24. Post's Correspondence Problem and Rice's Theorem

Goal: Introduce two big undecidability results that rely on more sophisticated reductions.

24.1 Post's Correspondence Problem (PCP)

PCP:

Given: A sequence of pairs of words $(x_1, y_1) \dots (x_k, y_k)$.

Question: Is there a sequence of indices $i_1 \dots i_n$ (non-empty) with

$$x_{i_1} \dots x_{i_n} = y_{i_1} \dots y_{i_n} ?$$

Example:

Consider the instance $V = \underbrace{(1, 101)}_{1}, \underbrace{(10, 00)}_{2}, \underbrace{(011, 11)}_{3}$.

A solution is 1323, — indices.

because

$$1.011.10.011 = 101.11.00.11.$$

To reduce the halting problem, we define a modified version of PCP.

MPCP:

Given: A sequence of pairs of words $(x_1, y_1) \dots (x_k, y_k)$.

Question: Is there a non-empty sequence of indices $i_1 \dots i_n$ with

$$x_{i_1} \dots x_{i_n} = y_{i_1} \dots y_{i_n} \quad \text{and} \quad i_1 = 1.$$

Lemma: MPCP $\leq$ PCP.

Proof:

Let $\$$ and $\#$ be symbols that do not occur in the alphabet Σ of the given MPCP instance.

We define three variants of a given word $w = a_1 \dots a_m \in \Sigma^*$:

$$\overline{w} := \#a_1\#a_2\#\dots\#a_m\#$$

$$\grave{w} := \#a_1\#a_2\#\dots\#a_m$$

$$\acute{w} := a_1\#a_2\#\dots\#a_m\#.$$

Given an instance of MPCP

$$K = (x_1, y_1)\dots(x_k, y_k),$$

we construct

$$f(K) := (\overline{x_1}, \grave{y_1})(\acute{x_1}, \grave{y_1})(\acute{x_2}, \grave{y_2})\dots(\acute{x_k}, \grave{y_k})(\$, \#\$).$$

Function f is computable.

We show that it satisfies

K has a solution (with $i_1 = 1$) iff $f(K)$ has a solution (at all).

"⇒" If K has the solution $i_1 \dots i_n$ with $i_1 = 1$,
then the following is a solution for $f(K)$:

$$1,\ i_2+1,\ i_3+1, \dots, i_n+1,\ k+2.$$

"⇐" If $f(K)$ has a solution $i_1 \dots i_n \in \{1, \dots, k+2\}^*$,
it has a shortest such solution.

Then by construction we can only have

- $i_1 = 1$ // Since no other pair has $\#, \#$ leftmost.
- $i_n = k+2$ // Since no other pair has matching rightmost symbols.

Because the solution is the shortest one,

- 1 and $k+2$ do not occur inside the sequence,
- and hence $i_2, \dots, i_{n-1} \in \{2, \dots, k+1\}$.

Then $1, i_2 - 1, \dots, i_{n-1} - 1$ is a solution for K. □

<u>Proposition</u>: $HP \leq MPCP$.

<u>Proof</u>:

Consider the input to HP

$$w \# x$$

where w encodes $M_w = (Q, \Sigma, \Gamma, q_0, \sqcup, \delta, Q_F)$ and $x \in \Sigma^*$.

- Our task is to define a function that turns the pair into a sequence

$$(x_1, y_1) \ldots (x_k, y_k)$$

so that

$$M_w \text{ accepts } x \quad (x \in L(M_w))$$

iff $(x_1, y_1) \ldots (x_k, y_k)$ has a solution with $i_1 = 1$.

- Turing machine M_w accepts x iff there is a sequence of configurations

$$cf_0 \to cf_1 \to \ldots \to cf_t$$

with $cf_0 = q_0 x$ and $cf_t = u q_f v$.

We will make sure the MPCP instance has a solution of the form

$$\# cf_0 \# cf_1 \# \ldots \# cf_t \# cf_t' \# cf_t'' \# \ldots \# q_f \#\#.$$

So we encode in MPCP the <u>sequence of configurations</u>, the <u>computation</u> of the TM. (plus some extra ones)

- The alphabet of the MPCP instance is

$$\Gamma \cup Q \cup \{\#\}.$$

The initial pair is $(\#, \# q_0 x \#)$. // Initial configuration.

$\boxed{\text{The trick is to have the solution on } x \text{ fall behind by one configuration.}}$

This allows us to properly copy the current configuration.

In the following illustration, the numbers $\overbrace{}^{i}$ indicate when an element is added:

$$\text{x-sequence: } \# \overbrace{q_0 a_1}^{1} \overbrace{a_2}^{2} \dots \overbrace{a_n}^{n} \overbrace{\#}^{n+1}$$

$$\text{y-sequence: } \underbrace{\# q_0 a_1 a_2 \dots a_n \#}_{\text{initially given}} \underbrace{q_1 b}_{1} \underbrace{a_2}_{2} \dots \underbrace{a_n}_{n} \underbrace{\#}_{n+1}$$

We need the following pairs in the PCP instance:

1.) Copy-Rule: (a, a) for all $a \in \Gamma \cup \{\#\}$.

2.) Transition-Rules:

$(qa, q'b)$,	if $(q, a, b, N, q') \in \delta$
(qa, bq'),	if $(q, a, b, R, q') \in \delta$
$(cqa, q'cb)$,	if $(q, a, b, L, q') \in \delta$, for all $c \in \Gamma$
$(\#qa, \#q'\sqcup b)$,	if $(q, a, b, L, q') \in \delta$
$(q\#, q'b\#)$,	if $(q, \sqcup, b, N, q') \in \delta$
$(q\#, bq'\#)$,	if $(q, \sqcup, b, R, q') \in \delta$
$(cq\#, q'cb\#)$,	if $(q, \sqcup, b, L, q') \in \delta$, for all $c \in \Gamma$.

3.) Deletion-Rules:

(aq_f, q_f), $(q_f a, q_f)$ for all $a \in \Gamma$, $q_f \in Q_F$.

4.) Final-Rules:

$(q_f \#\#, \#)$ for all $q_f \in Q_F$.

One can show that if M_w accepts input x, we obtain a solution to the MPCP instance. In turn, a solution to the MPCP instance is an accepting computation of M_w on x. □

Theorem (Post '46):

PCP is undecidable.

Actually, we can restrict PCP even more, namely to the alphabet $0,1$, called $0,1$-PCP.

Lemma: $PCP \leq 0,1\text{-}PCP$.

Proof:

Let $\Sigma = \{a_1, \dots, a_n\}$ be the alphabet of the given PCP instance.

We define the homomorphism

$$h : \Sigma^* \longrightarrow \{0,1\}^* \quad \text{by}$$

$$a_j \longmapsto 0\,1^j.$$

With this, $(x_1, y_1) \dots (x_k, y_k)$ has a solution iff $(h(x_1), h(y_1)) \dots (h(x_k), h(y_k))$ has a solution. □

Theorem: $0,1$-PCP is undecidable.

Remark:

Define PCP_k to be the set of PCP instances with k pairs. It is known that PCP_9 is undecidable and PCP_2 is decidable. The problems PCP_3 to PCP_8 are open.

24.2 Rice's Theorem

<u>Goal</u>: Show a general undecidability result:

Every non-trivial property about the behavior (languages) of Turing machines is undecidable.

Phrased differently, undecidability is the rule not the exception.

<u>Definition</u>:

Let $RE(\Sigma^*)$ be the class of recursively-enumerable subsets of Σ^* (the languages of Turing machines).

- A <u>property</u> is a function
$$P: RE(\Sigma^*) \to \{0,1\}.$$
- A property P is <u>trivial</u>, if $P(L)=0$ or $P(L)=1$ for all $L \in RE(\Sigma^*)$.

Otherwise, it is called <u>non-trivial</u>.

<u>Note</u>:

- To ask whether a property P is decidable, the language L has to be represented in a finite form that can be given as an input to a decision procedure / an algorithm.

We assume that L is given by a TM M with $L = L(M)$.

So algorithmically, property P is the set
$$P := \{ w \in \{0,1\}^* \mid P(L(M_w)) = 1 \}.$$

<u>Deciding the property</u> means deciding this set.

- But note that a property is a property about L, not about M_w.

The property has to be <u>independent of the TM</u> that accepts L:

Either $P(L(M_w))=1$ for <u>all</u> $w \in \{0,1\}^*$ with $L(M_w)=L$

or $P(L(M_w))=0$ for <u>all</u> $w \in \{0,1\}^*$ with $L(M_w)=L$.

Example:

Non-trivial properties of recursively-enumerable sets:

$L = L(M_w)$ is finite,

$L = L(M_w)$ is regular,

L is context-free,

$10110 \in L$ (M_w accepts 10110)

$L = \Sigma^*$.

The following are properties of Turing machines that are not properties of recursively-enumerable sets:

M_w has 481 states,

M_w has a rejecting computation on 10110,

there is a smaller TM that accepts the language.

These are not properties of recursively-enumerable sets, because in each case

- one can give a TM M with $L = L(M)$ and M has the property and
- one can give another TM M' with $L = L(M')$ and M' does not have the property.

Theorem (Rice '53):
Every non-trivial property of the recursively-enumerable sets is undecidable.

Proof:
Let P be a non-trivial property of the recursively-enumerable sets.

- Assume wlog. that $P(\emptyset) = 0$, for 1 the argument is symmetric. Since P is non-trivial, there is a recursively-enumerable set L with $P(L) = 1$.

  Let K be a TM that accepts L, so $L = L(K)$.

- We reduce the halting-problem to the set

$$P = \{ w \in \{0,1\}^* \mid P(L(M_w)) = 1 \},$$

which means problem P is undecidable.

- Given $w \# x$ representing M_w on input x, we construct a machine $M_{w,x}^{K}$ that, on input y, does the following:

  (1) save y somewhere (separate track)

  (2) write x to the tape (x is hard-wired in $M_{w,x}^{K}$)

  (3) run M_w on x

  (4) if M_w accepts x (we can assume accept $\Leftrightarrow$ halt)

  $M_{w,x}^{K}$ runs K on input y.

  $M_{w,x}^{K}$ accepts iff K accepts y.

Now M_w halts on x or it does not halt.

Case (1): M_w halts on x

Then $M_{w,x}^{K}$ reaches (4) and runs K on y.

Now $y \in L(M_{w,x}^{K})$ iff $y \in L(K)$ iff $y \in L$.

Case (2): M_w does not halt on x

Then $M_{w,x}^{K}$ does not reach (4).

Then $M_{w,x}^K$ does not accept y,

no matter what was y.

Hence, $L(M_{w,x}^K) = \emptyset$.

Summary:

M_w halts on $x \Rightarrow L(M_{w,x}^K) = L$.

M_w does not halt on $x \Rightarrow L(M_{w,x}^K) = \emptyset$.

Hence,

M_w halts on $x \Rightarrow P(L(M_{w,x}^K)) = P(L) = 1$.

M_w does not halt on $x \Rightarrow P(L(M_{w,x}^K)) = P(\emptyset) = 0$.

$\square$

There is another version of Rice's theorem.

A property P is called monotone,

if for all $L_1, L_2 \in RE(\Sigma^*)$ we have

$$L_1 \subseteq L_2 \quad \Rightarrow \quad P(L_1) \leq P(L_2).$$

Otherwise, the property is non-monotone.

Monotone means that whenever a recursively-enumerable set L_1 has the property,

so does every superset.

Theorem (Rice '56):

Every non-monotone property of the recursively-enumerable sets is not semi-decidable.