Computational Models — Lecture 9

Handout Mode

Iftach Haitner.

Tel Aviv University.

December 11, 2018

---

1 Based on frames by Benny Chor, Tel Aviv University, modifying frames by Maurice Herlihy, Brown University.
Talk Outline

- Computable functions
- Reducibility
- Mapping reductions
- Busy Beaver
- Rice theorem and friends

Sipser’s book, 5.1 – 5.2
Reminder

We have already

- Established Turing Machines as the **gold standard** of computers and computability . . .
- Seen examples of decidable problems . . .
- Saw that

\[ A_{TM} = \{ \langle M, w \rangle : M \text{ is a TM that accepts } w \} \]

and

\[ H_{TM} = \{ \langle M, w \rangle : M \text{ is a TM and } M \text{ halts on input } w \} \]

are undecidable.

Today, we look at other computationally undecidable problems via **reductions** and introduce the techniques of **mapping reductions**.
Section 1

Computable Functions
Computable functions

### Definition 1 (total computable functions)

A TM $M$ computes a total function $f : \Sigma^* \mapsto \Sigma^*$, if when starting with an input $w$, $M$ halts with (only) $f(w)$ written on tape.$^a$

---

$^a$The definition naturally extends to functions of more than one variable, where some special separator symbol indicates end of one variable and beginning of next.

Computable functions are also called recursive functions.
Claim 2

All the “usual” arithmetic functions on integers are computable. These include addition, subtraction, multiplication, division (quotient and remainder), exponentiation, roots (to a specified precision), modular exponentiation, greatest common divisor.

Even non-arithmetic functions, like logarithms and trigonometric functions, can be computed (to a specified precision), using Taylor expansion or other numeric mathematic techniques.

Exercise 3

Design a TM that on input $\langle m, n \rangle$, halts with $\langle m + n \rangle$ on tape.
Example, 2

A useful class of functions modifies TM descriptions. For example:

**Algorithm 4**

On input $w$:
If $w = \langle M \rangle$ for some TM $M$, return $\langle \tilde{M} \rangle$, where $\tilde{M}$ is TM s.t.,
- $L(\tilde{M}) = L(M)$, and
- $\tilde{M}$ does not halt on $x \notin L(\tilde{M})$

Otherwise, return $\varepsilon$.

**Question 5**

Is the function defined above total? computable?

Are all total functions computable?

No, $f(\langle M, w \rangle)$ that returns 1 if $M$ accepts $w$, and 0 otherwise, is not computable.
Definition of $\tilde{M}$

Given $M = (Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$, TM $\tilde{M} = (Q', \Sigma, \Gamma, \delta', q_0, q_a, q_r)$ is defined by

- $Q' = Q \cup \{q^*\}$

\[
\delta'(q, \sigma) := \begin{cases} 
(q^*, \sigma, R), & \delta(q, \sigma) = (q_r, \cdot, \cdot) \\
(q^*, \sigma, R) & q = q^* \\
(q', \sigma', D), & \delta(q, \sigma) = (q', \sigma', D) \land q \notin \{q_r, q^*\}
\end{cases}
\]
Section 2

Reducibility
Reducibility

- Finding your way around a new city, reduces to . . . obtaining a city map.
- Finding the median in an array, reduces to . . . sorting an array
- The core idea behind procedures
Reducibility, in our context

Involves two languages $A$ and $B$.

Desired property: If $A$ reduces to $B$, then any solution of $B$ can be used to find a solution of $A$.

This property says nothing about solving $A$ by itself, or $B$ by itself. but if $A$ reduces to $B$, then $A$ cannot be harder than $B$:

- if $B$ is decidable, so is $A$.
- if $A$ is undecidable, then $B$ is undecidable.

We next use reductions and the undecidability of $A_{TM}$, to show the undecidability of several problems.
Reminder, $H_{TM}$ is undecidable

- $A_{TM} = \{\langle M, w \rangle : M \text{ is a TM that accepts } w \}$
- $H_{TM} = \{\langle M, w \rangle : M \text{ is a TM and } M \text{ halts on input } w \}$

### Theorem 6

$H_{TM}$ is undecidable.

**Proof:** Assume, by way of contradiction, that $\exists$ TM $R$ that decides $H_{TM}$.

#### Algorithm 7 ($S$)

On input $\langle M, w \rangle$.

Reject, if input is not well formatted.

1. Emulate $R$ on $\langle M, w \rangle$.
2. If $R$ rejects, reject.
3. If $R$ accepts, emulate $M$ on $w$ until it halts.
4. Accept if $M$ accepted; otherwise reject.

TM $S$ decides $A_{TM}$, a contradiction ♠

What we actually did is a reduction from $A_{TM}$ to $H_{TM}$.
**Theorem 8**

**EMPTY**\textsubscript{TM} is undecidable.

Proof’s idea: By contradiction:

- Assume **EMPTY**\textsubscript{TM} is decidable and let \( R \) be a TM that decides **EMPTY**\textsubscript{TM}.
- Use \( R \) to construct \( S \), a TM that decides \( A\textsubscript{TM} \).

**Algorithm 9 (S – first attempt)**

On input \( \langle M, w \rangle \):

- Emulate \( R(\langle M \rangle) \) and reject if \( R \) accepts.

But what if \( R \) rejects?

Solution? Modify \( M \).
EMPTY$_{TM}$ is undecidable, the TM $M_w$

**Definition 10 ($M_w$)**

For TM $M$ and input $w$, the TM $M_w$ is defined as follows:
On input $x$, emulate $M$ on $w$, and accept if $M$ does.

$$L(M_w) = \begin{cases} \Sigma^* & w \in L(M) \\ \emptyset & \text{otherwise} \end{cases}$$

**Question 11**

Can a TM construct $M_w$ from $\langle M, w \rangle$?

**Answer:** Easily because we need only hardwire $w$, and add a few extra states to replace $x$ with $w$. 
**EMPTY**\(_\text{TM}\) is undecidable, the reduction

- **EMPTY**\(_\text{TM}\) = \{\langle M \rangle : M \text{ is a TM and } L(M) = \emptyset\}
- Assume **EMPTY**\(_\text{TM}\) is decidable and let \(R\) be a TM that decides **EMPTY**\(_\text{TM}\).

**Algorithm 12 (S)**

On input \(\langle M, w \rangle\). Reject, if input is not well formatted.

1. Construct \(M_w\).
2. Emulate \(R\) on input \(\langle M_w \rangle\).
3. Accept if \(R\) Rejects; reject if \(R\) accepts.

**Claim 13**

\(S\) decides \(A_{\text{TM}}\).

**Proof:** Recall that \(L(M_w) = \emptyset\) iff \(M\) does not accept \(w\). \(\square\)

A contradiction.
There is undecidable, building $M_w$

Given $M = (Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$ and $w = w_1, \ldots, w_n$ (assuming $n \geq 1$),

The TM $M_w = (Q', \Sigma, \Gamma, \delta', \overrightarrow{q}_1, q_a, q_r)$ is defined by

$Q' = Q \cup \{\overrightarrow{q}_1, \ldots, \overrightarrow{q}_{n+1}, \overleftarrow{q}, \overleftarrow{q}_1\}$, and

$$
\delta'(q, \sigma) := \begin{cases} 
(q_2, $, R) & q = \overrightarrow{q}_1 \\
(q_{i+1}, w_i, R) & q = \overrightarrow{q}_i, 2 \leq i \leq n \\
(q_{n+1}, $, R) & q = \overrightarrow{q}_{n+1}, \sigma \neq $ \\
(q, \sigma, L) & q = \overleftarrow{q}_{n+1}, \sigma = $ \\
(q, \sigma, L) & q = \overleftarrow{q}, \sigma \neq $ \\
(q_1, w_1, R) & q = \overrightarrow{q}, \sigma = $ \\
(q_0, \sigma, L) & q = \overleftarrow{q} \\
(q', \sigma', D), & \delta(q, \sigma) = (q', \sigma', D)
\end{cases}
$$

and none of the above conditions hold
**REG\_TM\_is\_undecidable**

\[ \text{REG}_\text{TM} = \{ \langle M \rangle : M \text{ is a TM and } L(M) \text{ is regular} \} \]

**Theorem 14**

\textit{REG}\_\textit{TM} is undecidable.

Proof’s idea:  By contradiction.

- Assume \textit{REG}\_\textit{TM} is decidable and let \( R \) be a TM that decides \textit{REG}\_\textit{TM}.
- Use \( R \) to construct \( S \) – a TM that decides A\_\textit{TM}.

**Question 15**

But how we construct \( S \)?

Intuition: On input \( \langle M, w \rangle \), build \( M_w \) that accepts regular language iff \( M \) accepts \( w \).
REG$_{TM}$ is undecidable, the TM $M_w$

The TM $M_w$ will have the following property.

- **if** $M$ **does not accept** $w$, then $L(M_w) = \{0^n1^n : n \geq 0\}$
- **if** $M$ **does accept** $w$, then $L(M_w) = \Sigma^*$

### Algorithm 16 ($M_w$)

On input $x$,

1. **Accept**, if $x$ is of the form $0^n1^n$.
2. (otherwise) Emulate $M$ on input $w$ and accept if $M$ does.

### Claim 17

1. **If** $M$ **does not accept** $w$, then $L(M_w)$ **is not regular**.
2. **If** $M$ **does accept** $w$, then $L(M_w)$ **is regular**.
3. The function $f(\langle M, w \rangle) = \langle M_w \rangle$ **is computable**.
REG$_{\text{TM}}$ is undecidable, the reduction

**Algorithm 18 ($S$)**

On input $\langle M, w \rangle$. Reject, if input is not well formatted.

1. Construct $M_w$.
2. Emulate $R$ on input $\langle M_w \rangle$ (recall that $R$ is a TM that decides REG$_{\text{TM}}$).
3. Accept, if $R$ accepts; Reject, if $R$ rejects.

**Claim 19**

$S$ decides $A_{\text{TM}}$.

A contradiction
\( \text{EQ}_{\text{TM}} \) is undecidable

\[ \text{EQ}_{\text{TM}} = \{ \langle M_1, M_2 \rangle : M_1, M_2 \text{ are TMs and } L(M_1) = L(M_2) \} \]

**Theorem 20**

\( \text{EQ}_{\text{TM}} \) is undecidable.

We are getting tired of reducing \( A_{\text{TM}} \) to everything. Let’s try instead a reduction from \( \text{EMPTY}_{\text{TM}} \) to \( \text{EQ}_{\text{TM}} \).

Proof’s idea:

- \( \text{EMPTY}_{\text{TM}} \) is the problem of testing whether a TM language is empty.
- \( \text{EQ}_{\text{TM}} \) is the problem of testing whether two TM languages are the same.
- If one of these two TM languages happens to be empty, then we are back to \( \text{EMPTY}_{\text{TM}} \).
- So \( \text{EMPTY}_{\text{TM}} \) is a special case of \( \text{EQ}_{\text{TM}} \).

The rest is easy.
EQ_{TM} is undecidable, 2

- EQ_{TM} = \{\langle M_1, M_2 \rangle: M_1, M_2 \text{ are TMs and } L(M_1) = L(M_2)\}
- Assume EQ_{TM} is decidable and let R be a TM deciding EQ_{TM}.

Proof:

**Algorithm 21 (M_{NO})**

On input x, reject

**Algorithm 22 (S)**

On input \langle M \rangle. Reject, if input is not well formatted.  
Emulate R on input \langle M, M_{NO} \rangle.  
If R accepts, accept; if R rejects, reject.

**Claim 23**

S decides EMPTY_{TM}.

A contradiction ♣
Bucket of undecidable problems

Similar techniques can be used to prove undecidability of

- Does a TM accept a **decidable** language?
- Does a TM accept a **context-free** language?
- Does a TM accept a **finite** language?
- Does a TM halt on **all inputs**?
- Is there an input string that causes a TM to **traverse all its states**?
Section 3

Mapping Reductions
Motivation

So far, we have seen many examples of reductions from one language to another, but the notion was neither defined nor treated formally.

Reductions play an important role in

- decidability theory (here and now)
- complexity theory (to come)

Time to get formal.
Mapping Reductions

**Definition 24**

A total computable function \( f : \Sigma^* \leftrightarrow \Sigma^* \) is a mapping reduction from language \( A \) to language \( B \), if \( x \in A \iff f(x) \in B \), for every \( x \in \Sigma^* \).

If a mapping reduction from \( A \) to \( B \) exists, we say that \( A \) is mapping reducible to \( B \), denoted by \( A \leq_m B \).

A mapping reduction converts questions about membership in \( A \) to membership in \( B \).

**Remark 25**

Note that \( A \leq_m B \iff \overline{A} \leq_m \overline{B} \).
Applications

**Theorem 26**

If \( A \leq_m B \) and \( B \) is decidable, then \( A \) is decidable.

**Proof:** Let \( M \) be the decider for \( B \), and \( f \) a (mapping) reduction from \( A \) to \( B \).

**Algorithm 27 \((N)\)**

On input \( x \)

1. Compute \( f(x) \)
2. Emulate \( M \) on input \( f(x) \). Accepts iff \( M \) does.

\[ \]

**Corollary 28**

If \( A \leq_m B \) and \( A \) is undecidable, then \( B \) is undecidable.

In fact, this has been our principal tool for proving undecidability of languages other than \( A_{TM} \).
H_{TM} is undecidable — Revisited

- \( A_{TM} = \{ \langle M, w \rangle : M \text{ is a TM that accepts } w \} \)
- \( H_{TM} = \{ \langle M, w \rangle : M \text{ is a TM and } M \text{ halts on input } w \} \)

Claim 29

\( A_{TM} \leq_m H_{TM} \)

The following computable function \( f \) establishes \( \langle M, w \rangle \in A_{TM} \iff f(\langle M, w \rangle) \in H_{TM}. \)

Definition 30 (f)

On input \( x = \langle M, w \rangle \). If \( x \) is not well formatted, return (some fixed) \( x' \notin H_{TM}. \)
Otherwise, return \( \langle \tilde{M}, w \rangle. \)

Definition 31 (TM \( \tilde{M} \))

On input \( y \): emulate \( M(y) \). Accept, if \( M \) accepts; enter a loop, if \( M \) rejects.
**H\textsubscript{TM},\epsilon** is undecidable

Recall that

- \( H\textsubscript{TM} = \{\langle M, w \rangle: M \text{ is a TM that halts on input } w \} \)
- \( H\textsubscript{TM},\epsilon = \{\langle M \rangle: M \text{ is a TM that halts on input } \epsilon \} \)

**Claim 32**

\[ H\textsubscript{TM} \leq_m H\textsubscript{TM},\epsilon \]

The following **computable** function \( f \) establishes

\[ \langle M, w \rangle \in H\textsubscript{TM} \iff f(\langle M, w \rangle) \in H\textsubscript{TM},\epsilon. \]

**Definition 33 (\( f \))**

On input \( x = \langle M, w \rangle \). If \( x \) is not well formatted, return \( x' \notin H\textsubscript{TM},\epsilon \). Otherwise, return \( \langle M_w \rangle \).

(Recall \( M_w \): on input \( x \), emulates \( M \) on \( w \)).

Note that \( M_w \) halts on \( \epsilon \iff M \) halts on \( w \).

Therefore, \[ \langle M \rangle \in H\textsubscript{TM},\epsilon \iff \langle M, w \rangle \in H\textsubscript{TM},\epsilon. \]
Mapping reducible relation is not symmetric

Claim 34

Let $A \in \mathcal{R}$ and $B \notin \mathcal{R}$, then $A \leq_m B$, but $B \not\leq_m A$.

Proof: It is clear that $B \not\leq_m A$ (?)

For proving $A \leq_m B$, define $f$ as follows:

Definition 35 ($f$)

On input $w$, return $x_Y \in B$ if $w \in A$, and $x_N \notin B$ otherwise.

- Do $x_Y$ and $X_N$ (always) exist?
- How can the TM implementing $f$ find $x_Y$ and $x_N$?
Enumerability

Theorem 36

If $A \leq_m B$ and $B \in \mathcal{RE}$, then $A \in \mathcal{RE}$.

Proof is same as in the decidability case, using accepters instead of deciders.

Corollary 37

If $A \leq_m B$ and $A \not\in \mathcal{RE}$, then $B \not\in \mathcal{RE}$.
TM equality, revisited

\[ \text{EQ}_{TM} = \{ \langle M_1, M_2 \rangle : M_1, M_2 \text{ are TMs and } L(M_1) = L(M_2) \} \]

Theorem 38

Both \( \text{EQ}_{TM} \) and, its complement, \( \text{EQ}_{TM}^c \), are not enumerable.

Stated differently, \( \text{EQ}_{TM} \) is neither enumerable nor co-enumerable, or \( \text{EQ}_{TM} \notin \mathcal{RE} \cup \text{co-RE} \).

- We show (next slide) that \( A_{TM} \leq_m \text{EQ}_{TM} \), and \( A_{TM} \leq_m \text{EQ}_{TM}^c \).
- Hence, \( \overline{A_{TM}} \leq_m \text{EQ}_{TM} \) and \( \overline{A_{TM}} \leq_m \text{EQ}_{TM} \) (?).
- Hence, neither \( \text{EQ}_{TM} \) nor \( \text{EQ}_{TM}^c \) are enumerable.
Claim 39
\( A_{TM} \leq_m EQ_{TM} \).

Definition 40 \((f)\)

On input \( x = \langle M, w \rangle \). If \( x \) is not well formatted, return \( x' \notin EQ_{TM} \).

1. Construct the TM \( M_w \) (recall that \( M_w \) emulates \( M \) on \( w \) [while ignoring its input]), and the TM \( M_{all} \) that accepts \( \Sigma^* \).
2. Return \( \langle M_w, M_{all} \rangle \).

- If \( M \) accepts \( w \), then \( M_w \) accepts everything. Otherwise, \( M_w \) accepts nothing.
- Hence, \( \langle M, w \rangle \in A_{TM} \iff \langle M_w, M_{all} \rangle \in EQ_{TM} \).
Claim 41

\[ A_{TM} \leq_m \overline{EQ_{TM}} \]

Definition 42 (f)

On input \( x = \langle M, w \rangle \). If \( x \) is not well formatted, return \( x' \notin EQ_{TM} \).

1. Construct the TM \( M_w \), and the TM \( M_{NO} \) that accepts \( \emptyset \).
2. Return \( \langle M_w, M_{NO} \rangle \).

- If \( M \) accepts \( w \), then \( M_w \) accepts everything. Otherwise, \( M_w \) accepts nothing.
- Hence, \( \langle M, w \rangle \in A_{TM} \iff \langle M_w, M_{NO} \rangle \in \overline{EQ_{TM}} \)

♣
Section 4

Busy Beaver
The Busy Beaver

(taken from http://www.saltine.org/joebeaver1.jpg)
**The Busy Beaver**

We focus on one tape TMs, with $\Sigma = \{0, 1\}$ and $\Gamma = \{0, 1, \omega\}$.

**Definition 43 ($S_n$ and $BB(n)$)**

For $n \in \mathbb{N}$, let

- $S_n = \{ \text{all } n\text{-state TM's that halt on } \varepsilon \}$.
- $BB(n) = \max_{M \in S_n} \{ \text{# of steps taken by } M \text{ on input } \varepsilon \}$.

- The set $S_n$ is finite (under standard encoding).
- Every $M \in S_n$ runs for finitely many steps on $\varepsilon$.
- $BB(n)$ is a total function from $\mathbb{N}$ to $\mathbb{N}$ (in particular, $BB(n) \in \mathbb{N}$ for every $n \in \mathbb{N}$).

**Values of $BB$ (size not including accept and reject states):**

<table>
<thead>
<tr>
<th>size</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>$BB$</td>
<td>6</td>
<td>21</td>
<td>107</td>
<td>$\geq 47,176,870$</td>
<td>$\geq 7.4 \times 10^{36534}$</td>
</tr>
</tbody>
</table>
The *Busy Beaver* function is not computable

**Theorem 44**

BB is *not* computable.

Proof: Assume BB is computable by the TM $R$ and consider the undecidable language $H_T M, \varepsilon = \{\langle M \rangle: M$ is a TM that halts on $\varepsilon\}$.

**Algorithm 45 (S)**

On input $\langle M \rangle$

1. Let $n$ be # of states in $M$, and let $m = R(n)$.
2. Emulate $M$ on $\varepsilon$ for $m + 1$ steps.
3. Accept if $M$ halts; otherwise reject

Note that if $M$ did not halt in $m + 1$ steps, then it will never halt!

**Claim 46**

S decides $H_T M, \varepsilon$. 
The bounded *Busy Beaver* function is computable

**Definition 47**

For \( d \in \mathbb{N} \), define the function \( \text{BB}_d : \mathbb{N} \to \mathbb{N} \) as

\[
\text{BB}_d(n) := \begin{cases} 
\text{BB}(n), & n \leq d, \\
0, & \text{otherwise}.
\end{cases}
\]

**Theorem 48**

*The function* \( \text{BB}_d \) *is computable for every* \( d \in \mathbb{N} \).

*Proof’s idea:* “Hardwire” the values \( \text{BB}(1) \ldots, \text{BB}(d) \) into a \( \text{TM} \) to compute \( \text{BB}_d \).
Section 5

Rice’s Theorem
Non-trivial properties of \( \mathcal{R}E \) languages

A few examples

- \( L \) is finite.
- \( L \) is infinite.
- \( L \) contains the empty string.
- \( L \) contains no prime number.
- \( L \) is co-finite.
- . . .

All these are non-trivial properties of \( \mathcal{R}E \) – for each of them there is \( L_1, L_2 \in \mathcal{R}E \) such that \( L_1 \) satisfies the property but \( L_2 \) does not.

Question 49

Are there any trivial properties of \( \mathcal{R}E \) languages?
Rice’s Theorem

**Theorem 50**

For non-empty $C \subseteq \mathcal{RE}$, it holds that

$L_C = \{\langle M \rangle : M \text{ is a TM and } L(M) \in C\} \notin \mathcal{R}$. 

Note that both $L_\emptyset$ and $L_{\mathcal{RE}}$ are in $\mathcal{R}$.

Proof’s idea: Mapping reduction from $H_{TM}$.

Given $M$ and $w$, we construct $M_w^C$ such that:

- If $M$ halts on $w$, then $\langle M_w^C \rangle \in L_C$.
- If $M$ does not halt on $w$, then $\langle M_w^C \rangle \notin L_C$. 

Iftach Haitner (TAU)  Computational Models, Lecture 9  December 11, 2018  41 / 49
Proving Rice’s Theorem

Assume $\emptyset \notin C$ (will take care of the other case later).

Fix $A \in C$ and let $M_A$ be a TM accepting it (recall $C \subseteq \mathcal{RE}$).

Algorithm 51 ($M^C_w$)

On input $y$:
1. Emulate $M(w)$.
2. Emulate $M_A(y)$:
   Accept, if $M_A$ accepts; reject, if $M_A$ rejects.

Let $f(\langle M, w \rangle) := \langle M^C_w \rangle$

Claim 52

$f$ is a mapping reduction from $H_{TM}$ to $L_C$

Since $f$ is clearly computable (?), it is left to show that

$$\langle M, w \rangle \in H_{TM} \iff f(\langle M, w \rangle) \in L_C$$
Proving $\langle M, w \rangle \in H_{TM} \iff f(\langle M, w \rangle) \in L_C$

Proof:

- If $\langle M, w \rangle \in H_{TM}$, then $M^C_w$ gets to Step 2, and emulates $M^A(y)$.
  Hence $L(M^C_w) = A \in C \implies M^C_w \in L_C$.

- Otherwise, $M^C_w$ never gets to Step 2.
  Hence $L(M^C_w) = \emptyset \notin C \implies M^C_w \notin L_C$.

- Thus, $\langle M, w \rangle \in H_{TM}$ iff $\langle M^C_w \rangle \in L_C$. 
Completing the proof: the case $\emptyset \in C$

Since $C \not\subset RE$ and $C$ is non-empty, it follows that $\overline{C} = RE \setminus C$ is non-empty, and $\emptyset \not\in \overline{C} \subset RE$.

$\implies$ (by the first part of the proof) $L_{\overline{C}} \notin R$

$\implies$ $\overline{L_{C}} \notin R$ (?)

($\overline{L_{C}} = L_{\overline{C}} \cup \{\langle M \rangle : M$ is not a TM$\}$)

$\implies$ $L_{C} \notin R$
Example: \( \text{Primes}_{\text{TM}} \notin \mathcal{R} \)

- \( \text{Primes} = \{ p \in \mathbb{N} : p \text{ is a prime} \} \)
- \( \text{Primes}_{\text{TM}} = \{ \langle M \rangle : M \text{ is a TM and } L(M) = \text{Primes} \} \)

Theorem 53

Primes \( \in \mathcal{R} \)

Proof: (?)

Theorem 54

\( \text{Primes}_{\text{TM}} \notin \mathcal{R} \).

Proof: \( L_{\text{Primes}} \) is a non trivial subset of \( \mathcal{RE} \).
Reflections on Rice theorem

- Rice’s theorem can be used to show **undecidability** of properties like
  - Does $L(M)$ contain infinitely many primes
  - Does $L(M)$ contain an arithmetic progression of length 15
  - Is $L(M)$ empty

- Decidability of properties related to the **encoding** itself cannot be inferred from Rice.
  - The question *does $\langle M \rangle$ has an even number of states* is decidable.
  - The question *does $M$ reaches state $q_6$ on the empty input string* is undecidable, but this does not follow from Rice’s theorem.

- Rice says **nothing** on membership in $\mathcal{RE}$

- Rice’s Theorem is a powerful tool, but use it with care!
Section 6

Proving Non-Enumerability
Proving non-enumerability

**Theorem 55**

For non-empty \( \mathcal{C} \subseteq \mathcal{R} \setminus \{ \Sigma^* \} \), it holds that

\[
L_C = \{ \langle M \rangle : \text{M is a TM and } L(M) \in \mathcal{C} \} \notin \mathcal{RE}.
\]

Corollary: Primes_{TM} \notin \mathcal{RE}.

Proof: We show that \( \overline{A\text{TM}} \leq_m L_C \).

Let \( D \) be a decider for some \( A \in \mathcal{C} \).

Define \( f(x = \langle M, w \rangle) \) to return \( B_{M,w} \) if \( x \) is well formatted, and \( D \) o/w.

**Definition 56 (B_{M,w})**

On input \( x \):

- Emulate \( D(x) \) and accept if \( D \) does.
- Emulate \( M(w) \) and accept if \( M \) does.

\[
\langle M, w \rangle \in \overline{A_{TM}} \implies L(B_{M,w}) = A \in \mathcal{C}
\]

\[
\langle M, w \rangle \notin \overline{A_{TM}} \implies L(B_{M,w}) = \Sigma^* \notin \mathcal{C}.
\]

What about All_{TM} := \( L_{\Sigma^*} \)?
Rice theorem vs. Thm 55

Rice speaks about non-membership in \( \mathcal{R} \) where Thm 55 about non-membership in \( \mathcal{RE} \).

But is it always the case that when Rice is applicable then so is Thm 55?

- Let \( \mathcal{C}_a = \{ L \in \mathcal{RE} : a \in L \} \).
- Hence, \( \mathcal{L}_{\mathcal{C}_a} = \{ \langle M \rangle : M \text{ is a TM and } L(M) \in \mathcal{C}_a \} \)
- \( \mathcal{L}_{\mathcal{C}_a} = \{ \langle M \rangle : M \text{ is a TM and } a \in L(M) \} \)
- By Rice, \( \mathcal{L}_{\mathcal{C}_a} \notin \mathcal{R} \)
- But clearly, \( \mathcal{L}_{\mathcal{C}_a} \in \mathcal{RE} \)
- Thm 55 is not applicable here since \( \mathcal{C}_a \notin \mathcal{R} \).