Lecture 21

Instructor: Karteek Sreenivasaiah

1st March 2020

Recap

In the last few lectures, we have covered the following topics:

- Definition of Turing Machines, and several toy examples
- The definition of a decidable language (a.k.a recursive)
- The definition of a Turing-recognizable language (a.k.a recursively-enumerable)
- Examples of decidable languages including: A_{DFA}, E_{DFA}, EQ_{DFA}, A_{CFG}
- Undecidability

Recap - Undecidability

We showed the following languages to be undecidable:

$$
\begin{aligned}
A_{\mathsf{TM}} &= \{\langle M, w\rangle \mid M \text{ is a TM that accepts } w\} \\
\mathsf{HALT}_{\mathsf{TM}} &= \{\langle M, w\rangle \mid M \text{ is a TM that halts on input } w\} \\
\mathsf{E}_{\mathsf{TM}} &= \{\langle M\rangle \mid M \text{ is a TM with } L(M) = \emptyset\} \\
\mathsf{REG}_{\mathsf{TM}} &= \{\langle M\rangle \mid M \text{ is a TM and } L(M) \text{ is regular}\} \\
\mathsf{EQ}_{\mathsf{TM}} &= \{\langle M, N\rangle \mid M \text{ and } N \text{ are TMs and } L(M) = L(N)\} \\
\mathsf{ALL}_{\mathsf{CFG}} &= \{\langle G\rangle \mid G \text{ is a CFG and } L(G) = \Sigma^*\}
\end{aligned}
$$

- We used an explicit diagonalization argument to show that A_{TM} is undecidable.
- For the other languages, we showed a *reduction* from another undecidable language.

We look at the proof for $\mathsf{REG}_{\mathsf{TM}}$ to serve as recap.

Recap - REG_{TM} is undecidable

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

Theorem

REG_{TM} is undecidable.

Proof:

- Suppose REG_{TM} is decidable. Let R decide it.
- We know that A_{TM} is undecidable.
- We show that R can be used to construct a decider N for A_{TM}.

Recap - REG_{TM} is undecidable

We construct a decider TM N for A_{TM} as follows:

- Input: $\langle M, w \rangle$.
- Construct TM H during runtime as follows:
  - Input: x
  - If $x = 0^n 1^n$ for some $n > 0$, then accept.
  - Else, run M on w. Accept x only if M accepts w.
- Run R on H. If R accepts, then accept. Else Reject.

Note that the TM H constructed during runtime has the following property:

$$L(H) = \begin{cases} \{0^n 1^n \mid \ n > 0\} \text{ if } M \text{ does not accept } w \\ \{0,1\}^* \text{ if } M \text{ accepts } w \end{cases}$$

So $L(H)$ is regular if and only if M accepts w. Hence the machine N decides A_{TM}.

Post Correspondence Problem

Define a *domino* to be a tuple of two strings (s, t) written as: $\left[\frac{s}{t}\right]$

Consider a set of dominos such as:

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

The *Post Correspondence Problem* (PCP) asks if the set of dominos given can be used to form a list so that the top string equals the bottom string.

Post Correspondence Problem

Define a *domino* to be a tuple of two strings (s, t) written as: $\left[\frac{s}{t}\right]$
Consider a set of dominos such as:

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

The *Post Correspondence Problem* (PCP) asks if the set of dominos given can be used to form a list so that the top string equals the bottom string.
For the above set of dominos, here is a list that works:

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

The string at the top, and bottom are both “abcaaabc”. Such a list is called a *match*.
Note how a domino was used two times to create the match.

Post Correspondence Problem

Formally, an instance of PCP is a set P of dominos:

$$P = \left\{ \left[\frac{t_1}{b_1} \right], \cdots, \left[\frac{t_k}{b_k} \right] \right\}$$

The t_i are strings in the **t**op and b_i are the strings in the **b**ottom.
A *match* is a sequence $i_1, \ldots, i_\ell$ such that $t_{i_1} t_{i_2} \cdots t_{i_\ell} = b_{i_1} b_{i_2} \cdots b_{i_\ell}$
Note: A domino can be used any number of times in a match!

PCP

$\text{PCP} = \{\langle P \rangle \mid P \text{ is an instance of PCP with a match}\}$

Post Correspondence Problem

Theorem

PCP is undecidable

Proof:
The proof is by reduction from A_{TM}.
i.e., We have to construct a machine N such that:

- Input is $\langle M, w \rangle$.
- Using M and w, we generate (at runtime) a set P of dominos.
- The set P has a match if and only if M accepts w.

Post Correspondence Problem

Theorem

PCP is undecidable

Proof:
The proof is by reduction from A_{TM}.
i.e., We have to construct a machine N such that:

- Input is $\langle M, w \rangle$.
- Using M and w, we generate (at runtime) a set P of dominos.
- The set P has a match if and only if M accepts w.

However, the usual "template" that we used for most of the reductions so far does not work.
The proof is very creative. The rest of the lecture is dedicated to this proof.

Post Correspondence Problem

Recall that an *accepting computation history* of a TM M that accepts w is a sequence of configurations $C_1, C_2, \ldots, C_\ell$ such that:

- C_1 is the starting configuration of M on w.
- C_i follows C_{i-1}.
- C_ℓ is an accept configuration.

The idea to take input M, w and construct dominos such that the only match that can exist is an accepting computation history of M on w.

i.e., The string formed on the top and bottom of the match will be the accepting computation history of M on w. There will be no match if M does not accept w.

Post Correspondence Problem

We first consider a slightly modified version of PCP:

MPCP

$$\text{MPCP} = \left\{ \langle P \rangle \mid \begin{array}{l} P \text{ is an instance of PCP with a match} \\ \text{that starts with the first domino} \end{array} \right\}$$

We will show that MPCP is undecidable by reduction from A_{TM}:

- Construct N that takes M, w as input.
- Programme N to generate a set of dominos that have a match if and only if M accepts w.

Post Correspondence Problem

At a very abstract level, we would like to create dominos like this:

$$\left[\frac{}{C_1}\right], \left[\frac{C_1}{C_2}\right], \left[\frac{C_2}{C_3}\right]$$

where the blue domino is the first one.

Notice how the match of C_1 of the first domino will place C_2 at the bottom of the second domino. And to match the string C_2, we will be forced to place C_3 at the bottom of the next domino, and so on.

This is the general idea. But we cannot create these dominos explicitly without running the machine ourselves! If we run the machine to create the dominos, then our procedure might never terminate!

Post Correspondence Problem

The First Domino:

In any accepting history of M on w, the start configuration is $q_0 w_1 w_2 \cdots w_\ell$.

So we create the first domino as: $\left[\frac{\#}{\#q_0 w_1 w_2 \cdots w_\ell \#}\right]$

where $\#$ is a delimiter of sorts.

- Notice how the top symbol "#" matches the first symbol in the bottom.
- However, after the first symbol, the bottom string has to be matched using other dominos that we will create.

Post Correspondence Problem

Continuing the run:

- The first domino begins simulating M with the start config at the bottom.
- We would like to create dominos that can continue this run.

Suppose $\delta(q_0, w_1) = (q_5, w_1', R)$, we would like to create a domino like this:

$$\left[\frac{\#q_0 w_1 w_2 \cdots w_\ell \#}{\#w_1' q_5 w_2 \cdots w_\ell \#}\right]$$

so that the top string matches the bottom string of the first domino and the bottom string is next configuration in a run of M on w.

Post Correspondence Problem

Continuing the run:

- The first domino begins simulating M with the start config at the bottom.
- We would like to create dominos that can continue this run.

Suppose $\delta(q_0, w_1) = (q_5, w_1', R)$, we would like to create a domino like this:

$$\left[\frac{\#q_0 w_1 w_2 \cdots w_\ell \#}{\#w_1' q_5 w_2 \cdots w_\ell \#}\right]$$

so that the top string matches the bottom string of the first domino and the bottom string is next configuration in a run of M on w.

But we cannot really create such dominos because it would require us to run the machine ourselves. If M does not halt, our procedure would also create dominos forever!

Post Correspondence Problem

Continuing the run:

To execute the idea above, we will create dominos using the transition table δ of M.

For all $a, b, c \in \Gamma$ and all $p, q \in Q$ such that $q \neq q_{\text{reject}}$,

$$\text{If } \delta(p, a) = (q, b, R), \text{ then create } \left[\frac{qa}{bq}\right]$$

$$\text{If } \delta(p, a) = (q, b, L), \text{ then create } \left[\frac{cqa}{qcb}\right]$$

The above kinds of dominos are the only ones to contain state symbols.

To match parts of the configuration that are not near the read/write head, we create the following "filler" dominos:

For all $a \in \Gamma$, create domino $\left[\frac{a}{a}\right]$.

Post Correspondence Problem

Example: Let's see how the dominos so far look like with an example.
Let $\Gamma = \{0, 1, 2, \sqcup\}$, and let $\delta(q_0, 0) = (q_7, 2, R)$.
Suppose $w = 0100$.
The dominos our procedure creates are:

$$P = \left\{ \left[\frac{\#}{\#q_0 0100\#}\right], \left[\frac{q_0 0}{2q_7}\right], \left[\frac{0}{0}\right], \left[\frac{1}{1}\right], \left[\frac{2}{2}\right], \left[\frac{\sqcup}{\sqcup}\right] \right\}$$

Let's try and arrange them starting with the first domino.

Post Correspondence Problem

Since we are working with MPCP, we have to start with the first domino:

$$\left[\frac{\#}{\#q_0 0100\#}\right]$$

To match the bottom string, we are forced to choose the second domino in P:

$$\left[\frac{\#}{\#q_0 0100\#}\right]\left[\frac{q_0 0}{2q_7}\right]$$

The rest of the symbols are matched by the filler dominos:

$$\left[\frac{\#}{\#q_0 0100\#}\right]\left[\frac{q_0 0}{2q_7}\right]\left[\frac{1}{1}\right]\left[\frac{0}{0}\right]\left[\frac{0}{0}\right]$$

The top string is $\#q_0 0100$ and the bottom string is $\#q_0 0100\#2q_7 100$. Notice how the the starting configuration has been matched. However, this leaves the next configuration to be matched in the future.

Post Correspondence Problem

Continuing the run: We need the following dominos to match the delimiter # and handle blank symbols ⊔:

$$\left[\frac{\#}{\#}\right], \left[\frac{\#}{\sqcup\#}\right]$$

We need the second domino of this kind because sometimes the configuration might look like $a\,b\,b\,a\,c\,q_3\sqcup$.

Post Correspondence Problem

Completing the run: Finally, we need a way to handle the case when the machine halts and accepts.
We create these dominos for all $a \in \Gamma$:

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

The idea here is to *absorb* all neighboring symbols once we reach q_{accept}.
To allow for a complete match, we create the domino:

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

Post Correspondence Problem

Example: Let $\Gamma = \{0, 1, 2, \sqcup\}$, $\delta(q_0, 0) = (q_7, 2, R)$ and $\delta(q_7, 1) = (q_{\text{accept}}, 0, R)$. Let $w = 0100$. The dominos we create:

$$\left\{ \left[\frac{\#}{\#q_0 0100\#}\right], \left[\frac{q_0 0}{2q_7}\right], \left[\frac{q_7 1}{0q_{\text{accept}}}\right], \left[\frac{0}{0}\right], \left[\frac{1}{1}\right], \left[\frac{2}{2}\right], \left[\frac{\sqcup}{\sqcup}\right], \left[\frac{\#}{\#}\right], \left[\frac{\#}{\sqcup\#}\right] \right\}$$

We are not writing the q_{accept} dominos here to save space.

Post Correspondence Problem

Example: Let $\Gamma = \{0, 1, 2, \sqcup\}$, $\delta(q_0, 0) = (q_7, 2, R)$ and $\delta(q_7, 1) = (q_{\text{accept}}, 0, R)$. Let $w = 0100$. The dominos we create:

$$\left\{ \left[\frac{\#}{\#q_0 0100\#}\right], \left[\frac{q_0 0}{2 q_7}\right], \left[\frac{q_7 1}{0 q_{\text{accept}}}\right], \left[\frac{0}{0}\right], \left[\frac{1}{1}\right], \left[\frac{2}{2}\right], \left[\frac{\sqcup}{\sqcup}\right], \left[\frac{\#}{\#}\right], \left[\frac{\#}{\sqcup\#}\right] \right\}$$

We are not writing the q_{accept} dominos here to save space.
Just like before, we start with the first domino:

$$\left[\frac{\#}{\#q_0 0100\#}\right]$$

To match the bottom string, we are forced to choose the second domino in P:

$$\left[\frac{\#}{\#q_0 0100\#}\right]\left[\frac{q_0 0}{2 q_7}\right]$$

The rest of the symbols are matched by the filler dominos:

$$\left[\frac{\#}{\#q_0 0100\#}\right]\left[\frac{q_0 0}{2 q_7}\right]\left[\frac{1}{1}\right]\left[\frac{0}{0}\right]\left[\frac{0}{0}\right]\left[\frac{\#}{\#}\right]$$

Post Correspondence Problem

We proceed to match the unmatched part of the bottom string:

$$\left[\frac{\#}{\#q_0 0100\#}\right]\left[\frac{q_0 0}{2q_7}\right]\left[\frac{1}{1}\right]\left[\frac{0}{0}\right]\left[\frac{0}{0}\right]\left[\frac{\#}{\#}\right]\left[\frac{2}{2}\right]\left[\frac{q_7 1}{0q_{\text{accept}}}\right]\left[\frac{0}{0}\right]\left[\frac{0}{0}\right]\left[\frac{\#}{\#}\right]$$

Before proceeding, note the partial match so far:

$$\left[\frac{\#}{\#q_0 0100\#}\right]\left[\frac{q_0 0}{2q_7}\right]\left[\frac{1}{1}\right]\left[\frac{0}{0}\right]\left[\frac{0}{0}\right]\left[\frac{\#}{\#}\right]\left[\frac{2}{2}\right]\left[\frac{q_7 1}{0q_{\text{accept}}}\right]\left[\frac{0}{0}\right]\left[\frac{0}{0}\right]\left[\frac{\#}{\#}\right]$$

The colors blue and magenta show the match so far.
The unmatched string now is $20q_{\text{accept}}00\#$

Post Correspondence Problem

Now we use the dominos with q_{accept} to complete the match:

$$\cdots \left[\frac{2}{2}\right] \left[\frac{q_7 1}{0 q_{\text{accept}}}\right] \left[\frac{0}{0}\right] \left[\frac{0}{0}\right] \left[\frac{\#}{\#}\right] \left[\frac{2}{2}\right] \left[\frac{0 q_{\text{accept}}}{q_{\text{accept}}}\right] \left[\frac{0}{0}\right] \left[\frac{0}{0}\right] \left[\frac{\#}{\#}\right]$$

Note how the 0 got absorbed. We proceed this way:

$$\cdots \left[\frac{2}{2}\right] \left[\frac{0 q_{\text{accept}}}{q_{\text{accept}}}\right] \left[\frac{0}{0}\right] \left[\frac{0}{0}\right] \left[\frac{\#}{\#}\right] \left[\frac{2}{2}\right] \left[\frac{q_{\text{accept}} 0}{q_{\text{accept}}}\right] \left[\frac{0}{0}\right] \left[\frac{\#}{\#}\right]$$

Followed by:

$$\cdots \left[\frac{2}{2}\right] \left[\frac{q_{\text{accept}} 0}{q_{\text{accept}}}\right] \left[\frac{0}{0}\right] \left[\frac{\#}{\#}\right] \left[\frac{2}{2}\right] \left[\frac{q_{\text{accept}} 0}{q_{\text{accept}}}\right] \left[\frac{\#}{\#}\right]$$

Post Correspondence Problem

And:

$$\ldots \left[\frac{2}{2}\right] \left[\frac{q_{\text{accept}}0}{q_{\text{accept}}}\right] \left[\frac{\#}{\#}\right] \left[\frac{2q_{\text{accept}}}{q_{\text{accept}}}\right] \left[\frac{\#}{\#}\right]$$

Finally:

$$\ldots \left[\frac{2q_{\text{accept}}}{q_{\text{accept}}}\right] \left[\frac{\#}{\#}\right] \left[\frac{q_{\text{accept}}\#\#}{\#}\right]$$

The entire match is long:

$$\left[\frac{\#}{\#q_0 0100\#}\right] \left[\frac{q_0 0}{2q_7}\right] \left[\frac{1}{1}\right] \left[\frac{0}{0}\right] \left[\frac{0}{0}\right] \left[\frac{\#}{\#}\right] \left[\frac{2}{2}\right] \left[\frac{q_7 1}{0q_{\text{accept}}}\right] \left[\frac{0}{0}\right] \left[\frac{0}{0}\right] \left[\frac{\#}{\#}\right]$$

$$\left[\frac{2}{2}\right] \left[\frac{0q_{\text{accept}}}{q_{\text{accept}}}\right] \left[\frac{0}{0}\right] \left[\frac{0}{0}\right] \left[\frac{\#}{\#}\right] \left[\frac{2}{2}\right] \left[\frac{q_{\text{accept}}0}{q_{\text{accept}}}\right] \left[\frac{0}{0}\right] \left[\frac{\#}{\#}\right]$$

$$\left[\frac{2}{2}\right] \left[\frac{q_{\text{accept}}0}{q_{\text{accept}}}\right] \left[\frac{\#}{\#}\right] \left[\frac{2q_{\text{accept}}}{q_{\text{accept}}}\right] \left[\frac{\#}{\#}\right] \left[\frac{q_{\text{accept}}\#\#}{\#}\right]$$

We do not formally prove the correctness.
To show that PCP is undecidable:

- We can reduce MPCP to PCP using a simple trick.
- You can either figure out by yourself how to remove the restriction of using the first domino, or see the textbook for the trick.

Thank you!