Abrahim Ladha

Lecture 15 Post's Correspondence Problem

Last time we proved A_{TM}, E_{TM}, EQ_{TM}, are undecidable. The keen eyed among you may notice these are all problems which have to do with language problems relating to Turing machines. It ends up being true by Rice's Theorem: Most nontrivial semantic properties are undecidable. A syntatic property is about the encoding of the machine. For example "M has 17 states". Easily decidable, count the states. A semantic property might be "M recognizes a language which has some (maybe different) TM to recognize the same language with 17 states". Syntatic properties are about the machines, Semantic properties are about their execution, or the language they decide.

Your second thought should be "The only undecidable problems we have seen are again, language acceptance problems." It feels conditional on the Church-Turing Thesis. There do exist unsolvable problems with nothing to do with language theory. Here, we give a puzzle with no algorithmic solution. It is provably unsolvable. P.C.P.

Let a "domino" or "tile" be a pair of strings, consisting of an upper and lower portion. For example, a set of dominoes could be

$$\left\{ \left[\frac{b}{ca}\right], \left[\frac{a}{ab}\right], \left[\frac{ca}{a}\right], \left[\frac{abc}{c}\right] \right\}$$

we say a set has a match if there exists a repeated sequence where the top = bottom

$$\left[\frac{a}{ab}\right]\left[\frac{b}{ca}\right]\left[\frac{ca}{a}\right]\left[\frac{a}{ab}\right]\left[\frac{abc}{c}\right] \qquad \begin{aligned} a \cdot b \cdot ca \cdot a \cdot abc &= abcaaabc \\ ab \cdot ca \cdot a \cdot ab \cdot c &= abcaaabc \end{aligned}$$

so this set has a match.

We prove that PCP is algorithmically unsolvable. That is

$$PCP = \{\langle P \rangle \mid P \text{ a set of tiles with a match}\}$$

is undecidable. The proof idea is simple but has a lot of small details. First lets explore its universality in some way. First note we can set up a set of tiles such that we can force any decision making procedure to temper its behavior a certain way. ~~Force~~ for example, the following set of tiles, the first (and last) choices are fixed. Any procedure is tempered into picking the first tile first.

$$\left\{\begin{bmatrix}\#b\\\#\end{bmatrix}\begin{bmatrix}a\\b\end{bmatrix}\begin{bmatrix}\$\\a\$\end{bmatrix}\right\}$$ for no other reason than simply by the fact that there

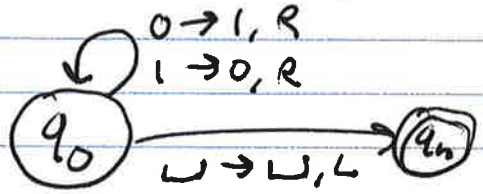
exists only one tile with the same starting letter. Similarly for the ending tile. We can also force a ~~~~ procedure to try and find matches, infinitely without halting.

$$\left\{\begin{bmatrix}\#a\\\#\end{bmatrix}\begin{bmatrix}a\\a\end{bmatrix}\right\} \text{ or } \left\{\begin{bmatrix}\#a\\\#\end{bmatrix}\begin{bmatrix}b\\a\end{bmatrix}\begin{bmatrix}a\\b\end{bmatrix}\right\}$$

The first one should select the first tile, and then infinitely add tiles to chase $\begin{matrix}\#a^{k+1}\\\#a^{k}\end{matrix}$. It does provably loop however, using these ideas, we can encode the transition function of a Turing machine into a set of tiles. With the right setup, we can ensure that the tile instance only has a match if M accepts w.

Notice that beggining with a tile that initiates a deficiency, we can force the next tile to be like $\begin{bmatrix}?\\a\ldots\end{bmatrix}$

A computation history is a sequence of configurations. In some string encoding made useful. Here, we will construct a set of tiles such that its only string match is this computation history. For example, the following is a computation history for the following machine.

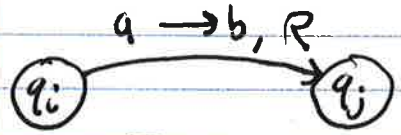


$\#q_0 10 \# 0 q_0 0 \# 01 q_0 \sqcup \# 0 q_h 1 \sqcup \#$

We begin our tile with this starting one. Notice that the next tile is forced to begin with q_0 at the top. We will only add two such tiles, one with $q_0 a$ and the other with $q_0 b$, so only one gets picked to match to $q_0 w_1$

1 $\left[\begin{matrix}\# \\ \# q_0 w_1 w_2 \ldots w_n \#\end{matrix}\right]$

given such a transition in our machine $\delta(q_i, a) = (q_j, b, R)$ our configurations would change looking like $q_i a \to b q_j$ so we emulate this in our tiles. Of course, we must also simulate left moves, so if $\delta(q_i, a) = (q_j, b, L)$, our configurations would change like $c q_i a \to q_j c b$.



2 $\left[\begin{matrix}q_i a \\ b q_j\end{matrix}\right]$

3 $\left[\begin{matrix}a\, q_i a \\ q_j\, a\, b\end{matrix}\right], \left[\begin{matrix}b\, q_i a \\ q_j\, b\, b\end{matrix}\right], \left[\begin{matrix}\sqcup\, q_i a \\ q_j\, \sqcup\, a\end{matrix}\right]$

We add one domino per selection of c, here suppose Γ = just $\{a, b, \sqcup\}$. We need one for each possible left move of our machine. I hope you see the pattern here. We have created a set of tiles such that the decisions made to create a match are forced to simulate the starting machine according to its transition function.

4 $\begin{bmatrix} a \\ a \end{bmatrix}\begin{bmatrix} b \\ b \end{bmatrix}\begin{bmatrix} \sqcup \\ \sqcup \end{bmatrix}$ we need some more tiles make sure everything is set up. We add one singleton tile like so $\forall a \in \Gamma$ to make copies of the rest of the tape for us. Recall in a sequence of configurations, only a local part of each changes. Most of the tape remains unchanged. These tiles do that copying for us. We also need a cap between configurations and a way to use more space. Recall a configuration can have more $\sqcup$ like leading zeroes because the tape is infinite. We only choose to write as many as necessary, one. If we want more, it will have to be done for the next configuration. The

5 $\begin{bmatrix} \# \\ \# \end{bmatrix}\begin{bmatrix} \# \\ \sqcup\ \# \end{bmatrix}$

accept state being q_a, we add the following tiles. This basically has the aq_a "eat" the rest of the tape. Once you reach

6 $\begin{bmatrix} a\,q_a \\ q_a \end{bmatrix}\begin{bmatrix} b\,q_a \\ q_a \end{bmatrix}\begin{bmatrix} \sqcup\,q_a \\ q_a \end{bmatrix}$.

the accept state in a Turing machine, you halt. However, our match will keep going so we have the following nice end cap.

7 $\begin{bmatrix} q_a\ \#\ \# \\ \qquad\ \# \end{bmatrix}$

This completes the match. If our last real configuration was like the left, we cap it like the right

$$\begin{bmatrix} \# & & \\ & & \\ & & \\ \# & a & a\ q_a\ a\ a\ \# \end{bmatrix} \qquad \begin{bmatrix} \# & a & q_a \\ \vdots & & \\ \# & q_a & \end{bmatrix}\begin{bmatrix} \# \\ \\ \# \end{bmatrix}\begin{bmatrix} q_a\ \#\ \# \\ \\ \qquad\ \# \end{bmatrix}$$

You may have noticed we added tiles to not enforce the rule of a single start, like $\left[\frac{a}{a}\right]$ or $\left[\frac{\#}{\#}\right]$ given a set of dominoes we modify them in the the following way for $u = u_1 \dots u_n$ let

$$\bullet u = \bullet u_1 \bullet u_2 \bullet \cdots \bullet u_n$$
$$u\bullet = u_1 \bullet u_2 \bullet \cdots \bullet u_n \bullet$$
$$\bullet u \bullet = \bullet u_1 \bullet u_2 \bullet \cdots \bullet u_n \bullet$$

let $\left[\frac{t_s}{b_s}\right]$ be the start tile, $\left[\frac{t_e}{b_e}\right]$ be the end tile.

given our set of tiles

$$\left\{\left[\frac{t_s}{b_s}\right], \left[\frac{t_1}{b_1}\right] \cdots \left[\frac{t_k}{b_k}\right] \left[\frac{t_e}{b_e}\right]\right\}$$

we modify them like

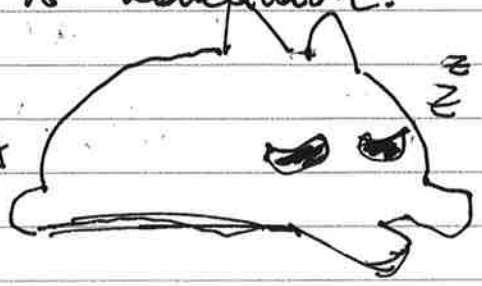
$$\left[\frac{\bullet t_s}{\bullet b_s \bullet}\right], \left[\frac{\bullet t_1}{b_1 \bullet}\right] \cdots \left[\frac{\bullet t_k}{b_k \bullet}\right] \left[\frac{\bullet t_e \bullet}{b_e \bullet}\right]$$

This can be generalized to make our reduction more like

$$A_{TM} \leq_m MPCP \leq_m PCP.$$

Now we show Baba is you is undecidable.

$$A_{TM} \leq_m MPCP \leq_m PCP \leq_m BABA$$



next. lets stress why the computation is correct we begin like.

#	then we are forced to add tiles
# C_0	in which the tops match C_0. By
. . . .	doing so, we have closed the bottoms
. . . .	to compute and define C_1
. . . . .	
# C_0 #	now, we must repeat matching
# C_0 # C_1	C_1 to force us to compute C_2
# C_0 # C_1 #	and so on.
# # C_1 # C_2	

Our only match exists if and only if there is an accepting computation of M on w. We had no reject end tile. If the machine loops, the computation history would be infinite and so there would be no match. We see that our construction

$$f(\langle M, w \rangle) = \langle P \rangle \quad \text{is correct.}$$

$$\langle M, w \rangle \in A_{TM} \iff \langle P \rangle \in PCP$$

So we conclude PCP is undecidable.