UNDECIDABILITY OF THE POST'S CORRESPONDENCE PROBLEM

Aditya Gupta
Armaan Khetarpaul
Shankaradithyaa Venkateswaran
Suhas Vundavilli

CONTENTS

POST CORRESPONDENCE PROBLEM

- Formulated by Emil Post in 1946.
- Presented to you by Shankar in 2024.
- The need for this problem is due to its simplicity compared to the halting problem or the Entscheindungs problem.

STATEMENT

Given two lists M and N, of non-empty strings over an alphabet Σ which has at least 2 symbols:

$$M = (x_1, x_2, x_3, \dots, x_n) \text{ and } N = (y_1, y_2, y_3, \dots, y_n)$$

We need to pick a sequence of indices (repetition allowed) $(i_k)_{1 \le k \le K}$ with $K \geq 1$ and $1 \le i_k \le n \; \forall k$ such that:

$$x_{i_1} x_{i_2} \dots x_{i_k} = y_{i_1} y_{i_2} \dots y_{i_k}$$

EXAMPLE

a_1	a_2	a_3		b_1	b_2	b_3
0	01	110		100	00	11

Can we find a solution to the PCP here, in this case?

SOLUTION

a_1	a_2	a_3		b_1	b_2	b_3
0	01	110		100	00	11

- Label the indices $\{1, 2, 3\}$.
- From the table it is evident that we start with index 3.
- From this, what will be the next step?

SOLUTION

a_1	a_2	a_3		b_1	b_2	b_3
0	01	110		100	00	11

- One possible solution could be $(3, 2, 3, 1)$.
- This would be the following:

$$a_3a_2a_3a_1 = 110 * 01 * 110 * 0$$
$$b_3b_2b_3b_1 = 11 * 00 * 11 * 100$$
$$a_3a_2a_3a_1 = 110011100 = b_3b_2b_3b_1$$

- From this we can also say that all the repetitions of this indexing is also a solution such as $(3, 2, 3, 1, 3, 2, 3,1)$ and so on.

ANOTHER EXAMPLE

a_1	a_2	a_3		b_1	b_2	b_3
ab	bab	bbaaa		a	ba	bab

What would be a solution for this problem?

ANOTHER EXAMPLE

a_1	a_2	a_3		b_1	b_2	b_3
ab	bab	bbaaa		a	ba	bab

In fact, this example has no solution!

ANOTHER FORMULATION

- Now let us consider another instance of the Post Correspondence Problem in the domino form.
- We have a set of $\boldsymbol{N}$ dominos each with two strings on it, a numerator and denominator strings: $\left[\frac{a_1}{b_1}\right], \ldots, \left[\frac{a_n}{b_n}\right]$.
- The task is to arrange the dominos so that the strings of the numerators line up with the strings of the denominators.

A HARDER EXAMPLE

abab	aaabbb	aab	ba	ab	aa
ababaaa	bb	baab	baa	ba	a

Can we find a solution for the problem here, or is there no solution at all?

A HARDER EXAMPLE

- Yes! In fact, there are multiple solutions.
- A solution is (123).
- Another solution could be (456).
- And joining the solutions like (123456) is also valid.
- And of course, any repetition of the same is also a solution.

A HARDER EXAMPLE

abab	aaabbb	aabab	ba	aba	aa
ababaaa	bb	baaba	baa	bab	a

Does this problem have a solution?

PCP TO MPCP (REDUCTIONS)

We want to try and show that PCP is undecidable. But direct proof is way too hard. We proceed through the path of reductions.

RECOGNIZABILITY AND DECIDABILITY

- A language is said to be **DECIDABLE** if there exists a Turing Machine that accepts all strings in the language and rejects every other string.
- In simpler words, we require a Turing Machine that "halts" on every input.
- Similarly, a language is said to be **RECOGNIZABLE** if there exists a Turing Machine which accepts all strings in the language, and either rejects or does not halt on any other string.

REDUCTIONS

Let $L \subseteq A^*$ and $M \subseteq B^*$ be two languages. We say $\boldsymbol{L\ reduces\ to\ M}$ and write

$L \leq M \Leftrightarrow \exists$ a computable map $\sigma: A^* \to B^*$ such that

$$w \in L \ \Leftrightarrow \sigma(w) \in M$$

Also, if $L \leq M$ then:

- $M\ is\ recognizable \to L\ is\ recognizable.$
- $M\ is\ decidable\ \to L\ is\ decidable$

PCP IS RECOGNIZABLE

- It is quite simple to construct a Turing Machine which halts only on solutions to the PCP.
- First, we encode the strings in the tuple form, i.e. $((a_1, b_1), (a_2, b_2), \dots, (a_k, b_k))$.
- Next, we test systematically longer and longer sequences $i_1, i_2, \dots, i_n \in \{1,2,\dots,k\}$ whether they represent a match.
- We accept and terminate if we find a match, else we enter an infinite loop.

PCP IS UNDECIDABLE

- Proving the undecidability of PCP is in general quite tough, but by using the simple trick of reducing a known undecidable problem (in this case the Halting Problem) into PCP, our job would be done.
- However, in order to reduce it, we make use of an intermediate problem.

REDUCTION OF PCP

- We first define a new problem, namely the Modified Post Correspondence Problem (MPCP).
- Then, we reduce the MPCP into PCP.
- Finally, we reduce the Halting problem to MPCP, and we are done.

DEFINING MPCP

Given two lists M and N, of non-empty strings over an alphabet Σ which has at least two symbols:

$M = (x_1, x_2, \dots, x_n)$ and $N = (y_1, y_2, \dots, y_n)$

We need to pick a sequence of indices $\{i_k\}$ $(1 \leq k \leq K)$ with $K \geq 1$ and $1 \leq i_k \leq N$ $\forall k$ and $i_1 = 1$ such that:

$$x_{i_1} x_{i_2} \dots x_{i_K} = y_{i_1} y_{i_2} \dots y_{i_K}$$

REDUCING MPCP TO PCP

Choose any two characters not already in the alphabet Σ, say $\$$ and $\#$. Now define the following transformations of a word $w = a_1 a_2 \dots a_n \in \Sigma^+$:

$$\overline{w} = \# a_1 \# a_2 \# \dots \# a_n \#$$

$$\dot{w} = a_1 \# a_2 \# \dots \# a_n \#$$

$$\ddot{w} = \# a_1 \# a_2 \# \dots \# a_n$$

REDUCING MPCP TO PCP

- Let the tuple encoding of lists be given by $C = ((a_1, b_1), (a_2, b_2), \dots, (a_k, b_k))$.
- Define a function f on C as follows

$$f(C) = ((\overline{a_1}, \ddot{b}_1), (\dot{a}_1, \ddot{b}_1), (\dot{a}_2, \ddot{b}_2), \dots, (\dot{a}_k, \ddot{b}_k), (\$, \#\$))$$

- Clearly, the function f is computable.

REDUCING MPCP TO PCP

- Claim : C has a solution to MPCP $\Leftrightarrow f(C)$ has a solution to PCP.
- We shall prove this over the next few slides.

Idea: Given an instance of MPCP to f that has a solution, it returns an instance of PCP, the solution to which can be computed from the solution to MPCP by an algorithm.

FORWARD IMPLICATION

- Suppose $1, i_1, i_2, \dots i_n$ are indices in C corresponding to a solution to the MPCP. We claim that $1, i_1 + 1, i_2 + 1, \dots, i_n + 1, k + 2$ are the indices in $f(C)$ corresponding to a solution of the PCP.
- Suppose such a solution exists. Then, it must start with 1 and end with $k + 2$.
- Also, all middle elements must be the corresponding elements from C.

BACKWARD IMPLICATION

- Suppose $i_1, i_2, \dots, i_m$ correspond to the indices in $f(C)$ which is a solution to the PCP.
- Using a similar argument as previously, we can see that i_1 must be 1, i_m must be $k+2$.
- We now show that $1, i_2 - 1, \dots, i_{m-1} - 1$ are the indices in C corresponding to a solution of the PCP.

REDUCTION OF MPCP

- We have shown that MPCP can be reduced to PCP.
- This completes the first half of the proof.
- Now we show that the Halting Problem can be reduced to MPCP.

MPCP TO HALTING

Now that we have MPCP in our hands, we are ready to reduce it to MPCP and prove its undecidability.

DEFINING HALTING PROBLEM

- Assume we have a Turning Machine $\boldsymbol{M}$ and an input string $\boldsymbol{x}$.
- Will the Turing Machine “Halt” (accept) on this input?
- This problem is undecidable.

SNAPSHOT OF A TURING MACHINE

- Consider a Turing Machine $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ with input string $\boldsymbol{w}$ at state $\boldsymbol{q}$, with its pointer at the $\boldsymbol{i^{th}}$ position.
- The snapshot at this instance is $\boldsymbol{aqb}$, where $\boldsymbol{a} = \boldsymbol{w_{1\ldots i}}$ and $\boldsymbol{b} = \boldsymbol{w_{(i+1)\ldots}}$

HALTING OF TURING MACHINE

- A Turing Machine is said to "Halt" if it terminates in a finite number of steps.
- These steps can be represented as snapshots.
- For halting, the snapshots tuple is $(q_0w,\ a_1q_1b_1, \dots , a_kq_kb_k)$ where q_0 is the initial state and q_k is a final state.

REDUCING HALTING TO MPCP

- We map the transitions of the Turing Machine to instances of the MPCP set.
- Consider the Turing Machine M with input w. For every step in the Turing Machine, we construct a domino corresponding to the MPCP.

DOMINO CONSTRUCTION

- For every step in the Turing Machine, a domino is a tuple consisting of snapshots of its state before and after the step.
- By keeping these together, we get the full sequence of steps the Turing Machine went through.
- If the Turing Machine halted, this sequence would be finite.

PROCEDURE TO PUT DOMINOS

- The first domino we input is:

$$\left[\frac{\#}{\#qw\#}\right]$$

- The term at the bottom is the initial snapshot of the Turing Machine.

RIGHT TRANSITIONS

- For every $a, b \epsilon \Gamma$, $q, r \epsilon Q$, where $q \neq q_{reject}$, if there is a transition $\delta(q, a) = (r, b, R)$, we append the domino:

$$\left[\frac{qa}{br}\right]$$

LEFT TRANSITIONS

- For every $a, b, c \in \Gamma$, $q, r \in Q$, where $q \neq q_{reject}$, if there is a transition $\delta(q, a) = (r, b, L)$, we append the domino:

$$\left[\frac{cqa}{rcb}\right] \forall c \in \Gamma$$

ACCOUNTING FOR SEPARATORS

- For every $a \epsilon \Gamma$, add the domino:

$$\left[\frac{a}{a}\right]$$

- To separate each configuration, add the dominoes:

$$\left[\frac{\#}{\#}\right] \text{ and } \left[\frac{\#}{-\#}\right]$$

ACCEPTING STATE TRANSITIONS

- For every $a \in \Gamma$, we append the dominos:

$$\left[\frac{a q_{accept}}{q_{accept}}\right], \left[\frac{q_{accept} a}{q_{accept}}\right]$$

- These transitions balance the tape after termination.

FINAL DOMINO

- Finally, we input the domino:

$$\left[\frac{q_{accept}\#\#}{\#}\right]$$

- This helps account for the initial domino where the bottom part was longer.

HP IMPLIES MPCP

- If the Turing Machine halts, the domino representation can be used as a solution for MPCP.
- The length of the denominator is always larger than the numerator, unless we hit an accepting stage.
- Thus, the halting problem implies MPCP.

MPCP IMPLIES HP

- Suppose we have an MPCP problem with a finite solution. This writes out the finite steps that the Turing Machine took.
- If the MPCP problem has no solutions, then there is no finite way to make the Turing Machine stop.
- Thus, MPCP and HP are equivalent, i.e., undecidable.

SUMMARY

To summarise, we have constructed an instance of MPCP from our Turing Machine M on word w which will have a solution iff M accepts w.

QUICK EXAMPLE

Let $\Gamma = \{0, 1, 2, \_\}$, and $w = 0100$ be the starting string.

Corresponding to TM M let a transition be: $\delta(q_0, 0) = (q_7, 2, R)$.

So, we first place the domino $\left[\frac{\#}{\#q_0 0100\#}\right]$ and begin matching. Our aim is to simulate the snapshot sequence of 0100 on M.

Solution string: $numStr = \#$, $denomStr = \#q_0 0100\#$

QUICK EXAMPLE

Next, we add a domino $\left[\frac{q_0 0}{2 q_7}\right]$ corresponding to the given transition.

Then we add the dominoes $\left[\frac{0}{0}\right], \left[\frac{1}{1}\right], \left[\frac{2}{2}\right], \left[\frac{\_}{\_}\right], \left[\frac{\#}{\#}\right], \left[\frac{\#}{\_\#}\right]$

Solution string:

$$numStr = \# \cdot q_0 0 \cdot 1 \cdot 0 \cdot 0 \cdot \#,$$
$$denomStr = \# q_0 0100\# \cdot 2q_7 \cdot 1 \cdot 0 \cdot 0 \cdot \#$$

QUICK EXAMPLE

If $\delta(q_7, 1) = (q_5, 0, R)$ then we add domino $\left[\frac{q_7 1}{0 q_5}\right]$. Then we again try to match.

Solution string:

$$numStr = \cdots 2 \cdot q_7 1 \cdot 0 \cdot 0 \cdot \#$$
$$denomStr = \cdots (2q_7 \cdot 1 \cdot 0 \cdot 0 \cdot \#) \cdot 2 \cdot 0q_5 \cdot 0 \cdot 0 \cdot \#$$

QUICK EXAMPLE

We continue this way till we reach the accept state. Then we add $\left[\frac{aq_f}{q_f}\right], \left[\frac{q_f a}{q_f}\right] \forall a \in \Gamma$ to the dominoes (and to the solution string if required) and finally add $\left[\frac{q_f\#\#}{\#}\right]$ to complete the match. Thus, making a match can force the simulation of M to occur.

TRY EXAMPLE:

Simulate the process for the TM that accepts the universal binary language $(0+1)^*$. The TM is given by:

$$M = \left(\{t\}, \{0,1\}, \{0,1,\vdash,\_\}, t, \delta, b, NIL\right)$$

$$\delta(t,a) = (t,a,R)\forall a \in \{0,1,\vdash,\_\}$$

Simulate the word $\mathbf{01}$ on the TM.

TRY EXAMPLE:

I'll help you out by giving the dominoes:

$$\left[\frac{\#}{\#t01\#}\right]_1, \left[\frac{t\ \vdash}{\vdash\ t}\right]_2, \left[\frac{t0}{0t}\right]_3, \left[\frac{t1}{1t}\right]_4, \left[\frac{t\_}{\_t}\right]_5, \left[\frac{\vdash}{\vdash}\right]_6, \left[\frac{0}{0}\right]_7, \left[\frac{1}{1}\right]_8, \left[\frac{\_}{\_}\right]_9,$$

$$\left[\frac{\#}{\#}\right]_{10}, \left[\frac{\#}{\_\#}\right]_{11}, \left[\frac{\vdash t}{t}\right]_{12}, \left[\frac{0t}{t}\right]_{13}, \left[\frac{1t}{t}\right]_{14}, \left[\frac{\_t}{t}\right]_{15}, \left[\frac{t\#\#}{\#}\right]_{15}$$

TRY EXAMPLE:

I'll help you out by giving the dominoes:

$$\left[\frac{\#}{\#t01\#}\right]_1, \left[\frac{t0}{0t}\right]_2, \left[\frac{t1}{1t}\right]_3, \left[\frac{t\_}{\_t}\right]_4, \left[\frac{0}{0}\right]_5, \left[\frac{1}{1}\right]_6, \left[\frac{\_}{\_}\right]_7,$$

$$\left[\frac{\#}{\#}\right]_8, \left[\frac{\#}{\_\#}\right]_9, \left[\frac{0t}{t}\right]_{10}, \left[\frac{1t}{t}\right]_{11}, \left[\frac{\_t}{t}\right]_{12}, \left[\frac{t\#\#}{\#}\right]_{13}$$

TRY EXAMPLE:

The solution sequence is given by:

$$\left[\frac{\#}{\#t01\#}\right]_1 \left[\frac{t0}{0t}\right]_2 \left[\frac{1}{1}\right]_6 \left[\frac{\#}{\#}\right]_8 \left[\frac{0}{0}\right]_5 \left[\frac{t1}{1t}\right]_3 \left[\frac{\#}{\#}\right]_8 \left[\frac{0}{0}\right]_5 \left[\frac{1t}{t}\right]_{11} \left[\frac{\#}{\#}\right]_8 \left[\frac{0t}{t}\right]_{10} \left[\frac{\#}{\#}\right]_8 \left[\frac{t\#\#}{\#}\right]_{13}$$

Where the string is:

$$\#t01\#0t1\#01t\#0t\#t\#\#$$

And the corresponding sequence is: 1 2 6 8 5 3 8 5 11 8 10 8 13

TRY EXAMPLE:

$$\#t01\#0t1\#01t\#0t\#t\#\#\#$$

If you look closely, the strings separated by # are the Computation History/ Snapshot history of the machine:

$$t01 \rightarrow 0t1 \rightarrow 01t \rightarrow 0t \rightarrow t$$

Which is a good sign that it works!

VARIANTS OF PCP

Let’s discuss the variants of PCP.

MONOID MORPHISMS

- $\Sigma \rightarrow$ Alphabet, then $\Sigma^* \rightarrow$ Free Monoid
- We represent PCP using monoid morphisms.
- Let A, B be finite alphabet
- An instance of PCP is a pair of monoid morphism (g, h), $g, h: A^* \rightarrow B^*$. (with $|A| = n$)

MONOID MORPHISMS

- Aim is to determine:

$$Does\ \exists w \in A^{+}\ such\ that\ g(w) = h(w)\ ?$$

- $A \rightarrow Indexing\ List$
- $B \rightarrow Word\ List$

MONOID MORPHISMS

Example:- Let $A = \{1,2,3\}, B = \{0,1\}$ where

a	$g(a)$
1	100
2	0
3	1

a	$h(a)$
1	1
2	100
3	00

$$Consider\ w = 1311322$$
$$then\ g(w) = 10011100100100 = h(w)$$

$PCP(n)$

- $PCP(n)$ is PCP with $|A| = n$. That is, the size of input lists is n.
- Arguably one of the best boundaries between the undecidable and decidable regimes.
- The boundary is given by the problem:

$$\textit{Find smallest } u \textit{ and largest } l \textit{ such that } PCP(u) \textit{ is undecidable}, PCP(l) \textit{ is decidable}.$$

$PCP(n)$

Through extensive research (which can't be condensed in an hour) the following results have been established for $PCP(n)$:

- Decidable if $n \leq 2$ (*Ehrenfeucht, A.; Karhumäki, J.; Rozenberg, G. (November 1982)*)
- Undecidable if $n \geq 5$ (*T. Neary (2015)*)
- Unknown behaviour for $n = 3\ or\ 4$ (Open Problems)

BOUNDED PCP

- Total number of tiles that can be used (including repetitions) is k.
- NP-Complete.
- RNP (Random NP), i.e., Hard even in the average case.

INFINITE PCP

- Search for an infinite sequence of indices.
- Undecidable in the general case.
- For input lists of size $\boldsymbol{n}$, it is proven to be undecidable for $\boldsymbol{n} \geq \mathbf{9}$ and decidable for $\boldsymbol{n} \leq \mathbf{2}$.

MARKED PCP

- $\boldsymbol{Pref_k(z)}$ = Prefix of length $\boldsymbol{k}$ of $\boldsymbol{z}$, $\boldsymbol{(Pref_k(z) = z \; if \; |z| \leq k)}$
- A morphism $\boldsymbol{g: A^+ \rightarrow B^+}$ is $\boldsymbol{k}$-marked if $\boldsymbol{Pref_k\big(g(a)\big) \neq Pref_k\big(g(b)\big) \; \forall a \neq b \in \Sigma}$ (different indices map to words with different prefix)
- An instance of PCP is $\boldsymbol{k}$-marked if both $\boldsymbol{g, h}$ are $\boldsymbol{k}$-marked.

MARKED PCP

- Marked PCPs are $\boldsymbol{EXPTIME}$.
- $\boldsymbol{n}$-marked PCP is decidable $\forall \boldsymbol{n} \in \mathbb{N}$
- Loose $\mathbf{2}$-marked PCP is undecidable.
- $\boldsymbol{Infinite}$-marked PCP is decidable.

VARIANTS OF PCP

Many variants of PCP have been considered because, when one tries to prove undecidability of some new problem by reducing from PCP, it often happens that the first reduction one finds is not from PCP itself but from an apparently weaker version.

THANK YOU

"If a machine is expected to be infallible, it cannot also be intelligent" – Alan Turing