In this class we give another example of an undecidable problem: the Post Correspondence problem (PCP). This problem looks very different than the halting problem.

Given two lists of strings $\alpha_1, \ldots, \alpha_N$ and $\beta_1, \ldots, \beta_N$, check if there exist a sequence of indices $i_1, \ldots, i_k$ such that

$$\alpha_{i_1} \cdots \alpha_{i_k} = \beta_{i_1} \cdots \beta_{i_k}.$$ 

We will use a domino of the form $\begin{bmatrix} \alpha_i \\ \beta_i \end{bmatrix}$ denote the words $\alpha_i$ and $\beta_i$. Now our problem becomes a domino arrangement problem, where the goal is to see if there is a possible arrangement of the dominoes to form matching sequences at the top and bottom. Note that a domino is allowed to be used multiple times.

We will use a slightly modified version of PCP (MPCP), where we are restricted to $i_1 = 1$. In other words, we add an extra condition that the first domino has to be used at the beginning. This can be shown to be equivalent to PCP (we will skip this detail).

## 1 Reducions

Given PCP, we can do several reductions to show even more problems are undecidable.

Recall from last lecture that a problem $A$ reduces to another problem $B$ if we can solve $A$ by solving another instance of $B$. Note that this means that if there is an algorithm for solving $B$, there would be an algorithm for solving problem $A$.

By reducing from the undecidability of $A_{TM}$, we can show that the following problems are undecidable:

1. Checking if a Turing machine $M$ halts on input $w$.
2. Checking if a Turing machine $M$ accepts input $w$.
3. Checking if the language accepted by a Turing machine $M$ is regular.
4. Checking if $M$ accepts $w$ whenever $M$ accepts its reversal $w^R$.

## 2 PCP is undecidable

To show that the Post correspondence problem is undecidable, we will reduce the Membership problem to it. The membership problem is defined as follows: Given the description of a Turing machine $M$ and a string $x$, check if $M$ accepts $x$. 

The idea for this is similar to the proof of the Cook-Levin theorem from last lecture. Given an arbitrary instance of the membership problem \(< M, x >\), we want to construct two lists of strings such that \( M \) accepts \( x \) if and only if there exists a sequence of indices producing matching strings.

Given \( M \) and \( w \), we add dominoes to our set to “simulate” the running of \( M \) on \( w \). We use 7 rules to add dominoes:

1. The starting domino: 
   \[
   \begin{bmatrix}
   \# \\
   \#q_0w_0 \cdots w_n\#
   \end{bmatrix}
   \]

2. The head-moves-right domino. For every transition of the form \( \delta(q, a) = (q', b, R) \) add the domino: 
   \[
   \begin{bmatrix}
   qa \\
   bq'
   \end{bmatrix}
   \]

3. The head-moves-left domino. For every transition of the form \( \delta(q, a) = (q', b, L) \) add the domino: 
   \[
   \begin{bmatrix}
   cqa \\
   q'cb
   \end{bmatrix}
   \]

4. The untouched domino. For every tape alphabet \( a \), add the domino: 
   \[
   \begin{bmatrix}
   a \\
   a
   \end{bmatrix}
   \]

5. The divider and extender dominos: 
   \[
   \begin{bmatrix}
   \# \\
   \#
   \end{bmatrix}
   \quad \text{and} \quad 
   \begin{bmatrix}
   \# \\
   \#
   \end{bmatrix}
   \]

6. The symbol eating dominos: 
   \[
   \begin{bmatrix}
   a_{\text{accept}} \\
   q_{\text{accept}}
   \end{bmatrix}
   \quad \text{and} \quad 
   \begin{bmatrix}
   q_{\text{accept}}a \\
   q_{\text{accept}}
   \end{bmatrix}
   \]

7. The ending domino: 
   \[
   \begin{bmatrix}
   q_{\text{accept}}\#\# \\
   \#
   \end{bmatrix}
   \]

At a high level, let us see why if \( M \) accepts \( w \), we will definitely have a sequence of dominoes which form a match. Let \( C_0, C_1, \ldots C_{\text{accept}} \) be the sequence of configurations of \( M \) (ignoring blank spaces). Start from rule 1, we have the partial domino match: 
   \[
   \begin{bmatrix}
   \# \\
   \#C_0
   \end{bmatrix}
   \]. Using rules 2-5, we can then get to \[
   \begin{bmatrix}
   \#C_0\# \\
   \#C_0#C_1
   \end{bmatrix}\]. Repeating these, we can get 
   \[
   \begin{bmatrix}
   \#C_0#C_1# \cdots \# \\
   \#C_0#C_1# \cdots #C_{\text{accept}}
   \end{bmatrix}\]. Rule 6 then allows pseudo-transitions which convert \( C_{\text{accept}} \) to the singleton character \( q_{\text{accept}} \). Lastly, rule 7 completes the top sequence to match the bottom.
Note: We made a few assumptions and simplifications:

- M never moves to the left of the input
- The first domino used must always be the starting domino.

Section 5.2 of the textbook shows how to deal with these.