Decidability
continued...
Theorem:

For a recursively enumerable language $L$ it is undecidable to determine whether $L$ is finite.

Proof:

We will reduce the halting problem to this problem.
Let $M$ be the machine that accepts $L$

$L(M) = L$

Assume we have the finite language algorithm:

Algorithm for finite language problem

YES $L(M)$ finite

NO $L(M)$ not finite
We will design the halting problem algorithm:
First construct machine $M_w$:

Initially, simulates $M$ on input $w$

When $M$ enters a halt state, accept any input (infinite language)

Otherwise accept nothing (finite language)
$M$ halts on $w$ if and only if $L(M_w)$ is not finite.
Algorithm for halting problem:

Inputs: machine $M$ and string $w$

1. Construct $M_w$

2. Determine if $L(M_w)$ is finite

   YES: then $M$ doesn’t halt on $w$

   NO: then $M$ halts on $w$
Machine for halting problem

\[ M \quad \text{construct} \quad M_w \quad \text{Check if} \quad L(M_w) \quad \text{is finite} \]

\[ M \quad w \]

YES \quad NO

YES \quad NO
Theorem:

For a recursively enumerable language $L$, it is undecidable to determine whether $L$ contains two different strings of the same length.

Proof: We will reduce the halting problem to this problem.
Assume we have the two-strings algorithm:

Let $M$ be the machine that accepts $L$

$L(M) = L$

Algorithm for two-strings problem

YES $L(M)$ contains two equal length strings

NO $L(M)$ Doesn't contain two equal length strings
We will design the halting problem algorithm:

Algorithm for Halting problem

\[ M \rightarrow \text{YES} \rightarrow M \text{ halts on } w \]

\[ w \rightarrow \text{NO} \rightarrow M \text{ doesn't halt on } w \]
First construct machine $M_w$:

Initially, simulates $M$ on input $w$

When $M$ enters a halt state, accept symbols $a$ or $b$
(two equal length strings)
$M$ halts on $w$ if and only if $M_w$ accepts $a$ and $b$ (two equal length strings)
Algorithm for halting problem:

Inputs: machine $M$ and string $w$

1. Construct $M_w$

2. Determine if $M_w$ accepts two strings of equal length

YES: then $M$ halts on $w$

NO: then $M$ doesn’t halt on $w$
Machine for halting problem

\[ M \rightarrow \text{construct } M_w \rightarrow \text{Check if } L(M_w) \text{ has two equal length strings} \]

YES \quad YES \quad NO \quad NO
The Post Correspondence Problem
Some **undecidable** problems for context-free languages:

- Is context-free grammar $G$ ambiguous?

- Is $L(G_1) \cap L(G_2) = \emptyset$?
We need a tool to prove that the previous problems for context-free languages are undecidable:

The Post Correspondence Problem
The Post Correspondence Problem

Input: Two sequences of strings

\[ A = w_1, w_2, \ldots, w_n \]

\[ B = v_1, v_2, \ldots, v_n \]
There is a Post Correspondence Solution if there is a sequence $i, j, K, k$ such that:

$$w_i w_j \Lambda w_k = v_i v_j \Lambda v_k$$

PC-solution
Example:

<table>
<thead>
<tr>
<th></th>
<th>$w_1$</th>
<th>$w_2$</th>
<th>$w_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$:</td>
<td>100</td>
<td>11</td>
<td>111</td>
</tr>
<tr>
<td>$B$:</td>
<td>001</td>
<td>111</td>
<td>11</td>
</tr>
</tbody>
</table>

PC-solution: 2,1,3

\[ w_2 w_1 w_3 = \nu_2 \nu_1 \nu_3 \]

11100111
Example:

<table>
<thead>
<tr>
<th></th>
<th>$w_1$</th>
<th>$w_2$</th>
<th>$w_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$:</td>
<td>00</td>
<td>001</td>
<td>1000</td>
</tr>
<tr>
<td>$B$:</td>
<td>$v_1$</td>
<td>$v_2$</td>
<td>$v_3$</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>11</td>
<td>011</td>
</tr>
</tbody>
</table>

There is no solution

Because total length of strings from $B$ is smaller than total length of strings from $A$. 
The Modified Post Correspondence Problem

Inputs:

\[ A = w_1, w_2, \ldots, w_n \]

\[ B = v_1, v_2, \ldots, v_n \]

MPC-solution:

\[ 1, i, j, \ldots, k \]

\[ w_1 w_i w_j \ldots w_k = v_1 v_i v_j \ldots v_k \]
Example:

\[
\begin{array}{cccc}
A: & w_1 & w_2 & w_3 \\
   & 11 & 111 & 100 \\
\end{array}
\]

\[
\begin{array}{cccc}
B: & v_1 & v_2 & v_3 \\
   & 111 & 11 & 001 \\
\end{array}
\]

**MPC-solution:** 1,3,2  \[ w_1w_3w_2 = v_1v_2v_3 \]

11100111
1. We will prove that the MPC problem is undecidable

2. We will prove that the PC problem is undecidable
1. We will prove that the MPC problem is undecidable

We will reduce the membership problem to the MPC problem
Membership problem

Input: recursive language \( L \)

string \( w \)

Question: \( w \in L \) ?

Undecidable
Membership problem

Input: unrestricted grammar $G$

string $w$

Question: $w \in L(G)$?

Undecidable
The reduction of the membership problem to the MPC problem:

For unrestricted grammar $G$ and string $w$, we construct a pair $A, B$ such that

$A, B$ has an MPC-solution if and only if $w \in L(G)$.
<table>
<thead>
<tr>
<th>$A$</th>
<th>$B$</th>
<th>Grammar $G$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$FS \Rightarrow$</td>
<td>$F$</td>
<td>$S$ : start variable</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$F$ : special symbol</td>
</tr>
<tr>
<td>$a$</td>
<td>$a$</td>
<td>For every symbol $a$</td>
</tr>
<tr>
<td>$V$</td>
<td>$V$</td>
<td>For every variable $V$</td>
</tr>
<tr>
<td>A</td>
<td>B</td>
<td>Grammar G</td>
</tr>
<tr>
<td>------------</td>
<td>------------</td>
<td>-----------</td>
</tr>
<tr>
<td>( E )</td>
<td>( \Rightarrow wE )</td>
<td>string ( w )</td>
</tr>
<tr>
<td>( y )</td>
<td>( x )</td>
<td>( E ) : special symbol</td>
</tr>
<tr>
<td>( \Rightarrow )</td>
<td>( \Rightarrow )</td>
<td>For every production</td>
</tr>
</tbody>
</table>

\( x \rightarrow y \)
Example:

**Grammar**  \( G : \)

\[
S \rightarrow aABb \mid Bbb \\
Bb \rightarrow C \\
AC \rightarrow aac
\]

**String**  \( w = aaac \)
<table>
<thead>
<tr>
<th>$A$</th>
<th>$B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_1 :$</td>
<td>$FS \Rightarrow$</td>
</tr>
<tr>
<td>$w_2 :$</td>
<td>$a$</td>
</tr>
<tr>
<td></td>
<td>$b$</td>
</tr>
<tr>
<td></td>
<td>$c$</td>
</tr>
<tr>
<td>$|,$</td>
<td>$A$</td>
</tr>
<tr>
<td></td>
<td>$B$</td>
</tr>
<tr>
<td></td>
<td>$C$</td>
</tr>
<tr>
<td>$w_8 :$</td>
<td>$S$</td>
</tr>
<tr>
<td></td>
<td>( A )</td>
</tr>
<tr>
<td>---</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>( w_9 ):</td>
<td>( E \Rightarrow )</td>
</tr>
<tr>
<td></td>
<td>( aABb )</td>
</tr>
<tr>
<td>( \Gamma )</td>
<td>( Bbb )</td>
</tr>
<tr>
<td></td>
<td>( C )</td>
</tr>
<tr>
<td></td>
<td>( aac )</td>
</tr>
<tr>
<td>( w_{14} ):</td>
<td>( \Rightarrow )</td>
</tr>
</tbody>
</table>
Grammar $ G : \quad S \rightarrow aABb \mid Bbb $

$ Bb \rightarrow C $

$ AC \rightarrow aac $

$ aaac \in L(G) $

$ S \Rightarrow aABb \Rightarrow aAC \Rightarrow aaac $
\[ S \Rightarrow aAAb \]
\[ S \Rightarrow aABb \Rightarrow aAC \]
$S \Rightarrow aABb \Rightarrow aAC \Rightarrow aaac$

A

$\begin{align*}
F & \Rightarrow a \\
S & \Rightarrow A \\
B & \Rightarrow b \\
b & \Rightarrow a \\
A & \Rightarrow C \\
C & \Rightarrow a \\
a & \Rightarrow a \\
a & \Rightarrow a \\
a & \Rightarrow c \\
E & \Rightarrow 
\end{align*}$

B

$\begin{align*}
\nu_1 & \Rightarrow \\
\nu_{10} & \Rightarrow \\
\nu_{14} & \Rightarrow \\
\nu_2 & \Rightarrow \\
\nu_5 & \Rightarrow \\
\nu_{12} & \Rightarrow \\
\nu_{14} & \Rightarrow \\
\nu_2 & \Rightarrow \\
\nu_{13} & \Rightarrow \\
\nu_9 & \Rightarrow 
\end{align*}$
Theorem:

\[(A, B)\] has an MPC-solution if and only if \( w \in L(G) \)
Algorithm for membership problem:

Input: unrestricted grammar $G$
string $w$

Construct the pair $A, B$

If $A, B$ has an MPC-solution then $w \in L(G)$
else $w \notin L(G)$
Membership machine

\[
G \rightarrow \text{construct } A,B \rightarrow A,B \rightarrow \text{MPC algorithm } \rightarrow \begin{cases} \text{solution } w \in L(G) \\ \text{No-solution } w \notin L(G) \end{cases}
\]
2. We will prove that the PC problem is undecidable

We will reduce the MPC problem to the PC problem
$A, B$ : input to the MPC problem

\[ A = w_1, w_2, K, w_n \]

\[ B = v_1, v_2, K, v_n \]
We construct new sequences $C, D$

$C = w'_0, w'_1, K, w'_n, w'_{n+1}$

$D = v'_0, v'_1, K, v'_n, v'_{n+1}$

$A = w_1, w_2, K, w_n$

$B = v_1, v_2, K, v_n$
We insert a special symbol between any two symbols
B \quad v_i = abcad

D \quad v'_i = *a* b* c* a* d
Special Cases

\begin{align*}
C & \\
  w'_0 &= *w_1 \\
  w'_{n+1} &= \diamond
\end{align*}

\begin{align*}
D & \\
  v'_0 &= v'_1 \\
  v'_{n+1} &= \ast\diamond
\end{align*}
Observation:

There is a PC-solution for $C, D$ if and only if there is a MPC-solution for $A, B$. 
**PC-solution** \[ w'_0 \Lambda w'_k w'_{n+1} = v'_0 \Lambda w'_k v'_{n+1} \]

**MPC-solution** \[ w_1 \Lambda w_k = v_1 \Lambda v_k \]
MPC-algorithm

Input: sequences $A, B$

Construct sequences $C, D$

Solve the PC problem for $C, D$
MPC algorithm

$A, B \rightarrow C, D \rightarrow \text{construct} C, D \rightarrow \text{PC algorithm} \rightarrow \text{solution} \rightarrow \text{No-solution}$