Chapter 9

Undecidability

(part c)

(2015/12/25)

Coliseum in Rome, Italy 1998

Outline

9.0 Introduction (in part a)

9.1 A Language That Is Not Recursively Enumerable (in part a)

9.2 An Undecidable Problem That Is RE (in part a)

9.3 Undecidable Problems about TM's (in part b)

9.4 Post Correspondence Problem (in this part)

9.5 Other Undecidable Problems (in this part)

9.4 Post's Correspondence Problems

- **Concepts to be taught ---**
  - We will study Post's correspondence problem (PCP) which involve strings rather than TM's.
  - We will define a *modified* PCP (MPCP)
  - We will reduce MPCP to the original PCP.
  - We will also reduce L_u to the MPCP.
  - So L_u is reduced in two steps to PCP, thus proving PCP undecidable.
  - The double reductions from L_u to PCP may be illustrated by Figure 9.16 (Figure 9.11 in the textbook):

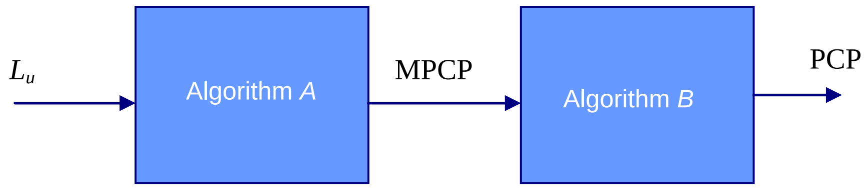

Figure 9.16 Double reductions from L_u to PCP.

9.4.1 Definition of PCP

- **Definition ---**

An instance of PCP consists of two lists of strings over some alphabet Σ, where

- the two lists are of equal length, denoted as *A* and *B*;
- the instance is denoted as (A, B);
- we write them as $A = w_1, w_2, \ldots, w_k$, $B = x_1, x_2, \ldots, x_k$ for some integer k;
- for each i, the pair (w_i, x_i) is said a *corresponding pair*;
- We say this instance of PCP has a *solution*, if there is a sequence of integers, $i_1, i_2, \ldots, i_m$, that, when interpreted as indexes for strings in the *A* and *B* lists, yields the same string, that is, $w_{i_1}w_{i_2}\ldots w_{i_m} = x_{i_1}x_{i_2}\ldots x_{i_m}$.
- We say the sequence is a *solution* to this instance of PCP, if so.

- **Definition ---**

The Post's corresponding problem is: *given an instance of PCP, tell whether this instance has a solution.*

- **Properties of Post's corresponding problem ---**
  - The solution to an instance of PCP sometimes is *not* unique.
  - Also, an instance of PCP might have no solution. See Example 9.14.

- **Example 9.13 ---**

Two lists of an instance of PCP are shown in Fig. 9.17. Find a solution for it.

- A solution is 2, 1, 1, 3 because

$$w_2w_1w_1w_3 = 101111110 = 101111110 = x_2x_1x_1x_3$$

♦ Another solution is 2, 1, 1, 3, 2, 1, 1, 3.

	List *A*	List *B*
i	w_i	x_i
1	1	111
2	10111	10
3	10	0

Fig. 9.17 Two lists of an instance of PCP.

9.4.2 The Modified PCP (MPCP)

■ **Definition ---**

In the MPCP, it is additionally required that the first pair on lists *A* & *B* must be the first pair in the solution.

♦ That is, for lists $A = w_1, w_2, \ldots, w_k$, $B = x_1, x_2, \ldots, x_k$, the solution is a list $i_1, i_2, \ldots, i_m$ such that

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

■ **Example 9.15 ---**

If Fig.9.17 is used as an instance of MPCP, then it has no solution.

■ **Reduction of MPCP to PCP ---**

We now show how to reduce MPCP to PCP. We construct an instance of PCP from an instance of MPCP as follows:

♦ Introduce two new symbols * and $:
♦ In list *A*, * follows each symbol in the string;
♦ In list *B*, * precedes each symbol in the string.
♦ Put an extra * before list *A*.
♦ A final pair ($, *$) is added to the PCP instance.

(The purpose of using extra symbols: to make lengths equal and put an end mark $)

■ **Example 9.16 ---**

Suppose Fig. 9.17 is an MPCP instance. Then, the corresponding PCP instance constructed in the above way is as shown in Fig. 9.18.

■ **Theorem 9.17 ---**

MPCP reduces to PCP.

Proof.

	List A	List B
i	y_i	z_i
0	*1*	*1*1*1
1	1*	*1*1*1
2	1*0*1*1*1*	*1*0
3	1*0*	*0
4	$	*$

Fig. 9.18 A PCP instance for the MPCP shown in Figure 9.17.

Proof of the "if" part ---

- ♦ Suppose $i_1, i_2, \ldots, i_m$ is the solution to the given MPCP instance with lists A and B.
- ♦ Accordingly, we have $w_1 w_{i_1} w_{i_2} \ldots w_{i_m} = x_1 x_{i_1} x_{i_2} \ldots x_{i_m}$.
- ♦ Now, replacing w's by y's and x's by z's constructed as above (see Example 9.16), we get $y_1 y_{i_1} y_{i_2} \ldots y_{i_m} = z_1 z_{i_1} z_{i_2} \ldots z_{i_m}$ which are *almost* the same.
- ♦ The difference is: the 1st string is missing a * at the beginning, and the 2nd is missing a * at the end.
- ♦ We know $y_0 = {}^*y_1$, $z_0 = z_1$, $y_{k+1} = \$$, $z_{k+1} = {}^*\$$ which, when used to replace the initials and ends of the two strings, yield $y_0\, y_{i_1} y_{i_2} \ldots y_{i_m} y_{k+1} = z_0\, z_{i_1} z_{i_2} \ldots z_{i_m} z_{k+1}$.
- ♦ That is, $0, i_1, i_2, \ldots, i_m, k+1$ is a solution to the instance of PCP.

Proof of the "only if" part ---

- ♦ If the instance of PCP constructed as above has a solution, then since
  - only the 0th pair has strings y_0 and z_0 that begin with the same symbol *; and
  - only the $(k+1)$st pair has strings that end with the same symbol $,

  we get to know that the solution is of the form $0, i_1, i_2, \ldots, i_m, k+1$, which means $y_0\, y_{i_1} y_{i_2} \ldots y_{i_m} y_{k+1} = z_0\, z_{i_1} z_{i_2} \ldots z_{i_m} z_{k+1}$.
- ♦ Now remove all *'s and $'s from the two sides, we can get $w_1 w_{i_1} w_{i_2} \ldots w_{i_m} = x_1 x_{i_1} x_{i_2} \ldots x_{i_m}$ which is a solution to the MPCP instance.
- ♦ Therefore, the construction above of an MPCP instance from a PCP instance is a reduction from MPCP to PCP. **Done**.

9.4.3 <u>Completion of Proof of PCP Undecidability</u>

■ **Reducing L_u to MPCP ---**

Now, we want to reduce L_u to MPCP.

- ♦ For this, give a pair (M, w), we construct an instance (A, B) of MPCP such that TM M accepts w if and only if (A, B) has a solution.
- ♦ The construction is essentially to use the MPCP instance (A, B) to simulate, in its partial solutions, the computation of M on input w.
- ♦ The partial solutions will consist of strings that are prefixes of the sequence of the ID's of M,

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

  where
- ♦ α_1 is the initial ID of M;

- ♦ $\alpha_i \vdash \alpha_{i+1}$ for all i.
- ♦ The string from the list B will always one ID ahead of the string from the list A, unless M enters an accepting state.
- ♦ In that case, there will be pairs to use for A to catch up to B.
- ♦ But if no accepting state is entered, then these "catching up" pairs will not be used, and so no solution can be found.
- ♦ We will assume the TM used is one with semi-finite (one-sided) tape with no blank printed on the tape (see Theorem 8.12). Such a TM is equivalent to the original TM.
- ♦ Given a TM of this type, $M = (Q, \Sigma, \Gamma, d, q_0, B, F)$, with input $w \in \Sigma^*$, we construct an instance of MPCP as follows.

1. The first pair for TM initialization (simulating initial ID):

List A	List B
#	$\#q_0w\#$

2. Tape symbols and the separator # can be appended to both lists:

List A	List B
X	X
#	#

3. Simulation of moves of M:

for all $q \in (Q - F)$ (non-accepting state), $p \in Q$, and $X, Y, Z \in \Gamma$ and B the blank symbol:

List A	List B	
qX	Yp	if $\delta(q, X) = (p, Y, R)$
ZqX	pZY	if $\delta(q, X) = (p, Y, L)$; $Z \in \Gamma$
$q\#$	$Yp\#$	if $\delta(q, \text{B}) = (p, Y, R)$
$Zq\#$	$pZY\#$	if $\delta(q, \text{B}) = (p, Y, L)$; $Z \in \Gamma$

4. Accepting:

for each final state q and for all possible tape symbols X and Y in Γ:

List A	List B
XqY	q
Xq	q
qY	q

5. Appending the ending symbols:

for each final state q:

List A	List B
$q\#\#$	#

■ **Example 9.18 ---**

Given TM M with its transition table as shown in Table 9.1 and final state q_3 with input $w = 01$, the corresponding MPCP instance is shown in Figure 9.19.

- ♦ The moves of M to accept input 01 is

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

Table 9.1 Transition table of TM in Example 9.18.

q_i	$\delta(q_i, 0)$	$\delta(q_i, 1)$	$\delta(q_i, B)$
q_1	$(q_2, 1, R)$	$(q_2, 0, L)$	$(q_2, 1, L)$
q_2	$(q_3, 0, L)$	$(q_1, 0, R)$	$(q_2, 0, R)$
q_3	-	-	-

Rule	List A	List B	Source
(1)	#	$\#q_1 01\#$	
(2)	0	0	
	1	1	
	#	#	
(3)	$q_1 0$	$1q_2$	from $(q_1, 0)=(q_2, 1, R)$
	$0q_1 1$	$q_1 00$	from $(q_1, 1)=(q_2, 0, L)$
	$1q_1 1$	$q_1 10$	from $(q_1, 1)=(q_2, 0, L)$
	$0q_1\#$	$q_1 01\#$	from $(q_1, B)=(q_2, 1, L)$
	$1q_1\#$	$q_1 11\#$	from $(q_1, B)=(q_2, 1, L)$
	$0q_2 0$	$q_3 00$	from $(q_2, 0)=(q_3, 0, L)$
	$1q_2 0$	$q_3 10$	from $(q_2, 0)=(q_3, 0, L)$
	$q_2 1$	$0q_1$	from $(q_2, 1)=(q_1, 0, R)$
	$q_2\#$	$0q_2\#$	from $(q_2, B)=(q_2, 0, R)$
(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)	$q_3\#\#$	#	

Figure 9.19 MPCP instance of Example 9.18.

♦ The sequence of partial solutions which mimics the above moves of M is shown below:

Derivations		*Used rule*
(Initialization)		
	A: #	(1)
	B: $\#q_1 01\#$	(1)
(moves and copying)		
⇒	A: $\#q_1 0$	(3.1)
	B: $\#q_1 01\#1q_2$	(3.1)
⇒	A: $\#q_1 01\#1$	(2.2)(2.3)(2.2)
	B: $\#q_1 01\#1q_2 1\#1$	(2.2)(2.3)(2.2)
⇒	A: $\#q_1 01\#1q_2 1$	(3.8)
	B: $\#q_1 01\#1q_2 1\#10q_1$	(3.8)
⇒	A: $\#q_1 01\#1q_2 1\#1$	(2)
	B: $\#q_1 01\#1q_2 1\#10q_1\#1$	(2)
⇒	A: $\#q_1 01\#1q_2 1\#10q_1\#$	(3.4)
	B: $\#q_1 01\#1q_2 1\#10q_1\#1q_2 01\#$	(3.4)
⇒	A: $\#q_1 01\#1q_2 1\#10q_1\#1q_2 0$	(3.7)
	B: $\#q_1 01\#1q_2 1\#10q_1\#1q_2 01\#q_3 10$	(3.7)
⇒	A: $\#q_1 01\#1q_2 1\#10q_1\#1q_2 01\#$	(2)
	B: $\#q_1 01\#1q_2 1\#10q_1\#1q_2 01\#q_3 101\#$	(2)
(start to eliminate all symbols but q_3 in list B)		

⇒ A: $\#q_1 01\#1q_2 1\#10q_1\#1q_2 01\#q_3 101\#$ (4.8)(2)

B: $\#q_1 01\#1q_2 1\#10q_1\#1q_2 01\#q_3 101\#q_3 01\#$ (4.8)(2)

⇒ A: $\#q_1 01\#1q_2 1\#10q_1\#1q_2 01\#q_3 101\#q_3 01\#$ (4.7)(2)

B: $\#q_1 01\#1q_2 1\#10q_1\#1q_2 01\#q_3 101\#q_3 01\#q_3 1\#$ (4.7)(2)

⇒ A: $\#q_1 01\#1q_2 1\#10q_1\#1q_2 01\#q_3 101\#q_3 01\#q_3 1\#$ (4.8)(2)

B: $\#q_1 01\#1q_2 1\#10q_1\#1q_2 01\#q_3 101\#q_3 01\#q_3 1\#q_3\#$ (4.8)(2)

⇒ A: $\#q_1 01\#1q_2 1\#10q_1\#1q_2 01\#q_3 101\#q_3 01\#q_3 1\#q_3\#\#$ (5)

B: $\#q_1 01\#1q_2 1\#10q_1\#1q_2 01\#q_3 101\#q_3 01\#q_3 1\#q_3\#\#$ (5)

Done!

- ♦ Therefore, there is a solution for this instance.
- ♦ Try to see the partial solutions for an input which is not accepted by *M* -- the two sets of rules (4) and (5) will not be used, and so lists *A* and *B* will not be of the same length, implying that no solution is possible.

■ **Theorem 9.19 ---**

PCP is undecidable.

Proof.

■ We still have to complete the reduction of *Lu* to MPCP. For this, we want to prove:

M accepts *w* if and only if the constructed MPCP instance has a solution.

Proof of the "if" part ---

- ♦ Example 9.18 gives the fundamental idea of proof of this part. If *w* is in *L*(*M*), then we can use the rules of (1) through (5) to generate a solution for the MCPC instance.

Proof of the "only if" part ---

- ♦ If the MPCP instance has a solution, then it could only be because *M* accepts *w*. If not accepting, then the final partial solution will not be of the same length because rules (4) and (5) will not used.

(For more details, see the textbook.)

9.5 Other Undecidable Problems

■ **Concepts to be taught ---**

- ♦ We may reduce PCP to a variety of other problems that we wish to prove undecidable.

9.5.1 Problems about Programs

■ **Reduction of PCP to computer programs ---**

- ♦ We may write a computer program that takes an instance of PCP (encoded as a string) and searches for solutions in a certain systematic manner (e.g., in order of lengths of partial solutions).
- ♦ When the PCP finds a solution, we can then have the program do any particular thing we want, e.g., print *hello world*, call a particular function, ring the console bell, etc.
- ♦ This completes the reduction.
- ♦ Therefore, problems about such things are *undecidable*.

♦ **Analog of Rice Theorem for programs ---**

Any nontrivial property that involves what the program does is undecidable.

9.5.2 <u>Undecidability of Ambiguity for CFG's</u>

- **Concepts to be taught ---**
  - We will prove the problem of deciding if a given CFG is ambiguous is undecidable.
  - We will also prove several problems about the CFG's undecidability.

- **Some definitions ---**
  - Let a PCP instance consists of lists $A = w_1, w_2, \ldots, w_k$, $B = x_1, x_2, \ldots, x_k$.
  - We construct a CFG with A as the only variable.
  - The terminals are all the symbols of the alphabet   used for this PCP instance, plus a distinct set of index symbols $a_1, a_2, \ldots, a_k$ that represent the choices of pairs of strings in a solution to the PCP instance.
  - That is, a_i means the choice of w_i from list A *or* x_i from list B.
  - The productions for the CFG for list A are:

    $$A \rightarrow w_1Aa_1 \mid w_2Aa_2 \mid \ldots \mid w_kAa_k \mid \ldots \mid w_1a_1 \mid w_2a_2 \mid \ldots \mid w_ka_k.$$

  - Call this grammar G_A and its language L_A.
  - L_A is called the language of list A.
  - Similarly, we construct a grammar G_B with language L_B from list B with the following productions:

    $$B \rightarrow x_1Ba_1 \mid x_2Ba_2 \mid \ldots \mid x_kBa_k \mid \ldots \quad \mid x_1a_1 \mid x_2a_2 \mid \ldots \mid x_ka_k.$$

  - L_B is called the language of list B.
  - Finally, we define a grammar G_{AB} by combining grammars G_A and G_B for the entire PCP instance, which consists of:
    - variables A, B, and S (the start symbol);
    - productions $S \rightarrow A \mid B$;
    - all the productions of G_A;
    - all the productions of G_B.
  - It is proved in the next theorem that G_{AB} is ambiguous if and only if the instance (A, B) of PCP has a solution.

- **Theorem 9.20 ---**

  Whether a CFG is ambiguous is undecidable.

Proof.

- We only have to show that G_{AB} is ambiguous if and only if the instance (A, B) of PCP has a solution.

Proof of the "if" part ---

- Suppose $i_1, i_2, \ldots, i_k$ is a solution. Consider the following two derivations:

$$\begin{aligned} S &\Rightarrow A \Rightarrow w_{i_1}Aa_{i_1} \Rightarrow w_{i_1}w_{i_2}Aa_{i_2}a_{i_1} \\ &\Rightarrow \ldots \Rightarrow w_{i_1}w_{i_2}\ldots w_{i_{m-1}}Aa_{i_{m-1}}\ldots a_{i_2}a_{i_1} \\ &\Rightarrow w_{i_1}w_{i_2}\ldots w_{i_{m-1}}w_{i_m}a_{i_m}a_{i_{m-1}}\ldots a_{i_2}a_{i_1}; \end{aligned}$$

$$\begin{aligned} S &\Rightarrow B \Rightarrow x_{i_1}Ba_{i_1} \Rightarrow x_{i_1}x_{i_2}Ba_{i_2}a_{i_1} \Rightarrow \ldots \\ &\Rightarrow x_{i_1}x_{i_2}\ldots x_{i_{m-1}}Ba_{i_{m-1}}\ldots a_{i_2}a_{i_1} \\ &\Rightarrow x_{i_1}x_{i_2}\ldots x_{i_{m-1}}x_{i_m}a_{i_m}a_{i_{m-1}}\ldots a_{i_2}a_{i_1}. \end{aligned}$$

- Since $i_1, i_2, ..., i_m$ is a solution, we know that $w_{i_1}w_{i_2}\ldots w_{i_m} = x_{i_1}x_{i_2}\ldots x_{i_m}$.
- Thus, the two *distinct* derivations are derivations of the same string.
- That means that G_{AB} is ambiguous.

Proof of the "only if" part ---

- Suppose that G_{AB} is ambiguous.
- It is easy to see that a given string cannot have two derivations in G_A, nor in G_B.
- Therefore, the only way that a string could have two leftmost derivations in G_{AB} is if one of the two derivations begins $S \Rightarrow A$ and continues with a derivation in G_A and the other begins with $S \Rightarrow B$ and continues with a derivation of the same string in G_B.
- The string with two derivations has a tail of indexes $a_{i_m}\ldots a_{i_2}a_{i_1}$ for some $m \geq 1$.
- This tail must be a solution to the PCP instance, because what precedes the tail in the string with two derivations is both $w_{i_1}w_{i_2}\ldots w_{i_m}$ and $x_{i_1}x_{i_2}\ldots x_{i_m}$ which are equal. **Done**.

9.5.3 The Complement of a List Language

- **Theorem 9.21 ---**

  If L_A is the language for list A, then $\overline{L}_A$ is CFL.

  - For the proof, see the textbook.
  - Note that L_A is also a CFL because its grammar is a CFG, as mentioned previously.

- Usefulness of the list languages --- We can use L_A, L_B, and their complements in various ways to show the undecidability about CFL's.

- **Theorem 9.22 ---**

  Let G_1 and G_2 be CFG's, and let R be a regular expression. The following are undecidable:

  - Is $L(G_1) \cap L(G_2) = \phi$?
  - Is $L(G_1) = L(G_2)$?
  - Is $L(G_1) = L(R)$?
  - Is $L(G_1) = T^*$ for some alphabet T?
  - Is $L(G_1) \subseteq L(G_2)$?
  - Is $L(R) \subseteq L(G_1)$?

Proof:

- We conduct the proofs by problem reduction from PCP to each case.
- That is, we show how to take an instance (A, B) of PCP and convert it to a question about CFG's and/or regular expressions that has answer "yes" if and only if the instance of PCP has a solution.
- Let the alphabet of the PCP instance be Σ and that of the index symbols be I.

Proof for "Is $L(G_1) \cap L(G_2) = \phi$?"

- Let $L(G_1) = L_A$ and $L(G_2) = L_B$.
- Then $L(G_1) \cap L(G_2) = L_A \cap L_B$ is the set of solutions to this instance of PCP.
- Why? See grammars for lists A and B *before and in the proof of the last theorem*.
- So we have reduced PCP to the problem "Is $L(G_1) \cap L(G_2) \neq \phi$?"
- So this problem is undecidable.
- And by Theorem 9.3 ("the recursive language is closed under complementation"), the complemented problem is also undecidable.
- That is , the problem "Is $L(G_1) \cap L(G_2) = \phi$?" is undecidable.

Proof for "Is $L(G_1) = L(G_2)$?"

- Since CFL's are closed under union, we may construct a CFG G_1 for $\overline{L}_A \cup \overline{L}_B$.
- Since $(\Sigma \cup I)^*$ is a regular language, we may construct a grammar G_2 for it.
- From the set theory, we have $\overline{L_A} \cup \overline{L_B} = \overline{L_A \cap L_B}$.
- This means that $L(G_1)$ does not contain those strings which are solutions to the instance of PCP.
- *On the other hand*, $L(G_2)$ contains all the strings.
- So $L(G_1) = L(G_2)$ if and only if the PCP instance has no solution.
- Or inversely, $L(G_1) \neq L(G_2)$ if and only if the PCP instance has a solution.
- That is, we have reduce the PCP to the complement of the current problem.
- So, the problem "is $L(G_1) \neq L(G_2)$?" is undecidable.
- So, the problem "is $L(G_1) = L(G_2)$?" is undecidable (by Theorem 9.3).

For proofs of other cases, see the textbook.