Thm (1): If L is recursive, so is $\bar{L} = \Sigma^* - L$.

Proof: If L is recursive, then it is decided by M. Let M' be the following modification of M: on input x, M' runs just like M.

However, when M is about to enter an accepting state, M' enters a new state q_{reject}, and when M is about to halt in a non-accepting state, M' enters a new state q_{accept}.

Clearly, $\bar{L} = L(M')$. Since M always halts, so does M', so by definition, $\bar{L}$ is recursive.

Thm (2): If both L and $\bar{L}$ are RE, then L is recursive.

Proof: Let $L = L(M_1)$ and $\bar{L} = L(M_2)$. Let M be the following TM: on input x it simulates M_1 on x on one tape, and M_2 on x on another tape.

Since $L \cup \bar{L}$ =everything, x must be accepted by one or the other.

If M_1 accepts, so does M,

if M_2 accepts, M rejects.

L_U, the **Universal Language**, is the set of binary strings that encode pairs (M, w) such that $w \in L(M)$.

That is, L_U is the set of all those strings of the form $w_i 111 w_j$ where the TM M_i accepts the input w_j.

We show that L_U is accepted by U, the **Universal Turing machine**, i.e., $L_U = L(U)$.

U is simply a TM which on input $w_i 111 w_j$ starts simulating the TM M_i on w_j and accepts iff M_i accepts. If the input to U, say x, is not of the form $w_i 111 w_j$, then it just rejects.

Thm: L_U is RE but not recursive.

Proof: Since $L_U = L(U)$, by definition L_U is RE.

Suppose that L_U is also recursive.
Then so is $\bar{L}_U$ by Theorem 1.
Then there exists a TM M which decides $\bar{L}_U$, i.e., M always halts, and $\bar{L}_U = L(M)$.

Let M' be a TM which does the following: on input w_i, M' runs just like M would on w_i111w_i. *Since M exists (by assumption) so does M'.*

M accepts w_i111w_i if M_i does not accept w_i (because $L(M) = \bar{L}_U$).

Therefore, M' accepts w_i iff M_i does not accept w_i, and THEREFORE, $L(M') = L_d$.

But there is no such M' (By Theorem 0)! Contradiction, so L_U is not recursive.

So we have L_d, L_U where the first is not RE, and the second is RE but not recursive.

Using the notion of **reduction** we can show that many other languages are not RE or not recursive.

$L_e = \{w | w = w_i, L(M_i) = \emptyset\}$, $L_{ne} = \{w | w = w_i, L(M_i) \neq \emptyset\}$

Note that $L_{ne} = \overline{L_e}$.

L_{ne} is RE; let N be a nondet TM which on input $w = w_i$, "guesses" some string x, and "simulates" M_i on x (it uses U as a subroutine for this simulation).

If $M_i(x) =$ accept, then N accepts w, $\therefore L_{ne} = L(N)$.

On the other hand, L_{ne} is not recursive (and hence L_e is not RE).

We use L_U to show L_{ne} is not recursive.

For a given pair (M, w) construct the following TM M':

On input x, ignore x and instead simulate $M(w)$. If $M(w) =$ accept, then accept.

There are two possibilities:

1. $M(w) =$ accept, so $L(M') = \Sigma^*$,
2. M does not accept w, so $L(M') = \emptyset$.

$\therefore (M, w) \in L_U \iff L(M') \neq \emptyset$

$\therefore$ if we could decide L_{ne}, then we could also decide L_U, which we know we cannot.

It turns out that all nontrivial properties of RE languages are undecidable, in the sense that the language consisting of codes of TMs having this property is not recursive.

E.g., the language consisting of codes of TMs whose languages are empty (i.e., L_e) is not recursive.

A **property** of RE languages is simply a subset of RE. A property is **trivial** if it is empty or if it is everything.

If $\mathcal{P}$ is a property of RE languages, the language $L_{\mathcal{P}}$ is the set of codes for TMs M_i s.t. $L(M_i) \in \mathcal{P}$.

When we talk about the decidability of $\mathcal{P}$, we formally mean the decidability of $L_{\mathcal{P}}$.

Rice's Thm: Every nontrivial property of RE languages is undecidable.

Proof: Suppose $\mathcal{P}$ is nontrivial. Assume $\emptyset \notin \mathcal{P}$ (if it is, consider $\overline{\mathcal{P}}$ which is also nontrivial).

Since $\mathcal{P}$ is nontrivial, some $L \in \mathcal{P}$, $L \neq \emptyset$.

Let M_L be the TM accepting L.

For a fixed pair (M, w) consider the TM M': on input x, it first simulates $M(w)$, and if it accepts, it simulates $M_L(x)$, and if that accepts, M' accepts.

$\therefore$ $L(M') = \emptyset \notin \mathcal{P}$ if M does not accept w, and $L(M') = L \in \mathcal{P}$ if M accepts w.

Thus, $L(M') \in \mathcal{P} \iff (M, w) \in L_U$, $\therefore$ $\mathcal{P}$ is undecidable.

Post's Correspondence Problem (PCP)

An instance of **PCP** consists of two finite lists of strings over some alphabet Σ. The two lists must be of equal length:

$A = w_1, w_2, \ldots, w_k$
$B = x_1, x_2, \ldots, x_k$

For each i, the pair (w_i, x_i) is said to be a *corresponding pair*. We say that this instance of PCP has a solution if there is a sequence of one or more indices:

$$i_1, i_2, \ldots, i_m \qquad m \geq 1$$

such that:

$$w_{i_1} w_{i_2} \ldots w_{i_m} = x_{i_1} x_{i_2} \ldots x_{i_m}$$

The PCP is: given (A, B), tell whether there is a solution.

Aside: To express **PCP** as a language, we let L_{PCP} be the language:

$$\{\langle A, B\rangle | (A, B) \text{ instance of PCP with solution}\}$$

Example: Consider (A, B) given by:

$A = 1, 10111, 10$
$B = 111, 10, 0$

Then $2, 1, 1, 3$ is a solution as:

$$\underbrace{10111}_{w_2}\underbrace{1}_{w_1}\underbrace{1}_{w_1}\underbrace{10}_{w_3} = \underbrace{10}_{x_2}\underbrace{111}_{x_1}\underbrace{111}_{x_1}\underbrace{0}_{x_3}$$

Note that $2, 1, 1, 3, 2, 1, 1, 3$ is another solution.

On the other hand, you can check that: $A = 10, 011, 101$ & $B = 101, 11, 011$ Does not have a solution.

Modified PCP (MPCP)

The **MPCP** has an additional requirement that the first pair in the solution must be the first pair of (A, B).

So $i_1, i_2, \ldots, i_m$, $m \geq 0$, is a solution to the (A, B) instance of **MPCP** if:

$$w_1 w_{i_1} w_{i_2} \ldots w_{i_m} = x_1 x_{i_1} x_{i_2} \ldots x_{i_m}$$

Def: We say that $i_1, i_2, \ldots, i_r$ is a **partial solution** of PCP if one of the following is the prefix of the other:

$$w_{i_1} w_{i_2} \ldots w_{i_r} \qquad x_{i_1} x_{i_2} \ldots x_{i_r}$$

Same def holds for MPCP, but w_1, x_1 must be at the beginning.

We now show:

1. If **PCP** is decidable, then so is **MPCP**.
2. If **MPCP** is decidable, then so is L_U.
3. Since L_U is *not* decidable, neither is **(M)PCP**.

$$\boxed{\textbf{PCP} \text{ decidable} \Longrightarrow \textbf{MPCP} \text{ decidable}}$$

We show that given an instance (A, B) of MPCP, we can construct an instance (A', B') of PCP such that:

$$(A, B) \text{ has solution} \iff (A', B') \text{ has solution}$$

Let (A, B) be an instance of MPCP over the alphabet Σ. Then (A', B') is an instance of PCP over the alphabet $\Sigma' = \Sigma \cup \{*, \$\}$.

If $A = w_1, w_2, w_3, \ldots, w_k$, then
$A' = *\mathbf{w}_1*, \mathbf{w}_1*, \mathbf{w}_2*, \mathbf{w}_3*, \ldots, \mathbf{w}_k*, \$$.

If $B = x_1, x_2, x_3, \ldots, x_k$, then $B' = *\mathbf{x}_1, *\mathbf{x}_1, *\mathbf{x}_2, *\mathbf{x}_3, \ldots, *\mathbf{x}_k, *\$$.

where if $x = a_1a_2a_3 \ldots a_n \in \Sigma^*$, then $\mathbf{x} = a_1 * a_2 * a_3 * \ldots * a_n$.

For example: If (A, B) is an instance if MPCP given as:

$A = 1, 10111, 10$
$B = 111, 10, 0$

Then (A', B') is an instance of PCP given as follows:

$A' = *1*, 1*, 1*0*1*1*1*, 1*0*, \$$
$B' = *1*1*1, *1*1*1, *1*0, *0, *\$$

$\boxed{\textbf{MPCP} \text{ decidable} \Longrightarrow L_U \text{ decidable}}$

Given a pair (M, w) we construct an instance (A, B) of MPCP such that:

TM M accepts $w \iff (A, B)$ has a solution.

Idea: The MPCP instance (A, B) simulates, in its partial solutions, the computation of M on w.

That is, partial solutions will be of the form:

$$\#\alpha_1\#\alpha_2\#\alpha_3\#\ldots$$

where α_1 is the initial ID of M on w, and for all i, $\alpha_i \vdash \alpha_{i+1}$.

The string from the B list will always be one ID ahead of the A list; the A list will be allowed to "catch-up" only when M accepts w.

To simplify things, we assume the result of Theorem 8.12 which says that we may assume that our TM M:

1. Never prints a blank.

2. Never moves left from its initial head position.

The IDs of M will always be of the form $\alpha q \beta$, where α, β are non-blank tape symbols and q is a state.

Let M be a TM satisfying Theorem 8.12, and $w \in \Sigma^*$. We construct an instance (A, B) of MPCP as follows:

1. A: $\#$
   B: $\#q_0w\#$

2. A: $X_1, X_2, \ldots, X_n, \#$
   B: $X_1, X_2, \ldots, X_n, \#$
   where the X_i are all the tape symbols.

3. To simulate a move of M, for all non-accepting $q \in Q$:

list A	list B	
qX	Yp	if $\delta(q, X) = (p, Y, \rightarrow)$
ZqX	pZY	if $\delta(q, X) = (p, Y, \leftarrow)$
$q\#$	$Yp\#$	if $\delta(q, B) = (p, Y, \rightarrow)$
$Zq\#$	$pZY\#$	if $\delta(q, B) = (p, Y, \leftarrow)$

4. If the ID at the end of B has an accepting state, then we need to allow A to catch up with B. So we need for all accepting states q, and all symbols X, Y:

list A	list B
XqY	q
Xq	q
qY	q

5. Finally, after using 4 and 3 above, we end up with $x\#$ and $x\#q\#$, where x is a long string. Thus we need $q\#\#$ in A and $\#$ in B to complete the catching up.

Ex. $\delta(q_1, 0) = (q_2, 1, \rightarrow), \delta(q_1, 1) = (q_2, 0, \leftarrow), \delta(q_1, B) = (q_2, 1, \leftarrow)$
$\delta(q_2, 0) = (q_3, 0, \leftarrow), \delta(q_2, 1) = (q_1, 0, \rightarrow), \delta(q_2, B) = (q_2, 0, \rightarrow)$

Rule	list A	list B	Source
1	#	$\#q_1 01\#$	
2	0	0	
	1	1	
	#	#	
3	$q_1 0$	$1q_2$	$\delta(q_1, 0) = (q_2, 1, \rightarrow)$
	$0q_1 1$	$q_2 00$	$\delta(q_1, 1) = (q_2, 0, \leftarrow)$
	$1q_1 1$	$q_2 10$	$\delta(q_1, 1) = (q_2, 0, \leftarrow)$
	$0q_1\#$	$q_2 01\#$	$\delta(q_1, B) = (q_2, 1, \leftarrow)$
	$1q_1\#$	$q_2 11\#$	$\delta(q_1, B) = (q_2, 1, \leftarrow)$
	$0q_2 0$	$q_3 00\#$	$\delta(q_2, 0) = (q_3, 0, \leftarrow)$
	$1q_2 0$	$q_3 10\#$	$\delta(q_2, 0) = (q_3, 0, \leftarrow)$
	$q_2 1$	$0q_1$	$\delta(q_2, 1) = (q_1, 0, \rightarrow)$
	$q_2\#$	$0q_2\#$	$\delta(q_2, B) = (q_2, 0, \rightarrow)$
4	$0q_3 0$	q_3	
	$0q_3 1$	q_3	
	$1q_3 0$	q_3	
	$1q_3 1$	q_3	
	$0q_3$	q_3	
	$1q_3$	q_3	
	$q_3 0$	q_3	
	$q_3 1$	q_3	
5	$q_3\#\#$	#	

The TM M accepts the input 01 by the sequence of moves:

$$q_1 01 \vdash 1q_2 1 \vdash 10q_1 \vdash 1q_2 01 \vdash q_3 101$$

We examine the sequence of partial solutions that mimics this computation of M and eventually leads to a solution.

We must start with the first pair (MPCP):

A: $\#$
B: $\#q_1 01\#$

The only way to extend this partial solution is with the corresponding pair $(q_1 0, 1q_2)$, so we obtain:

A: $\#q_1 0$
B: $\#q_1 01\#1q_2$

Now using copying pairs we obtain:

A: $\#q_1 01\#1$
B: $\#q_1 01\#1q_2 1\#1$

Next corresponding pair is $(q_2 1, 0q_1)$:

A: $\#q_1 01\#1q_2 1$
B: $\#q_1 01\#1q_2 1\#10q_1$

Now careful! We only copy the next two symbols to obtain:

A: $\#q_1 01\#1q_2 1\#1$
B: $\#q_1 01\#1q_2 1\#10q_1\#1$

because we need the $0q_1$ as the head now moves left, and use the next appropriate corresponding pair which is $(0q_1\#, q_2 01\#)$ and obtain:

A: $\#q_1 01\#1q_2 1\#10q_1\#$

B: $\#q_1 01\#1q_2 1\#10q_1\#1q_2 01\#$

We can now use another corresponding pair $(1q_20, q_310)$ right away to obtain:

A: $\#q_101\#1q_21\#10q_1\#1q_20$
B: $\#q_101\#1q_21\#10q_1\#1q_201\#q_310$

and note that we have an accepting state! We use two copying pairs to get:

A: $\#q_101\#1q_21\#10q_1\#1q_201\#$
B: $\#q_101\#1q_21\#10q_1\#1q_201\#q_3101\#$

and we can now start using the rules in 4. to make A catch up with B:

A: $\ldots\#q_31$
B: $\ldots\#q_3101\#q_3$

and we copy three symbols:

A: $\ldots\#q_3101\#$

B: $\ldots\#q_3101\#q_301\#$

And again catch up a little:

A: $\ldots\#q_3101\#q_30$
B: $\ldots\#q_3101\#q_301\#q_3$

Copy two symbols:

A: $\ldots\#q_3101\#q_301\#$
B: $\ldots\#q_3101\#q_301\#q_31\#$

and catch up:

A: $\ldots\#q_3101\#q_301\#q_31$
B: $\ldots\#q_3101\#q_301\#q_31\#q_3$

and copy:

A: $\ldots\#q_3101\#q_301\#q_31\#$
B: $\ldots\#q_3101\#q_301\#q_31\#q_3\#$

And now end it all with the corresponding pair $(q_3\#\#, \#)$ given by rule 5. to get matching strings:

A: $\ldots \# q_3 101 \# q_3 01 \# q_3 1 \# q_3 \#\#$
B: $\ldots \# q_3 101 \# q_3 01 \# q_3 1 \# q_3 \#\#$

THEREFORE: we reduced L_U to the MPCP. Now, we can solve L_U by producing a carefully crafted instance of MPCP (A, B), and asking if it has a solution. If yes, then we know that M accepts w.

Since we have already shown that L_U is undecidable, MPCP must also be undecidable. Thus, PCP is undecidable.

NEXT: We can now use the fact that PCP is undecidable to show that a number of questions about CFLs are undecidable.

Let $A = w_1, w_2, \ldots, w_k$, let G_A be the related CFG given by:

$$A \longrightarrow w_1 A a_1 | w_2 A a_2 | \cdots | w_k A a_k | w_1 a_1 | w_2 a_2 | \cdots | w_k a_k$$

Let $L_A = L(G_A)$, the language of the list A, and $a_1, a_2, \ldots, a_k$ are distinct index symbols not in alphabet of A.

The terminal strings of G_A are of the form:

$$w_{i_1} w_{i_2} \ldots w_{i_m} a_{i_m} \ldots a_{i_2} a_{i_1}$$

Let G_{AB} be a CFG consisting of G_A, G_B, with $S \longrightarrow A|B$.

$\therefore$ G_{AB} is ambiguous $\iff$ the PCP (A, B) has a solution.

Thm: It is undecidable whether a CFG is ambiguous.

$\overline{L_A}$ is also a CFL; we show this by giving a PDA P.

$\Gamma_P = \Sigma_A \cup \{a_1, a_2, \ldots, a_k\}$.

As long as P sees a symbol in Σ_A it stores it on the stack.

As soon as P sees a_i, it pops the stack to see if top of string is w_i^R. (i) if not, then accept no matter what comes next. (ii) if yes, there are two subcases:

(iia) if stack is not yet empty, continue.

(iib) if stack is empty, and the input is finished, reject.

If after an a_i, P sees a symbol in Σ_A, it rejects.

Thm: G_1, G_2 are CFGs, and R is a reg. exp., then the following are undecidable problems:

1. $L(G_1) \cap L(G_2) \stackrel{?}{=} \emptyset$
2. $L(G_1) \stackrel{?}{=} L(G_2)$
3. $L(G_1) \stackrel{?}{=} L(R)$
4. $L(G_1) \stackrel{?}{=} T^*$
5. $L(G_1) \stackrel{?}{\subseteq} L(G_2)$
6. $L(R) \stackrel{?}{\subseteq} L(G_2)$

Proofs: 1. Let $L(G_1) = L_A$ and $L(G_2) = L_B$, then $L(G_1) \cap L(G_2) = \emptyset$ iff PCP (A, B) has a solution.

2. Let G_1 be the CFG for $\overline{L_A} \cup \overline{L_B}$ (CFGs are closed under union). Let G_2 be the CFG for the reg. lang. $(\Sigma \cup \{a_1, a_2, \ldots, a_k\})^*$.

Note $L(G_1) = \overline{L_A} \cup \overline{L_B} = \overline{L_A \cap L_B} =$ everything but solutions to PCP (A, B).

$\therefore$ $L(G_1) = L(G_2)$ iff (A, B) has no solution.

3. Shown in 2.

4. Again, shown in 2.

5. Note that $A = B$ iff $A \subseteq B$ and $B \subseteq A$, so it follows from 2.

6. By 3. and 5.