Rice's Theorem

Computability and Complexity
Andrei Bulatov

Computation about Computation

We have seen that some problems concerning operation of TMs and other computational models are undecidable

In this lecture we shall see that this is an inherent property of all problems about computation

We show that the question "Does my program do what I want it to do?" is normally undecidable

Known Undecidable Problems

HALTING

Instance: A Turing Machine T and an input string x.

Question: Does $T(x)$ halt?

ACCEPTANCE

Instance: A Turing Machine (acceptor) T and an input string x.

Question: Does T accept x?

More Undecidable Problems

EMPTINESS

Instance: A Turing Machine T.

Question: Is the language $L(T)$ empty?

FULLNESS

Instance: A Turing Machine T.

Question: Is the language $L(T)$ equal to Σ^*?

The corresponding languages are:

$$L_{\text{Empty}} = \{"T" \mid L(T) = \varnothing\} \qquad L_{\text{Full}} = \{"T" \mid L(T) = \Sigma^*\}$$

Theorem L_{Empty} and L_{Full} are undecidable.

Proof

We show that $L_{\text{Halting}} \leq_m L_{\text{Full}}$

For every input "$T;x$" of L_{Halting}, let S be a machine operating on an input y as follows:

- Erase input y
- Write x on the tape
- Simulate T on x
- If $T(x)$ halts then "Accept"

Observe that

- If T halts on x, then $L(S)=\Sigma^*$ (i.e., S accepts every input y)
- If T does not halt on x, then $L(S)=\varnothing$

This function is computable and total ?

QED

A proof of $L_{\text{Halting}} \leq_m L_{\text{Empty}}$ is similar

The EQUIVALENCE Problem

EQUIVALENCE

Instance: Turing Machines T_1 and T_2.

Question: $\mathsf{L}(T_1) = \mathsf{L}(T_2)$?

The corresponding language is:

$$\mathsf{L}_{\text{Equiv}} = \{"T_1;T_2" \mid \mathsf{L}(T_1) = \mathsf{L}(T_2)\}$$

Theorem $\mathsf{L}_{\text{Equiv}}$ is undecidable.

Proof

- see next slide

We show that $\mathsf{L}_{\text{Empty}} \leq_m \mathsf{L}_{\text{Equiv}}$

Fix a TM T_0 with $\mathsf{L}(T_0) = \varnothing$

For every input "T" of $\mathsf{L}_{\text{Empty}}$, define an input of $\mathsf{L}_{\text{Equiv}}$ as $"T;T_0"$

Then $\mathsf{L}(T)=\varnothing$ if and only if $\mathsf{L}(T) = \mathsf{L}(T_0)$

QED

Properties of TMs

Usually, a problem can be solved in many different ways.

There are many (different) programs with indistinguishable behaviour

Definition A collection, R, of TM descriptions is called

- a **property** if, for any TMs T_1 and T_2, if $\mathsf{L}(T_1) = \mathsf{L}(T_2)$ then
  either T_1 and T_2 are in R,
  or T_1 and T_2 are not in R
- a **non-trivial property** if there exists a TM which is in R and there exists a TM which is not in R

Examples

- Does a TM halt on every input?
- Does a TM accepts any input?   (FULLNESS)
- Does a TM rejects any input?   (EMPTYNESS)
- Does a TM output the sum of two given naturals?
- Does a TM accept only strings longer than 3 symbols?
- Does a TM ever leave its initial state?
- Does a TM print its own description?

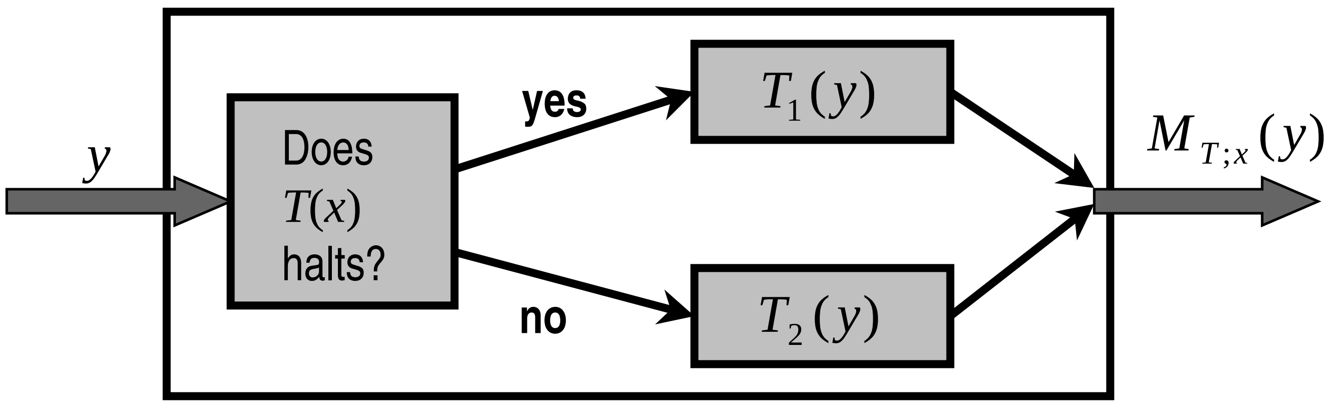
Rice's Theorem

Theorem	Any non-trivial property of TMs is undecidable.

Proof Idea

Suppose that there exists a decidable non-trivial property R.
We show that $L_{\text{Halting}} \leq_m R$

Fix $T_1 \in R$ and $T_2 \notin R$. For every input "$T;x$" of the HALTING problem, we built TM $M_{T;x}$ that work as follows



Observe that

- If $T(x)$ halts then $L(M_{T;x}) = L(T_1) \Rightarrow M_{T;x} \in R$
- If $T(x)$ does not halt then $L(M_{T;x}) = L(T_2) \Rightarrow M_{T;x} \notin R$

Therefore if we are able to decide whether $M_{T;x} \in R$ or not, we are able to decide whether $T(x)$ halts or not

Case 1. There is a TM $T_2 \notin R$ such that $\mathsf{L}(T_2) = \varnothing$

For every input "$T;x$" of $\mathsf{L}_{\text{Halting}}$, let $M_{T;x}$ be a machine operating on an input y as follows:

- Simulate T on x
- If $T(x)$ terminates, simulate T_1 on y

Then

- If $T(x)$ halts then $\mathsf{L}(M_{T;x}) = \mathsf{L}(T_1) \Rightarrow M_{T;x} \in R$
- If $T(x)$ does not halt then $\mathsf{L}(M_{T;x}) = \varnothing = \mathsf{L}(T_2) \Rightarrow M_{T;x} \notin R$

Case 2. There is a TM $T_1 \in R$ such that $\mathsf{L}(T_1) = \varnothing$

For every input "$T;x$" of $\mathsf{L}_{\text{Halting}}$, let $M_{T;x}$ be a machine operating on an input y as follows:

- Simulate T on x
- If $T(x)$ terminates, simulate T_2 on y

Then

- If $T(x)$ halts then $\mathsf{L}(M_{T;x}) = \mathsf{L}(T_2) \Rightarrow M_{T;x} \notin R$
- If $T(x)$ does not halt then $\mathsf{L}(M_{T;x}) = \varnothing = \mathsf{L}(T_1) \Rightarrow M_{T;x} \in R$

Example

Let e be a graph encoding scheme and

$$\mathsf{L}=\{x \mid x \text{ is the code of a Hamilton graph}\}$$

Property

$$R=\{T \mid \mathsf{L}(T)=\mathsf{L}\}$$

Then the question

"Does a TM possess property R?"

=

"Does my program recognises Hamiltonian graphs?"

is undecidable!

It is impossible to automatize software verification