
    - Our real focus from now on is on algorithms.

TERMINOLOGY FOR DESCRIBING TURING MACHINES

- A turning point in the study of the theory of computation.
- We continue to speak of Turing machines, but
  - Our real focus from now on is on algorithms.
  - Turing machine merely serves as a precise model for the algorithms.

TERMINOLOGY FOR DESCRIBING TURING MACHINES

- A turning point in the study of the theory of computation.
- We continue to speak of Turing machines, but
  - Our real focus from now on is on algorithms.
  - Turing machine merely serves as a precise model for the algorithms.
- We skip over the extensive theory of Turing machines.

TERMINOLOGY FOR DESCRIBING TURING MACHINES

- A turning point in the study of the theory of computation.
- We continue to speak of Turing machines, but
  - Our real focus from now on is on algorithms.
  - Turing machine merely serves as a precise model for the algorithms.
- We skip over the extensive theory of Turing machines.
- We do not spend much time on the low-level programming of Turing machines.

TERMINOLOGY FOR DESCRIBING TURING MACHINES

- A turning point in the study of the theory of computation.
- We continue to speak of Turing machines, but
  - Our real focus from now on is on algorithms.
  - Turing machine merely serves as a precise model for the algorithms.
- We skip over the extensive theory of Turing machines.
- We do not spend much time on the low-level programming of Turing machines.
- Believe that Turing machines capture all algorithms.

Standardizing the way to Describe Turing machine Algorithms

Initial Question

What is the right level of detail to give when describing such algorithms?

Three possibilities

Standardizing the way to Describe Turing machine Algorithms

Initial Question

What is the right level of detail to give when describing such algorithms?

Three possibilities

1. Formal description

Standardizing the way to Describe Turing machine Algorithms

Initial Question

What is the right level of detail to give when describing such algorithms?

Three possibilities

1. Formal description
2. Implementation description

Standardizing the way to Describe Turing machine Algorithms

Initial Question

What is the right level of detail to give when describing such algorithms?

Three possibilities

1. Formal description
2. Implementation description
3. High-level description

Formal Description

Formal Description

- Spells out in full the Turing machines states,

Formal Description

- Spells out in full the Turing machines states, transition function, and

Formal Description

- Spells out in full the Turing machines states, transition function, and so on.

Formal Description

- Spells out in full the Turing machines states, transition function, and so on.
- It is the lowest, most detailed level of description.

Implementation Description

Implementation Description

- We use English prose to describe the way that the Turing machine moves its head and the way that it stores data on its tape.

Implementation Description

- We use English prose to describe the way that the Turing machine moves its head and the way that it stores data on its tape.
- At this level we do not give details of states or transition function.

High Level Description

High Level Description

- We use English prose to describe an algorithm, ignoring the implementation details.

High Level Description

- We use English prose to describe an algorithm, ignoring the implementation details.
- At this level we do not need to mention how the machine manages its tape or head.

- I give formal and implementation-level descriptions of various examples of Turing machines.

- I give formal and implementation-level descriptions of various examples of Turing machines.
- Practicing with lower level Turing machine descriptions helps you understand Turing machines and gain confidence in using them.

- I give formal and implementation-level descriptions of various examples of Turing machines.
- Practicing with lower level Turing machine descriptions helps you understand Turing machines and gain confidence in using them.
- Once you feel confident, high-level descriptions are sufficient.

Setting up Format and Notation for Describing TM

Setting up Format and Notation for Describing TM

- The input to a Turing machine is always a string.

- The input to a Turing machine is always a string.
- If we want to provide an object other than a string as input,

Setting up Format and Notation for Describing TM

- The input to a Turing machine is always a string.
- If we want to provide an object other than a string as input,
  - We must first represent that object as a string.

Setting up Format and Notation for Describing TM

- The input to a Turing machine is always a string.
- If we want to provide an object other than a string as input,
  - We must first represent that object as a string.
- Strings can easily represent

Setting up Format and Notation for Describing TM

- The input to a Turing machine is always a string.
- If we want to provide an object other than a string as input,
  - We must first represent that object as a string.
- Strings can easily represent
  - Polynomials,

Setting up Format and Notation for Describing TM

- The input to a Turing machine is always a string.
- If we want to provide an object other than a string as input,
  - We must first represent that object as a string.
- Strings can easily represent
  - Polynomials,graphs,

Setting up Format and Notation for Describing TM

- The input to a Turing machine is always a string.
- If we want to provide an object other than a string as input,
  - We must first represent that object as a string.
- Strings can easily represent
  - Polynomials,graphs,grammars,

Setting up Format and Notation for Describing TM

- The input to a Turing machine is always a string.
- If we want to provide an object other than a string as input,
  - We must first represent that object as a string.
- Strings can easily represent
  - Polynomials,graphs,grammars, automata, and

Setting up Format and Notation for Describing TM

- The input to a Turing machine is always a string.
- If we want to provide an object other than a string as input,
  - We must first represent that object as a string.
- Strings can easily represent
  - Polynomials,graphs,grammars, automata, and any combination of those objects.

Setting up Format and Notation for Describing TM

- The input to a Turing machine is always a string.
- If we want to provide an object other than a string as input,
  - We must first represent that object as a string.
- Strings can easily represent
  - Polynomials,graphs,grammars, automata, and any combination of those objects.
- A Turing machine may be programmed to decode the representation so that it can be interpreted in the way we intend.

Notation

Notation

- To encode an object O into its representation as a string is $< O >$.

Notation

- To encode an object O into its representation as a string is $< O >$.
- If there are several objects $O_1, O_2, \ldots, O_k$,

Notation

- To encode an object O into its representation as a string is $< O >$.
- If there are several objects $O_1, O_2, \ldots, O_k$,
  - We denote their encoding into a single string $< O_1, O_2, \ldots, O_k >$.

Notation

- To encode an object O into its representation as a string is $< O >$.
- If there are several objects $O_1, O_2, \ldots, O_k$,
  - We denote their encoding into a single string $< O_1, O_2, \ldots, O_k >$.
- The encoding itself can be done in many reasonable.

Notation

- To encode an object O into its representation as a string is $< O >$.
- If there are several objects $O_1, O_2, \ldots, O_k$,
  - We denote their encoding into a single string $< O_1, O_2, \ldots, O_k >$.
- The encoding itself can be done in many reasonable.
- It doesnt matter which one we pick because a Turing machine can always translate one such encoding into another.

Our Format

Our Format

- we describe Turing machine algorithms with an indented segment of text within quotes.

Our Format

- we describe Turing machine algorithms with an indented segment of text within quotes.
- We break the algorithm into stages, each usually involving many individual steps of the Turing machines computation.

Our Format

- we describe Turing machine algorithms with an indented segment of text within quotes.
- We break the algorithm into stages, each usually involving many individual steps of the Turing machines computation.
- We indicate the block structure of the algorithm with further indentation.

Our Format

- we describe Turing machine algorithms with an indented segment of text within quotes.
- We break the algorithm into stages, each usually involving many individual steps of the Turing machines computation.
- We indicate the block structure of the algorithm with further indentation.
- The first line of the algorithm describes the input to the machine.

Our Format

- we describe Turing machine algorithms with an indented segment of text within quotes.
- We break the algorithm into stages, each usually involving many individual steps of the Turing machines computation.
- We indicate the block structure of the algorithm with further indentation.
- The first line of the algorithm describes the input to the machine.
- If the input description is simply w, the input is taken to be a string.

Our Format

- we describe Turing machine algorithms with an indented segment of text within quotes.
- We break the algorithm into stages, each usually involving many individual steps of the Turing machines computation.
- We indicate the block structure of the algorithm with further indentation.
- The first line of the algorithm describes the input to the machine.
- If the input description is simply w, the input is taken to be a string.
- If the input description is the encoding of an object as in $< A >$,

Our Format

- we describe Turing machine algorithms with an indented segment of text within quotes.
- We break the algorithm into stages, each usually involving many individual steps of the Turing machines computation.
- We indicate the block structure of the algorithm with further indentation.
- The first line of the algorithm describes the input to the machine.
- If the input description is simply w, the input is taken to be a string.
- If the input description is the encoding of an object as in $< A >$,
    - The Turing machine first implicitly tests whether

Our Format

- we describe Turing machine algorithms with an indented segment of text within quotes.
- We break the algorithm into stages, each usually involving many individual steps of the Turing machines computation.
- We indicate the block structure of the algorithm with further indentation.
- The first line of the algorithm describes the input to the machine.
- If the input description is simply w, the input is taken to be a string.
- If the input description is the encoding of an object as in $< A >$,
  - The Turing machine first implicitly tests whetherthe input properly encodes an object of the desired form

Our Format

- we describe Turing machine algorithms with an indented segment of text within quotes.
- We break the algorithm into stages, each usually involving many individual steps of the Turing machines computation.
- We indicate the block structure of the algorithm with further indentation.
- The first line of the algorithm describes the input to the machine.
- If the input description is simply w, the input is taken to be a string.
- If the input description is the encoding of an object as in $< A >$,
  - The Turing machine first implicitly tests whetherthe input properly encodes an object of the desired form and rejects it if it doesnt.

EXAMPLE

EXAMPLE

- Let **A** be the language consisting of all strings representing undirected graphs that are connected.

EXAMPLE

- Let **A** be the language consisting of all strings representing undirected graphs that are connected.
- A graph is connected if every node can be reached from every other node by traveling along the edges of the graph.

EXAMPLE

- Let **A** be the language consisting of all strings representing undirected graphs that are connected.
- A graph is connected if every node can be reached from every other node by traveling along the edges of the graph.
    - We write

$$A = \{< G > | G \text{ is a connected undirected graph}\}.$$

High level description of Tm that decides A

High level description of Tm that decides A

- M = "On input $< G >$, the encoding of a graph G:

High level description of Tm that decides A

- M = "On input $< G >$, the encoding of a graph G:
  1. Select the first node of G and mark it.

High level description of Tm that decides A

- M = "On input $< G >$, the encoding of a graph G:
  1. Select the first node of G and mark it.
  2. Repeat the following stage until no new nodes are marked:

High level description of Tm that decides A

- M = "On input $< G >$, the encoding of a graph G:
  1. Select the first node of G and mark it.
  2. Repeat the following stage until no new nodes are marked:
     - For each node in G, mark it if it is attached by an edge to a node that is already marked.

High level description of Tm that decides A

- M = "On input $< G >$, the encoding of a graph G:
  1. Select the first node of G and mark it.
  2. Repeat the following stage until no new nodes are marked:
     - For each node in G, mark it if it is attached by an edge to a node that is already marked.
  3. Scan all the nodes of G to determine whether they all are marked.

High level description of Tm that decides A

- M = "On input $< G >$, the encoding of a graph G:
  1. Select the first node of G and mark it.
  2. Repeat the following stage until no new nodes are marked:
     - For each node in G, mark it if it is attached by an edge to a node that is already marked.
  3. Scan all the nodes of G to determine whether they all are marked. If they are, accept ; otherwise, reject .

Practice

Practice

- Let us examine some implementation-level details of Turing machine M .

Practice

- Let us examine some implementation-level details of Turing machine M .
- Usually you wont need it.

Practice

- Let us examine some implementation-level details of Turing machine M .
- Usually you wont need it.
- First, we must understand how $< G >$ encodes the graph G as a string.

Practice

- Let us examine some implementation-level details of Turing machine M .
- Usually you wont need it.
- First, we must understand how $< G >$ encodes the graph G as a string.
- Consider an encoding that is a list of the nodes of G followed by a list of the edges of G.

Practice

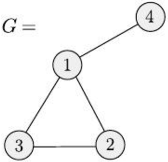
- Let us examine some implementation-level details of Turing machine M .
- Usually you wont need it.
- First, we must understand how $< G >$ encodes the graph G as a string.
- Consider an encoding that is a list of the nodes of G followed by a list of the edges of G.
- Each node is a decimal number, and each edge is the pair of decimal numbers

Practice

- Let us examine some implementation-level details of Turing machine M .
- Usually you wont need it.
- First, we must understand how $< G >$ encodes the graph G as a string.
- Consider an encoding that is a list of the nodes of G followed by a list of the edges of G.
- Each node is a decimal number, and each edge is the pair of decimal numbers that represent the nodes at the two endpoints of the edge.

Focus

$G =$



$\langle G \rangle =$

(1,2,3,4)((1,2),(2,3),(3,1),(1,4))

Contd..

Contd..

- When M receives the input $< G >$, it first checks to determine whether the input is the proper encoding of some graph.

Contd..

- When M receives the input $< G >$, it first checks to determine whether the input is the proper encoding of some graph.
- To do so, M scans the tape to be sure that there are two lists and

Contd..

- When M receives the input $< G >$, it first checks to determine whether the input is the proper encoding of some graph.
- To do so, M scans the tape to be sure that there are two lists and that they are in the proper form.

Contd..

- When M receives the input $< G >$, it first checks to determine whether the input is the proper encoding of some graph.
- To do so, M scans the tape to be sure that there are two lists and that they are in the proper form.
- The first list should be a list of distinct decimal numbers.

- When M receives the input $< G >$, it first checks to determine whether the input is the proper encoding of some graph.
- To do so, M scans the tape to be sure that there are two lists and that they are in the proper form.
- The first list should be a list of distinct decimal numbers.
- And the second should be a list of pairs of decimal numbers.

Contd..

- When M receives the input $< G >$, it first checks to determine whether the input is the proper encoding of some graph.
- To do so, M scans the tape to be sure that there are two lists and that they are in the proper form.
- The first list should be a list of distinct decimal numbers.
- And the second should be a list of pairs of decimal numbers.
- Then M checks several things.

Contd..

- When M receives the input $< G >$, it first checks to determine whether the input is the proper encoding of some graph.
- To do so, M scans the tape to be sure that there are two lists and that they are in the proper form.
- The first list should be a list of distinct decimal numbers.
- And the second should be a list of pairs of decimal numbers.
- Then M checks several things.
    - First, the node list should contain no repetitions;

Contd..

- When M receives the input $< G >$, it first checks to determine whether the input is the proper encoding of some graph.
- To do so, M scans the tape to be sure that there are two lists and that they are in the proper form.
- The first list should be a list of distinct decimal numbers.
- And the second should be a list of pairs of decimal numbers.
- Then M checks several things.
  - First, the node list should contain no repetitions;
  - And second, every node appearing on the edge list should also appear on the node list.

Contd..

- When M receives the input $< G >$, it first checks to determine whether the input is the proper encoding of some graph.
- To do so, M scans the tape to be sure that there are two lists and that they are in the proper form.
- The first list should be a list of distinct decimal numbers.
- And the second should be a list of pairs of decimal numbers.
- Then M checks several things.
  - First, the node list should contain no repetitions;
  - And second, every node appearing on the edge list should also appear on the node list.
- If the input passes these checks, it is the encoding of some graph G.

Contd..

- When M receives the input $< G >$, it first checks to determine whether the input is the proper encoding of some graph.
- To do so, M scans the tape to be sure that there are two lists and that they are in the proper form.
- The first list should be a list of distinct decimal numbers.
- And the second should be a list of pairs of decimal numbers.
- Then M checks several things.
  - First, the node list should contain no repetitions;
  - And second, every node appearing on the edge list should also appear on the node list.
- If the input passes these checks, it is the encoding of some graph G.
- This verification completes the input check, and M goes on to stage 1.

Contd..

Contd..

- For stage 1, M marks the first node with a dot on the leftmost digit.

Contd..

- For stage 1, M marks the first node with a dot on the leftmost digit.
- For stage 2,

Contd..

- For stage 1, M marks the first node with a dot on the leftmost digit.
- For stage 2,
  - M scans the list of nodes to find an undotted node n_1.

Contd..

- For stage 1, M marks the first node with a dot on the leftmost digit.
- For stage 2,
  - M scans the list of nodes to find an undotted node n_1.
  - And flags it by marking it differentlysay, by underlining the first symbol.

Contd..

- For stage 1, M marks the first node with a dot on the leftmost digit.
- For stage 2,
  - M scans the list of nodes to find an undotted node n_1.
  - And flags it by marking it differentlysay, by underlining the first symbol.
  - Then M scans the list again to find a dotted node n_2 and underlines it, too.

- For stage 1, M marks the first node with a dot on the leftmost digit.
- For stage 2,
  - M scans the list of nodes to find an undotted node n_1.
  - And flags it by marking it differentlysay, by underlining the first symbol.
  - Then M scans the list again to find a dotted node n_2 and underlines it, too.
- Now M scans the list of edges.

Contd..

- For stage 1, M marks the first node with a dot on the leftmost digit.
- For stage 2,
  - M scans the list of nodes to find an undotted node n_1.
  - And flags it by marking it differentlysay, by underlining the first symbol.
  - Then M scans the list again to find a dotted node n_2 and underlines it, too.
- Now M scans the list of edges.
- For each edge, M tests whether the two underlined nodes n_1 and n_2 are the ones appearing in that edge.

Contd..

- For stage 1, M marks the first node with a dot on the leftmost digit.
- For stage 2,
  - M scans the list of nodes to find an undotted node n_1.
  - And flags it by marking it differentlysay, by underlining the first symbol.
  - Then M scans the list again to find a dotted node n_2 and underlines it, too.
- Now M scans the list of edges.
- For each edge, M tests whether the two underlined nodes n_1 and n_2 are the ones appearing in that edge.
- If they are, M dots n_1, removes the underlines,

Contd..

- For stage 1, M marks the first node with a dot on the leftmost digit.
- For stage 2,
  - M scans the list of nodes to find an undotted node n_1.
  - And flags it by marking it differentlysay, by underlining the first symbol.
  - Then M scans the list again to find a dotted node n_2 and underlines it, too.
- Now M scans the list of edges.
- For each edge, M tests whether the two underlined nodes n_1 and n_2 are the ones appearing in that edge.
- If they are, M dots n_1, removes the underlines, and goes on from the beginning of stage 2.

Contd..

- For stage 1, M marks the first node with a dot on the leftmost digit.
- For stage 2,
  - M scans the list of nodes to find an undotted node n_1.
  - And flags it by marking it differentlysay, by underlining the first symbol.
  - Then M scans the list again to find a dotted node n_2 and underlines it, too.
- Now M scans the list of edges.
- For each edge, M tests whether the two underlined nodes n_1 and n_2 are the ones appearing in that edge.
- If they are, M dots n_1, removes the underlines, and goes on from the beginning of stage 2.
- If they arent, M checks the next edge on the list.

Contd..

Contd..

- If there are no more edges,

Contd..

- If there are no more edges,
  - $\{n_1, n_2\}$ is not an edge of G.

Contd..

- If there are no more edges,
  - $\{n_1, n_2\}$ is not an edge of G.
  - Then M moves the underline on n_2 to the next dotted node and now calls this node n_2 .

Contd..

- If there are no more edges,
  - $\{n_1, n_2\}$ is not an edge of G.
  - Then M moves the underline on n_2 to the next dotted node and now calls this node n_2 .
- It repeats these steps to check whether the new pair $\{n_1, n_2\}$ is an edge.

- If there are no more edges,
  - $\{n_1, n_2\}$ is not an edge of G.
  - Then M moves the underline on n_2 to the next dotted node and now calls this node n_2 .
- It repeats these steps to check whether the new pair $\{n_1, n_2\}$ is an edge.
- If there are no more dotted nodes, n_1 is not attached to any dotted nodes.

Contd..

- If there are no more edges,
  - $\{n_1, n_2\}$ is not an edge of G.
  - Then M moves the underline on n_2 to the next dotted node and now calls this node n_2 .
- It repeats these steps to check whether the new pair $\{n_1, n_2\}$ is an edge.
- If there are no more dotted nodes, n_1 is not attached to any dotted nodes.
- Then M sets the underlines so that n_1 is the next undotted node and n_2 is the first dotted node

Contd..

- If there are no more edges,
  - $\{n_1, n_2\}$ is not an edge of G.
  - Then M moves the underline on n_2 to the next dotted node and now calls this node n_2 .
- It repeats these steps to check whether the new pair $\{n_1, n_2\}$ is an edge.
- If there are no more dotted nodes, n_1 is not attached to any dotted nodes.
- Then M sets the underlines so that n_1 is the next undotted node and n_2 is the first dotted node and repeats the steps.

Contd..

- If there are no more edges,
  - $\{n_1, n_2\}$ is not an edge of G.
  - Then M moves the underline on n_2 to the next dotted node and now calls this node n_2 .
- It repeats these steps to check whether the new pair $\{n_1, n_2\}$ is an edge.
- If there are no more dotted nodes, n_1 is not attached to any dotted nodes.
- Then M sets the underlines so that n_1 is the next undotted node and n_2 is the first dotted node and repeats the steps.
- If there are no more undotted nodes, M has not been able to find any new nodes to dot, so it moves on to stage 4.

Contd..

- If there are no more edges,
  - $\{n_1, n_2\}$ is not an edge of G.
  - Then M moves the underline on n_2 to the next dotted node and now calls this node n_2 .
- It repeats these steps to check whether the new pair $\{n_1, n_2\}$ is an edge.
- If there are no more dotted nodes, n_1 is not attached to any dotted nodes.
- Then M sets the underlines so that n_1 is the next undotted node and n_2 is the first dotted node and repeats the steps.
- If there are no more undotted nodes, M has not been able to find any new nodes to dot, so it moves on to stage 4.
- For stage 4, M scans the list of nodes to determine whether all are dotted.

Contd..

- If there are no more edges,
  - $\{n_1, n_2\}$ is not an edge of G.
  - Then M moves the underline on n_2 to the next dotted node and now calls this node n_2 .
- It repeats these steps to check whether the new pair $\{n_1, n_2\}$ is an edge.
- If there are no more dotted nodes, n_1 is not attached to any dotted nodes.
- Then M sets the underlines so that n_1 is the next undotted node and n_2 is the first dotted node and repeats the steps.
- If there are no more undotted nodes, M has not been able to find any new nodes to dot, so it moves on to stage 4.
- For stage 4, M scans the list of nodes to determine whether all are dotted.
- If they are, it enters the accept state; otherwise, it enters the reject state.

THANK YOU