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.

Investigate the Power of Algorithms

Investigate the Power of Algorithms

- I begin to investigate the power of algorithms to solve problems.

Investigate the Power of Algorithms

- I begin to investigate the power of algorithms to solve problems.
- I demonstrate certain problems that can be solved algorithmically and others that cannot.

Investigate the Power of Algorithms

- I begin to investigate the power of algorithms to solve problems.
- I demonstrate certain problems that can be solved algorithmically and others that cannot.
- My objective is to explore the limits of algorithmic solvability.

Investigate the Power of Algorithms

- I begin to investigate the power of algorithms to solve problems.
- I demonstrate certain problems that can be solved algorithmically and others that cannot.
- My objective is to explore the limits of algorithmic solvability.
- You are probably familiar with solvability by algorithms because much of computer science is devoted to solving problems.

Investigate the Power of Algorithms

- I begin to investigate the power of algorithms to solve problems.
- I demonstrate certain problems that can be solved algorithmically and others that cannot.
- My objective is to explore the limits of algorithmic solvability.
- You are probably familiar with solvability by algorithms because much of computer science is devoted to solving problems.
- The unsolvability of certain problems may come as a surprise.

Study of Unsolvability

Study of Unsolvability

- Why should you study unsolvability?

Study of Unsolvability

- Why should you study unsolvability?
- After all, showing that a problem is unsolvable doesnt appear to be of any use if you have to solve it.

Study of Unsolvability

- Why should you study unsolvability?
- After all, showing that a problem is unsolvable doesnt appear to be of any use if you have to solve it.
- You need to study this phenomenon for two reasons.

Study of Unsolvability

- Why should you study unsolvability?
- After all, showing that a problem is unsolvable doesnt appear to be of any use if you have to solve it.
- You need to study this phenomenon for two reasons.
    - First, knowing when a problem is algorithmically unsolvable is useful.

Study of Unsolvability

- Why should you study unsolvability?
- After all, showing that a problem is unsolvable doesnt appear to be of any use if you have to solve it.
- You need to study this phenomenon for two reasons.
    - First, knowing when a problem is algorithmically unsolvable is useful.
    - Then you realize that the **problem must be simplified or altered** before you can find an algorithmic solution.

Study of Unsolvability

- Why should you study unsolvability?
- After all, showing that a problem is unsolvable doesnt appear to be of any use if you have to solve it.
- You need to study this phenomenon for two reasons.
    - First, knowing when a problem is algorithmically unsolvable is useful.
    - Then you realize that the **problem must be simplified or altered** before you can find an algorithmic solution.
    - Like any tool, **computers have capabilities and limitations** that must be appreciated if they are to be used well.

Study of Unsolvability

- Why should you study unsolvability?
- After all, showing that a problem is unsolvable doesnt appear to be of any use if you have to solve it.
- You need to study this phenomenon for two reasons.
    - First, knowing when a problem is algorithmically unsolvable is useful.
    - Then you realize that the **problem must be simplified or altered** before you can find an algorithmic solution.
    - Like any tool, **computers have capabilities and limitations** that must be appreciated if they are to be used well.
- The second reason is cultural.

Study of Unsolvability

- Why should you study unsolvability?
- After all, showing that a problem is unsolvable doesnt appear to be of any use if you have to solve it.
- You need to study this phenomenon for two reasons.
  - First, knowing when a problem is algorithmically unsolvable is useful.
  - Then you realize that the **problem must be simplified or altered** before you can find an algorithmic solution.
  - Like any tool, **computers have capabilities and limitations** that must be appreciated if they are to be used well.
- The second reason is cultural.
  - Even if you deal with problems that clearly are solvable,

Study of Unsolvability

- Why should you study unsolvability?
- After all, showing that a problem is unsolvable doesnt appear to be of any use if you have to solve it.
- You need to study this phenomenon for two reasons.
    - First, knowing when a problem is algorithmically unsolvable is useful.
    - Then you realize that the **problem must be simplified or altered** before you can find an algorithmic solution.
    - Like any tool, **computers have capabilities and limitations** that must be appreciated if they are to be used well.
- The second reason is cultural.
    - Even if you deal with problems that clearly are solvable,
        - A glimpse of the unsolvable can stimulate your imagination and

Study of Unsolvability

- Why should you study unsolvability?
- After all, showing that a problem is unsolvable doesnt appear to be of any use if you have to solve it.
- You need to study this phenomenon for two reasons.
    - First, knowing when a problem is algorithmically unsolvable is useful.
    - Then you realize that the **problem must be simplified or altered** before you can find an algorithmic solution.
    - Like any tool, **computers have capabilities and limitations** that must be appreciated if they are to be used well.
- The second reason is cultural.
    - Even if you deal with problems that clearly are solvable,
        - A glimpse of the unsolvable can stimulate your imagination and
        - Help you gain an important perspective on computation.

DECIDABLE LANGUAGES

DECIDABLE LANGUAGES

- I give some examples of languages that are decidable by algorithms.

DECIDABLE LANGUAGES

- I give some examples of languages that are decidable by algorithms.
- I focus on languages concerning automata and grammars.

DECIDABLE LANGUAGES

- I give some examples of languages that are decidable by algorithms.
- I focus on languages concerning automata and grammars.
- I present an algorithm that tests whether a string is a member of a context-free language (CFL).

DECIDABLE LANGUAGES

- I give some examples of languages that are decidable by algorithms.
- I focus on languages concerning automata and grammars.
- I present an algorithm that tests whether a string is a member of a context-free language (CFL).
- These languages are interesting for several reasons.

DECIDABLE LANGUAGES

- I give some examples of languages that are decidable by algorithms.
- I focus on languages concerning automata and grammars.
- I present an algorithm that tests whether a string is a member of a context-free language (CFL).
- These languages are interesting for several reasons.
    - First, certain problems of this kind are related to applications.

DECIDABLE LANGUAGES

- I give some examples of languages that are decidable by algorithms.
- I focus on languages concerning automata and grammars.
- I present an algorithm that tests whether a string is a member of a context-free language (CFL).
- These languages are interesting for several reasons.
    - First, certain problems of this kind are related to applications.
    - This problem of testing whether a CFG generates a string is related

DECIDABLE LANGUAGES

- I give some examples of languages that are decidable by algorithms.
- I focus on languages concerning automata and grammars.
- I present an algorithm that tests whether a string is a member of a context-free language (CFL).
- These languages are interesting for several reasons.
  - First, certain problems of this kind are related to applications.
  - This problem of testing whether a CFG generates a string is relatedto the problem of recognizing and compiling programs in a programming language.

DECIDABLE LANGUAGES

- I give some examples of languages that are decidable by algorithms.
- I focus on languages concerning automata and grammars.
- I present an algorithm that tests whether a string is a member of a context-free language (CFL).
- These languages are interesting for several reasons.
    - First, certain problems of this kind are related to applications.
    - This problem of testing whether a CFG generates a string is relatedto the problem of recognizing and compiling programs in a programming language.
    - Second, certain other problems concerning automata and grammars are not decidable by algorithms.

DECIDABLE LANGUAGES

- I give some examples of languages that are decidable by algorithms.
- I focus on languages concerning automata and grammars.
- I present an algorithm that tests whether a string is a member of a context-free language (CFL).
- These languages are interesting for several reasons.
    - First, certain problems of this kind are related to applications.
    - This problem of testing whether a CFG generates a string is relatedto the problem of recognizing and compiling programs in a programming language.
    - Second, certain other problems concerning automata and grammars are not decidable by algorithms.
    - Starting with examples where decidability is possible helps you to appreciate the undecidable examples.

DECIDABLE PROBLEMS CONCERNING REGULAR LANGUAGES

ALgorithms for Regular Languages

ALgorithms for Regular Languages

- I begin with certain computational problems concerning finite automata.

ALgorithms for Regular Languages

- I begin with certain computational problems concerning finite automata.
- I give algorithms for testing

ALgorithms for Regular Languages

- I begin with certain computational problems concerning finite automata.
- I give algorithms for testing
  - Whether a finite automaton accepts a string.

ALgorithms for Regular Languages

- I begin with certain computational problems concerning finite automata.
- I give algorithms for testing
  - Whether a finite automaton accepts a string.
  - Whether the language of a finite automaton is empty.

ALgorithms for Regular Languages

- I begin with certain computational problems concerning finite automata.
- I give algorithms for testing
  - Whether a finite automaton accepts a string.
  - Whether the language of a finite automaton is empty.
  - Whether two finite automata are equivalent.

Preliminaries

Preliminaries

- I chose to represent various computational problems by languages.

Preliminaries

- I chose to represent various computational problems by languages.
- Doing so is convenient because we have already set up terminology for dealing with languages.

Preliminaries

- I chose to represent various computational problems by languages.
- Doing so is convenient because we have already set up terminology for dealing with languages.
- For example, **the acceptance problem** for DFAs

Preliminaries

- I chose to represent various computational problems by languages.
- Doing so is convenient because we have already set up terminology for dealing with languages.
- For example, **the acceptance problem** for DFAs of testing whether a particular deterministic finite automaton accepts a given string

Preliminaries

- I chose to represent various computational problems by languages.
- Doing so is convenient because we have already set up terminology for dealing with languages.
- For example, **the acceptance problem** for DFAs of testing whether a particular deterministic finite automaton accepts a given string can be expressed as a language, A_{DFA} .

Preliminaries

- I chose to represent various computational problems by languages.
- Doing so is convenient because we have already set up terminology for dealing with languages.
- For example, **the acceptance problem** for DFAs of testing whether a particular deterministic finite automaton accepts a given string can be expressed as a language, A_{DFA} .
- This language contains the encodings of all DFAs together with strings that the DFAs accept. Let

  $A_{DFA} = \{< B, w > | B \text{ is a } DFA \text{ that accepts input string } w\}$.

Preliminaries

- I chose to represent various computational problems by languages.
- Doing so is convenient because we have already set up terminology for dealing with languages.
- For example, **the acceptance problem** for DFAs of testing whether a particular deterministic finite automaton accepts a given string can be expressed as a language, A_{DFA} .
- This language contains the encodings of all DFAs together with strings that the DFAs accept. Let

  $A_{DFA} = \{< B, w > | B \text{ is a } DFA \text{ that accepts input string } w\}$.
- The problem of testing whether a DFA B accepts an input w is the same as

Preliminaries

- I chose to represent various computational problems by languages.
- Doing so is convenient because we have already set up terminology for dealing with languages.
- For example, **the acceptance problem** for DFAs of testing whether a particular deterministic finite automaton accepts a given string can be expressed as a language, A_{DFA} .
- This language contains the encodings of all DFAs together with strings that the DFAs accept. Let

  $A_{DFA} = \{< B, w > | B \text{ is a } DFA \text{ that accepts input string } w\}$.
- The problem of testing whether a DFA B accepts an input w is the same as
  - The problem of testing whether $< B, w >$, is a member of the language A_{DFA}.

Contd..

Contd..

- Similarly, we can formulate other computational problems in terms of testing membership in a language.

Contd..

- Similarly, we can formulate other computational problems in terms of testing membership in a language.
- Showing that the **language is decidable** is the same as

Contd..

- Similarly, we can formulate other computational problems in terms of testing membership in a language.
- Showing that the **language is decidable** is the same as showing that the **computational problem is decidable.**

Theorem

Theorem

Theorem

A_{DFA} is decidable language.

Theorem

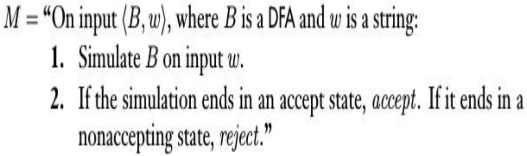
Theorem

A_{DFA} is decidable language.

Proof Idea

We simply need to present a TM M that decides A_{DFA}.

Algorithm



```
$M$ = "On input $\langle B, w \rangle$, where $B$ is a DFA and $w$ is a string:
    1. Simulate $B$ on input $w$.
    2. If the simulation ends in an accept state, *accept*. If it ends in a
       nonaccepting state, *reject*."
```

Proof

Proof

- I mention just a few implementation details of this proof.

Proof

- I mention just a few implementation details of this proof.
- For those of you familiar with writing programs in any standard programming lan- guage,

Proof

- I mention just a few implementation details of this proof.
- For those of you familiar with writing programs in any standard programming lan- guage,
- Imagine how you would write a program to carry out the simulation.

Proof

- I mention just a few implementation details of this proof.
- For those of you familiar with writing programs in any standard programming lan- guage,
- Imagine how you would write a program to carry out the simulation.
- First, lets examine the input $< B, w >$.

Proof

- I mention just a few implementation details of this proof.
- For those of you familiar with writing programs in any standard programming lan- guage,
- Imagine how you would write a program to carry out the simulation.
- First, lets examine the input $< B, w >$.
- It is a representation of a DFA B together with a string w.

Proof

- I mention just a few implementation details of this proof.
- For those of you familiar with writing programs in any standard programming lan- guage,
- Imagine how you would write a program to carry out the simulation.
- First, lets examine the input $< B, w >$.
- It is a representation of a DFA B together with a string w.
- One reasonable representation of B is simply

Proof

- I mention just a few implementation details of this proof.
- For those of you familiar with writing programs in any standard programming lan- guage,
- Imagine how you would write a program to carry out the simulation.
- First, lets examine the input $< B, w >$.
- It is a representation of a DFA B together with a string w.
- One reasonable representation of B is simply
  - A list of its five components: $Q, , , q_0$, and F .

Proof

- I mention just a few implementation details of this proof.
- For those of you familiar with writing programs in any standard programming lan- guage,
- Imagine how you would write a program to carry out the simulation.
- First, lets examine the input $< B, w >$.
- It is a representation of a DFA B together with a string w.
- One reasonable representation of B is simply
  - A list of its five components: $Q, , , q_0$, and F .
- When M receives its input,

Proof

- I mention just a few implementation details of this proof.
- For those of you familiar with writing programs in any standard programming lan- guage,
- Imagine how you would write a program to carry out the simulation.
- First, lets examine the input $< B, w >$.
- It is a representation of a DFA B together with a string w.
- One reasonable representation of B is simply
  - A list of its five components: $Q, , , q_0$, and F .
- When M receives its input,
  - M first determines whether it properly represents a DFA B and a string w.

Proof

- I mention just a few implementation details of this proof.
- For those of you familiar with writing programs in any standard programming lan- guage,
- Imagine how you would write a program to carry out the simulation.
- First, lets examine the input $< B, w >$.
- It is a representation of a DFA B together with a string w.
- One reasonable representation of B is simply
  - A list of its five components: $Q, , , q_0$, and F .
- When M receives its input,
  - M first determines whether it properly represents a DFA B and a string w.
  - If not, M rejects.

Contd..

Contd..

- Then M carries out the simulation directly.

Contd..

- Then M carries out the simulation directly.
- It keeps track of B's current state and

Contd..

- Then M carries out the simulation directly.
- It keeps track of B's current state and Bs current position in the input *w* by writing this information down on its tape.

Contd..

- Then M carries out the simulation directly.
- It keeps track of B's current state and Bs current position in the input w by writing this information down on its tape.
- Initially, B's current state is q_0 and

Contd..

- Then M carries out the simulation directly.
- It keeps track of B's current state and Bs current position in the input w by writing this information down on its tape.
- Initially, B's current state is q_0 and
- B's current input position is the leftmost symbol of w.

Contd..

- Then M carries out the simulation directly.
- It keeps track of B's current state and Bs current position in the input w by writing this information down on its tape.
- Initially, B's current state is q_0 and
- B's current input position is the leftmost symbol of w.
- he states and position are updated according to the specified transition function δ.

Contd..

- Then M carries out the simulation directly.
- It keeps track of B's current state and Bs current position in the input w by writing this information down on its tape.
- Initially, B's current state is q_0 and
- B's current input position is the leftmost symbol of w.
- he states and position are updated according to the specified transition function δ.
- When M finishes processing the last symbol of w,

Contd..

- Then M carries out the simulation directly.
- It keeps track of B's current state and Bs current position in the input w by writing this information down on its tape.
- Initially, B's current state is q_0 and
- B's current input position is the leftmost symbol of w.
- he states and position are updated according to the specified transition function δ.
- When M finishes processing the last symbol of w,
- M accepts the input if B is in an accepting state;

Contd..

- Then M carries out the simulation directly.
- It keeps track of B's current state and Bs current position in the input w by writing this information down on its tape.
- Initially, B's current state is q_0 and
- B's current input position is the leftmost symbol of w.
- he states and position are updated according to the specified transition function δ.
- When M finishes processing the last symbol of w,
- M accepts the input if B is in an accepting state;
- M rejects the input if B is in a nonaccepting state.

THANK YOU