Computational Models - Lecture 8[1]

Handout Mode

Iftach Haitner and Yishay Mansour.

Tel Aviv University.

April 7 / 23 , 2014

[1] Based on slides by Benny Chor, Tel Aviv University, modifying slides by Maurice Herlihy, Brown University.

Talk Outline

- Mapping Reductions
- Undecidability by Rice Theorem

- Sipser's book, 5.1–5.3

Section 1

Mapping Reductions

Reminder

We have already

- Established Turing Machines as the gold standard of computers and computability ...
- seen examples of solvable problems ...
- and saw one problem, $\mathrm{A_{TM}}$, that is computationally unsolvable.

$$\mathrm{A_{TM}} = \{\langle M, w\rangle | M \text{ is a TM that accepts } w\}$$

Today, we look at other computationally unsolvable problems via reductions and introduce the techniques of mapping reductions.

Computable Functions

Definition 1 (total computable functions)

A TM M computes a total function $f : \Sigma^* \longrightarrow \Sigma^*$, 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.

Definition 2 (partially computable functions)

A TM M computes a partial function $f : \Sigma^* \longrightarrow (\Sigma^* \cup \perp)$, if when starting with an input w:

- if $f(w)$ is defined (i.e., $\neq \perp$), M halts with only $f(w)$ on tape,
- if $f(w)$ is undefined, M does not halt.

Computable functions are also called (total or partial) recursive functions.

Example, 1

Claim 3

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 mathematatic techniques.

Exercise 4

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 5

On input w:

If $w = \langle M \rangle$ for some TM, construct $\langle M' \rangle$, where

- $\mathrm{L}(M') = \mathrm{L}(M)$, and
- M' does not halt for $w \notin \mathrm{L}(M')$

Question 6

Is the function defined above total? computable?

Example, 3

- Given $M = (Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$ build $M' = (Q', \Sigma, \Gamma, \delta', q_0, q_a, q_r)$
- Let $Q' = Q \cup \{q^*\}$
- Define δ' as follows:

$$\delta'(q,\sigma) := \begin{cases} (q^*, \sigma, R), & \text{if } \delta(q,\sigma) = (q_r, \cdot, \cdot) \\ (q^*, \sigma, R) & q = q^* \\ (q', \sigma', D), & \text{otherwise if } \delta(q,\sigma) = (q', \sigma', D) \end{cases}$$

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 problems, A and B.

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

Remark 7

This property says nothing about solving A by itself or B by itself.

but

Fact 8

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 serval problems.

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 9

H_{TM} *is undecidable.*

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

Algorithm 10 (S)

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

1. Emulate R on $\langle M, w\rangle$.
2. If R rejects, reject.
3. If R accepts (meaning M halts on w), emulate M on w until it halts (namely run U on $\langle M, w\rangle$).
4. If M accepted, accept; otherwise reject.

TM S decides A_{TM}, a contradiction ♣

What we actually did is a reduction from A_{TM} to H_{TM}.

$\mathrm{EMPTY_{TM}}$ is Undecidable

$$\mathrm{EMPTY_{TM}} = \{\langle M \rangle | M \text{ is a TM and } \mathrm{L}(M) = \emptyset\}$$

Theorem 11

$\mathrm{EMPTY_{TM}}$ *is undecidable.*

Proof's idea: By contradiction:

- Assume $\mathrm{EMPTY_{TM}}$ is decidable and let R be a TM that decides $\mathrm{EMPTY_{TM}}$.
- Use R to construct S, a TM that decides $\mathrm{A_{TM}}$.

Algorithm 12 (S – first attempt)

On input $\langle M, w \rangle$:
Emulate $R(\langle M \rangle)$ and reject if R accepts.

But what if R rejects?

EMPTY_{TM} is Undecidable, 2

Solution? Modify M

Definition 13 (M_w)

Given a TM M and input w, define the TM M_w as follows:
On input x,

1. if $x \neq w$, reject.
2. if $x = w$, run M on w and accept if M does.

M_w either

- accepts just w (in case M accepts w), or
- accepts nothing (otherwise).

EMPTY_{TM} is Undecidable, 3

Definition 14

The language of M_w:

$$L(M_w) := \begin{cases} \{w\}, & \text{M accepts w,} \\ \emptyset, & \text{M does not accept w} \end{cases}$$

Question 15

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 perform the "$x = w?$" test.

EMPTY_{TM} is Undecidable, building M_w

Let $w = w_1, \ldots w_n$, assume $n \geq 3$.

- Given $M = (Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$ and w build $M_w = (Q', \Sigma, \Gamma, \delta', \overrightarrow{q}_1, q_a, q_r)$
- Let $Q' = Q \cup \{\overrightarrow{q}_1, \ldots, \overrightarrow{q}_{n+1}, \overleftarrow{q}_2, \ldots, \overleftarrow{q}_n\}$
- Define δ' as follows:

$$\delta'(q,\sigma) := \begin{cases} (q_r, \sigma, R) & q = \overrightarrow{q}_i \text{ and } \sigma \neq w_i, \text{ for } i \leq n \\ (\overrightarrow{q}_{i+1}, \sigma, R) & q = \overrightarrow{q}_i \text{ and } \sigma = w_i, \text{ for } i \leq n \\ (q_r, \sigma, R) & q = \overrightarrow{q}_{n+1} \text{ and } \sigma \neq \sqcup, \\ (\overleftarrow{q}_n, \sigma, L) & q = \overrightarrow{q}_{n+1} \text{ and } \sigma = \sqcup, \\ (\overleftarrow{q}_{i-1}, \sigma, L) & q = \overleftarrow{q}_i \text{ and } \sigma = w_i, \text{ for } i \geq 3 \\ (q_0, \sigma, L) & q = \overleftarrow{q}_2 \text{ and } \sigma = w_2, \\ (q', \sigma', D), & \text{otherwise if } \delta(q,\sigma) = (q', \sigma', D) \end{cases}$$

EMPTY_{TM} is Undecidable, 4

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

Algorithm 16 (S)

On input $\langle M, w \rangle$

1. Construct M_w from M and w.
2. Emulate R on input $\langle M_w \rangle$.
3. If R accepts, reject; if R rejects, accept.

Claim 17

S decides A_{TM}.

A contradiction. ♣

$\mathrm{REG}_{\mathrm{TM}}$ is Undecidable

$$\mathrm{REG}_{\mathrm{TM}} = \{\langle M\rangle | M \text{ is a TM and } L(M) \text{ is regular}\}$$

Theorem 18

$\mathrm{REG}_{\mathrm{TM}}$ *is undecidable.*

Proof's idea: By contradiction.

- Assume $\mathrm{REG}_{\mathrm{TM}}$ is decidable and let R be a TM that decides $\mathrm{REG}_{\mathrm{TM}}$.
- Use R to construct S – a TM that decides A_{TM}.

Question 19

But how we construct S?

Intuition: On input $\langle M, w\rangle$, build M_w that accepts regular language iff M accepts w.

TM M_w

- if M does not accept w, then M_w accepts the (non-regular) language $\{0^n 1^n | n \geq 0\}$
- if M accepts w, then M_w accepts the (regular) language Σ^* .

Algorithm 20 (M_w)

On input x,

1. If x has the form $0^n 1^n$, accept it.
2. Otherwise, emulate M on input w and accept x if M accepts w.

Claim 21

1. If M does not accept w, then M_w accepts (the language) $\{0^n 1^n | n \geq 0\}$.
2. If M accepts w, then M_w accepts (the language) Σ^*.
3. The function: **on input** $\langle M, w \rangle$ **output** $\langle M_w \rangle$, is computable.

TM S

Algorithm 22 (S)

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

1. Construct M_w from M and w.
2. Emulate R on input $\langle M_w \rangle$, where R be a TM that decides $\mathrm{REG_{TM}}$.
3. If R accepts, accept; if R rejects, reject.

Claim 23

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

A contradiction ♣

$\mathsf{EQ_{TM}}$ is Undecidable

$$\mathsf{EQ_{TM}} = \{\langle M_1, M_2\rangle : M_1, M_2 \text{ are TMs and } \mathsf{L}(M_1) = \mathsf{L}(M_2)\}$$

Theorem 24

$\mathsf{EQ_{TM}}$ *is undecidable.*

We are getting tired of reducing $\mathsf{A_{TM}}$ to everything.
Let's try instead a reduction from $\mathsf{EMPTY_{TM}}$ to $\mathsf{EQ_{TM}}$.

Proof's idea:

- $\mathsf{EMPTY_{TM}}$ is the problem of testing whether a TM language is empty.
- $\mathsf{EQ_{TM}}$ 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 $\mathsf{EMPTY_{TM}}$.
- So $\mathsf{EMPTY_{TM}}$ is a special case of $\mathsf{EQ_{TM}}$.

The rest is easy.

$\mathrm{EQ_{TM}}$ is Undecidable, 2

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

Proof:

Algorithm 25 ($\mathrm{M_{NO}}$)

On input x, reject

Algorithm 26 (S)

On input $\langle M\rangle$:
Emulate R on input $\langle M, \mathrm{M_{NO}}\rangle$.
If R accepts, accept; if R rejects, reject.

Claim 27

S decides $\mathrm{EMPTY_{TM}}$.

A contradiction ♣

Bucket of Undecidable Problems

Same techniques 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?

The Busy Beaver

(taken from `http://www.saltine.org/joebeaver1.jpg`)

The Busy Beaver, 2

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

Definition 28 (S_n and $BB(n)$)

For $n \in \mathbb{N}$, let $S_n = \{$all n-state TM's that halt on $\varepsilon\}$.
Let $\mathrm{BB}(n)$ be maximum $\#$ of steps taken be some $M \in S_n$ on input ε.

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

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

size	2	3	4	5	6
BB	6	21	107	$\geq 47,176,870$	$\geq 7.4 \times 10^{36534}$

The Busy Beaver Function is Not Computable

Theorem 29

The busy beaver function is not computable.

Proof: Consider the undecidable language (proved latter)
$\mathrm{H}_{\mathrm{TM},\varepsilon} = \{\langle M\rangle | M$ is a TM and M halts on $\varepsilon\}$
BB is computable using R.

Algorithm 30 (S)

On input $\langle M\rangle$

1. Compute m the number of states in M, and compute $n = R(m)$.
2. Emulate M on ε for $n+1$ steps.
3. If M halts then accept otherwise reject

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

Claim 31

S decides $\mathrm{H}_{\mathrm{TM},\varepsilon}$.

The Bounded Busy Beaver Function Is Computable

Definition 32

For $d \in \mathbb{N}$, define the function $\mathrm{BB}_d : \mathbb{N} \mapsto \mathbb{N}$ as

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

Theorem 33

The function BB_d is computable for every $d \in \mathbb{N}$.

Proof's idea: "Hardwire" the values $\mathrm{BB}(1) \ldots, \mathrm{BB}(d)$ into a TM to compute BB_d.

Reducibility

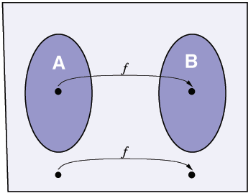
So far, we have seen many examples of reductions from one language to another, but the notion was neither defined nor treated formally.

Reductions play an important role in

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

Time to get formal.

Mapping Reductions



Definition 34

A computable function $f : \Sigma^* \longrightarrow \Sigma^*$ is a reduction from language A to language B, if $w \in A \iff f(w) \in B$, for every $w \in \Sigma^*$.

If a 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 35

Note that $A \leq_m B \iff \overline{A} \leq_m \overline{B}$

Applications

Theorem 36

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 37 (N)

On input w

1. Compute $f(w)$
2. Emulate M on input $f(w)$ and output whatever M outputs.

♣

Corollary 38

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}

$\mathrm{H_{TM}}$ is Undecidable – Revisited

Recall that

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

Earlier we proved that $\mathrm{H_{TM}}$ undecidable by (de facto) reduction from $\mathrm{A_{TM}}$. Let's reformulate this.

Claim 39

$\mathrm{A_{TM}} \leq_m \mathrm{H_{TM}}$

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

Definition 40 (f)

On input $\langle M, w\rangle$, return $\langle M^{\ell}, w\rangle$.
TM M^{ℓ} is defined as follows:
On input x, emulate $M(x)$ and

accepts if M accepts; enters a loop if M rejects.

Building M^ℓ

We actually already saw this,

- Given $M = (Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$ build $M^\ell = (Q', \Sigma, \Gamma, \delta', q_0, q_a, q_r)$
- Let $Q' = Q \cup \{q^*\}$
- Define δ' as follows:

$$\delta'(q, \sigma) := \begin{cases} (q^*, \sigma, R), & \text{if } \delta(q, \sigma) = (q_r, \cdot, \cdot) \\ (q^*, \sigma, R) & q = q^* \\ (q', \sigma', D), & \text{otherwise if } \delta(q, \sigma) = (q', \sigma', D) \end{cases}$$

Note that $L(M^\ell) = L(M)$ and $\langle M, w \rangle \in \mathrm{A_{TM}} \iff f(\langle M, w \rangle) \in \mathrm{H_{TM}}$

$\mathrm{H}_{\mathrm{TM},\varepsilon}$ is Undecidable

Recall that

- $\mathrm{H}_{\mathrm{TM}} = \{\langle M, w\rangle | M$ is a TM and M halts on input $w\}$
- $\mathrm{H}_{\mathrm{TM},\varepsilon} = \{\langle M\rangle | M$ is a TM and M halts on input $\epsilon\}$

Claim 41

$\mathrm{H}_{\mathrm{TM}} \leq_m \mathrm{H}_{\mathrm{TM},\varepsilon}$

The following computable function f establishes $\langle M, w\rangle \in \mathrm{H}_{\mathrm{TM}} \iff f(\langle M, w\rangle) \in \mathrm{H}_{\mathrm{TM},\varepsilon}$.

Definition 42 (f)

On input $\langle M, w\rangle$, return $\langle M_w^e\rangle$.
TM M_w^e is defined as follows:
On input x, erase x and write w and emulate $M(w)$.

Building M_w^e

- Given $M = (Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$ and $w = w_1, \ldots w_n$, build $M_w^e = (Q', \Sigma, \Gamma, \delta', \overrightarrow{q}_1, q_a, q_r)$, assume $n \geq 1$.
- Let $Q' = Q \cup \{\overrightarrow{q}_1, \ldots \overrightarrow{q}_{n+1}, \overleftarrow{q}, \overleftarrow{q}_1\}$
- Define δ' as follows:

$$\delta'(q,\sigma) := \begin{cases} (\overrightarrow{q}_2, \$, R) & q = \overrightarrow{q}_1 \\ (\overrightarrow{q}_{i+1}, w_i, R) & q = \overrightarrow{q}_i, 2 \leq i \leq n \\ (\overrightarrow{q}_{n+1}, \sqcup, R) & q = \overrightarrow{q}_{n+1}, \sigma \neq \sqcup \\ (\overleftarrow{q}, \sigma, L) & q = \overrightarrow{q}_{n+1}, \sigma = \sqcup \\ (\overleftarrow{q}, \sigma, L) & q = \overleftarrow{q}, \sigma \neq \$ \\ (\overleftarrow{q}_1, w_1, R) & q = \overleftarrow{q}, \sigma = \$ \\ (q_0, \sigma, L) & q = \overleftarrow{q}_1 \\ (q', \sigma', D), & \text{otherwise if } \delta(q,\sigma) = (q', \sigma', D) \end{cases}$$

$\mathrm{H}_{\mathrm{TM},\varepsilon}$ is Undecidable, concluding

For the reduction: M_w^e halts on $\varepsilon \iff M$ halts on w.

Therefore, $\langle M \rangle \in \mathrm{H}_{\mathrm{TM},\varepsilon} \quad \Leftrightarrow \langle M, w \rangle \in \mathrm{H}_{\mathrm{TM}}$.

The language of M_w^e:

$$L(M_w^e) = \begin{cases} \Sigma^* & \text{if } M \text{ halts and accepts } w \\ \emptyset & \text{if } M \text{ halts and rejects } w \\ \emptyset & \text{if } M \text{ does not halts on } w \end{cases}$$

The Mapping Reducible Relation is Not Symmetric

$\mathrm{H}_{\mathrm{TM},\varepsilon} = \{\langle M \rangle | M$ is a TM and M halts on $\varepsilon\}$

Claim 43

Let $L = \{0^n : n \in \mathbb{N}\}$. Then $L \leq_m \mathrm{H}_{\mathrm{TM},\varepsilon}$, but $\mathrm{H}_{\mathrm{TM},\varepsilon} \not\leq_m L$

Proof: It is clear that $\mathrm{H}_{\mathrm{TM},\varepsilon} \not\leq_m L$ (why?).
For proving $L \leq_m \mathrm{H}_{\mathrm{TM},\varepsilon}$, define f as follows:

Definition 44 (f)

On input w. Return $\langle M_H \rangle$ if $w \in L$, and return $\langle M_L \rangle$ otherwise.
Where M_H halts on ε, and M_L loops on ε.

♣

Enumerability

Theorem 45

If $A \leq_m B$ and B is enumerable, then A is enumerable.

Proof is same as before, using accepters instead of deciders.

Corollary 46

If $A \leq_m B$ and A is not enumerable, then B is not enumerable.

TM Equality

$$\mathrm{EQ_{TM}} = \{\langle M_1, M_2\rangle : M_1, M_2 \text{ are TMs and } \mathrm{L}(M_1) = \mathrm{L}(M_2)\}$$

Theorem 47

Both $\mathrm{EQ_{TM}}$ and, its complement, $\overline{\mathrm{EQ_{TM}}}$, are not enumerable.

Stated differently, $\mathrm{EQ_{TM}}$ is neither enumerable nor co-enumerable, or $\mathrm{EQ_{TM}} \notin \mathcal{RE} \cup \textbf{co}-\mathcal{RE}$.

We show that

- $\mathrm{A_{TM}} \leq_m \mathrm{EQ_{TM}}$, and $\mathrm{A_{TM}} \leq_m \overline{\mathrm{EQ_{TM}}}$.
- It follows that $\overline{\mathrm{A_{TM}}} \leq_m \overline{\mathrm{EQ_{TM}}}$ and $\overline{\mathrm{A_{TM}}} \leq_m \mathrm{EQ_{TM}}$ (why?)
- Hence, neither $\mathrm{EQ_{TM}}$ nor $\overline{\mathrm{EQ_{TM}}}$ are enumerable.

$A_{TM} \leq_m EQ_{TM}$

Claim 48

$A_{TM} \leq_m EQ_{TM}$.

Definition 49 (f)

On input $\langle M, w \rangle$

1. Construct a machine M_1 that accepts Σ^*
2. Construct a machine M_2 $(= M_w^e)$ that accepts x, if $M(w)$ accepts.
3. Return $\langle M_1, M_2 \rangle$.

Note

- if M accepts w, then M_2 accepts everything. Otherwise, M_2 accepts nothing.
- Hence, $\langle M, w \rangle \in A_{TM} \iff \langle M_1, M_2 \rangle \in EQ_{TM}$.

♣

$A_{TM} \leq_m \overline{EQ_{TM}}$

Claim 50

$A_{TM} \leq_m \overline{EQ_{TM}}$.

Definition 51 (f)

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

1. Construct a machine M_1 that accepts $\emptyset$.
2. Construct a machine M_2 $(= M_w^e)$ that accepts x, if $M(w)$ accepts.
3. Return $\langle M_1, M_2 \rangle$.

Note

- If M accepts w then M_2 accepts everything. Otherwise, M_2 accepts nothing.
- Hence, $\langle M, w \rangle \in A_{TM} \iff \langle M_1, M_2 \rangle \in \overline{EQ_{TM}}$

♣

TM Equality, Summary

- Since $\mathrm{A_{TM}} \leq_m \mathrm{EQ_{TM}}$, then $\overline{\mathrm{A_{TM}}} \leq_m \overline{\mathrm{EQ_{TM}}}$
- Since $\mathrm{A_{TM}} \leq_m \overline{\mathrm{EQ_{TM}}}$ then $\overline{\mathrm{A_{TM}}} \leq_m \mathrm{EQ_{TM}}$
- Hence, neither $\mathrm{EQ_{TM}}$ nor $\overline{\mathrm{EQ_{TM}}}$ are enumerable.
- Stated differently, $\mathrm{EQ_{TM}}$ is neither enumerable nor co-enumerable, or $\mathrm{EQ_{TM}} \notin \mathcal{RE} \cup \textbf{co}{-}\mathcal{RE}$.

Section 2

Rice's Theorem

Non Trivial Properties of $\mathcal{RE}$ 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 enumerable languages, since for each of them there is $L_1, L_2 \in \mathcal{RE}$ such that L_1 satisfies the property but L_2 does not.

Question 52

Are there any trivial properties of $\mathcal{RE}$ languages?

Theorem 53

Let $\mathcal{C}$ be a proper non-empty subset of the set of $\mathcal{RE}$ and let $\mathsf{L}_{\mathcal{C}} = \{\langle M \rangle : \mathsf{L}(M) \in \mathcal{C}\}$. Then $\mathsf{L}_{\mathcal{C}}$ is undecidable.

Proof's idea: Reduction from H_{TM}.

Given M and w, we construct M_0 such that:

- If M halts on w, then $\langle M_0 \rangle \in \mathsf{L}_{\mathcal{C}}$.
- If M does not halt on w, then $\langle M_0 \rangle \notin \mathsf{L}_{\mathcal{C}}$.

Proving Rice's Theorem

We assume wlg. that $\emptyset \notin \mathcal{C}$ (otherwise, look at $\overline{\mathcal{C}}$, also proper and non-empty).
Fix $\mathrm{L} \in \mathcal{C}$ and let M_{L} be a TM accepting it (recall $\mathcal{C} \subseteq \mathcal{RE}$).

Algorithm 54 (M_0)

On input y:

1. Emulate $M(w)$.
2. Emulate $M_{\mathrm{L}}(y)$:
   Accept if M_{L} accepts; reject if M_{L} rejects.

Let $f(\langle M, w \rangle) := \langle M_0 \rangle$, and let $f(x) = \emptyset$ if x is not of the form $\langle M, w \rangle$.

Claim 55

f is a mapping reduction from H_{TM} to $\mathrm{L}_{\mathcal{C}}$

f is Computable

Claim 56

f is computable.

Proof: On a valid pair $\langle M, w\rangle$, the TM $M_0 = f(\langle M, w\rangle)$ is simply a concatenation of two known TMs: the universal machine and M_{L}. ♣

$\langle M, w\rangle \in \mathrm{H_{TM}} \iff f(\langle M, w\rangle) \in \mathrm{L}_{\mathcal{C}}$

Claim 57

$\langle M, w\rangle \in \mathrm{H_{TM}} \iff f(\langle M, w\rangle) \in \mathrm{L}_{\mathcal{C}}$

(Hence, f is a mapping reduction from $\mathrm{H_{TM}}$ to $\mathrm{L}_{\mathcal{C}}$)

Proof:

- If $\langle M, w\rangle \in \mathrm{H_{TM}}$, then M_0 gets to Step 2, and emulates $M_{\mathrm{L}}(y)$. Hence $\mathrm{L}(M_0) = \mathrm{L} \in \mathcal{C}$.
- Otherwise (i.e., $\langle M, w\rangle \notin \mathrm{H_{TM}}$), M_0 never gets to Step 2. Hence $\mathrm{L}(M_0) = \emptyset \notin \mathcal{C}$.
- Thus, $\langle M, w\rangle \in \mathrm{H_{TM}}$ iff $\langle M_0\rangle \in \mathrm{L}_{\mathcal{C}}$.

♣

We proved that $\mathrm{H_{TM}} \leq_m L_{\mathcal{C}}$, thus $\mathrm{L}_{\mathcal{C}}$ is undecidable.

Reflections

- Rice's theorem can be used to show undecidability of properties like
  - *Does* $\mathrm{L}(M)$ *contain infinitely many primes*
  - *Does* $\mathrm{L}(M)$ *contain an arithmetic progression of length 15*
  - *Is* $\mathrm{L}(M)$ *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 does not say anything on membership in $\mathcal{RE}$ of questions like *is* $\mathrm{L}(M)$ *finite*.
- **Rice's Theorem is a powerful tool, but use it with care!**