5.8.13. Example

A program to carry out proper subtraction is as follows :

$[C]$ To A if $X_2 \neq 0$

$Y = X_1$

To E

$[A]$ To B if $X_1 \neq 0$

To A

$[B]$ $X_1 = X_1 - 1$

$X_2 = X - 1$

To C

$[E]$ print Y and stop.

Here, $\psi_p\ (a_1, a_2) = a_1 - a_2$ if $a_1 \geq a_2$; ψ_p is undefined if $a_1 < a_2$, that is the terminal snap shot is never reached.

5.8.14. Definition

An n-ary function f is partially computable if there is a program P such that

$$f\ (a_1, a_2, ..., a_n) = \psi_p\ (a_1, ..., a_n)$$

where = means not only that the two functions have the same value when both are defined, but also that if one function is undefined, then so too is the other.

5.8.15. Definition. An n-ary function f is computable if f is partially computable and total.

The above concepts link up the concepts of recursive function theory and programmed machines.

5.9. SOLVED PROBLEMS

1. Find the language $L\ (G)$ over $\{a, b, c\}$ generated by the unrestricted grammar G with productions $S \rightarrow aSb$, $aS \rightarrow Aa$, $Aab \rightarrow c$

Solution. First we must apply the first production one or more items to obtain the word $w = a^nSb^n$ where $n > 0$. To eliminate S, we must apply the second production to obtain the word $w' = a^m Aabb^m$ where $m = n - 1 \geq 0$. Now we can only apply the third production to obtain the word $w^n = a^m cb^m$ where $m \geq 0$. Accordingly

$$L\ (G) = \{a^m cb^m\ ;\ m \text{ non-negative}\}$$

That is $L(G)$ consists of all words with the same non-negative number of a's and b's separated by a c.

2. Find the language $L(G)$ generated by the context sensitive grammar $G = (N, T, S, P)$ where $N = \{S, Z, A, B\}$, $T = \{a, b\}$ and

$$P = \{S \to aSA/aZA,\ Z \to bB/bZB,\ BA \to AB,\ AB \to Ab,\ bB \to bb,\ bA \to ba,\ aA \to aa\}.$$

Solution. When $n > 1$, $m > 1$

$$\begin{aligned}
S &\overset{*}{\Rightarrow} a^{n-1} S A^{n-1} && \text{(applying } S \to aSA \ (n-1) \text{ times)}\\
&\Rightarrow a^n Z A^n && \text{(applying } S \to aZA \text{ once)}\\
&\overset{*}{\Rightarrow} a^n b^{m-1} Z B^{m-1} A^n && \text{(applying } Z \to bZB (m-1) \text{ times)}\\
&\Rightarrow a^n b^m B^m A^n && \text{(applying } Z \to bB \text{ once)}\\
&\overset{*}{\Rightarrow} a^n b^m A^n B^m && \text{(applying } BA \to AB \ mn \text{ times)}\\
&\Rightarrow a^n b^m A^n\ b\ B^{m-1} && \text{(applying } AB \to Ab \text{ once)}\\
&\overset{*}{\Rightarrow} a^n b^m A^n b^m && \text{(applying } bB \to bb (m-1 \text{ times)}\\
&\Rightarrow a^n b^m\ a\ A^{n-1}\ b^m && \text{(applying } bA \to ba \text{ once)}\\
&\Rightarrow a^n b^m a^n b^m && \text{(applying } aA \to aa \ (n-1) \text{ times)}
\end{aligned}$$

when $n = 1$, $m > 1$,

$$\begin{aligned}
S &\Rightarrow aZA && \text{(applying } S \to aZA \text{ once)}\\
&\overset{*}{\Rightarrow} ab^{m-1} ZB^{m-1} A && \text{(applying } Z \to bZB \ (m-1) \text{ times)}\\
&\Rightarrow ab^m B^m A && \text{(applying } Z \to bB \text{ once)}\\
&\overset{*}{\Rightarrow} ab^m AB^m && \text{(applying } BA \to AB \ m \text{ times)}\\
&\Rightarrow ab^m Ab\ B^{m-1} && \text{(applying } AB \to Ab)\\
&\overset{*}{\Rightarrow} ab^m Ab^m && \text{(applying } bB \to bb (m-1) \text{ times)}\\
&\Rightarrow ab^m ab^m && \text{(applying } bA \to ba \text{ once)}
\end{aligned}$$

Similarly, when $n > 1$, $m = 1$, $S \overset{*}{\Rightarrow} a^n ba^n b$ and when $n = 1$, $m = 1$, $s \overset{*}{\Rightarrow} abab$.

3. Describe the language generated by the unrestricted grammar

$S \to LaR$, $L \to LD/\varepsilon$, $Da \to aaD$, $DR \to R$, $R \to \varepsilon$.

Solution. $S \Rightarrow LaR$ (applying $S \Rightarrow LaR$ once)

$\Rightarrow aR$ (applying $L \rightarrow \varepsilon$ once)

$\Rightarrow a$ (applying $R \rightarrow \varepsilon$ once)

That is $S \overset{*}{\Rightarrow} a$. Thus $a \in L(G)$.

$S \Rightarrow LaR$ (applying $S \rightarrow LaR$ once)

$\Rightarrow LDaR$ (applying $L \rightarrow LD$ once)

$\Rightarrow LaaDR$ (applying $Da \rightarrow aaD$ once)

$\Rightarrow LaaR$ (applying $DR \rightarrow R$ once)

$\Rightarrow Laa$ (applying $R \rightarrow \varepsilon$ once)

$\Rightarrow aa = a^2$ (applying $L \rightarrow \varepsilon$ once)

Thus $a^2 \in L(G)$.

$S \Rightarrow LaR$ (applying $S \rightarrow LaR$ once)

$\Rightarrow LDaR$ (applying $L \rightarrow LD$ once)

$\Rightarrow LDDaR$ (applying $L \rightarrow LD$ once)

$\overset{*}{\Rightarrow} LaaaaDDR$ (applying $Da \rightarrow aaD$ 3 times)

$\overset{*}{\Rightarrow} LaaaaR$ (applying $DR \rightarrow R$ 2 times)

$\Rightarrow aaaaR$ (applying $L \rightarrow \varepsilon$ once)

$\Rightarrow aaaa = a^4$ (applying $R \rightarrow \varepsilon$ once)

Thus $a^4 \in L(G)$.

$S \Rightarrow LaR$ (applying $S \rightarrow LaR$ once)

$\overset{*}{\Rightarrow} LD^naR$ (applying $L \rightarrow LDn$ times)

$\overset{*}{\Rightarrow} La^{2^n} D^nR$ (applying $DR \rightarrow aa$D times 2^{n-1} times)

$\overset{*}{\Rightarrow} La^{2^n} R$ (applying $DR \rightarrow R$ n times)

$\Rightarrow a^{2^n} R$ (applying $L \rightarrow \varepsilon$ once)

$\Rightarrow a^{2^n}$ (applying $R \rightarrow \varepsilon$ once)

Thus $L(G) = \{a^{2^n} / n \geq 0\}$.

4. Describe the language generated by the unrestricted grammar with the given productions

$S \to LaR$, $L \to LD/LT/\varepsilon$, $Da \to aaD$, $Ta \to aaaT$, $DR \to R$, $TR \to R$, $R \to \varepsilon$.

Solution. Since we have only one S-production we have to apply $S \to LaR$ first. Since there are three L-productions then we can apply either $L \to LD$ or $L \to LT$. If we apply $L \to LD$ then it follows from problems that the productions $\{S \to LaR, L \to LD, L \to \varepsilon, Da \to aaD, DR \to R, R \to \varepsilon\}$ will generate the language

$$L_1 = \{a^{2^n} / n \geq 0\}.$$

If we apply $L \to LT$ we see that

$S \Rightarrow LaR \Rightarrow a$

$\therefore \quad a \in L(G)$

$S \Rightarrow LaR$ (applying $S \to LaR$ once)

$\Rightarrow LTaR$ (applying $L \to LT$ once)

$\Rightarrow LaaaTR$ (applying $Ta \to aaaT$ once)

$\Rightarrow aaaTR$ (applying $L \to \varepsilon$ once)

$\Rightarrow aaaR$ (applying $TR \to R$ once)

$\Rightarrow aaa = a^3$ (applying $R \to \varepsilon$ once)

Thus $a^3 \in L(G)$

$S \Rightarrow LaR$ (applying $S \to LaR$ once)

$\overset{*}{\Rightarrow} LTTaR$ (applying $L \to LT$ twice)

$\overset{*}{\Rightarrow} L\,a^9\,TTR$ (applying $Ta \to aaaT$ thrice)

$\overset{*}{\Rightarrow} La^9\,R$ (applying $TR \to R$ twice)

$\Rightarrow a^9\,R$ (applying $L \to \varepsilon$ once)

$\Rightarrow a^9$ (applying $R \to \varepsilon$ once).

In general

$S \Rightarrow LaR$ (applying $S \to LaR$ once)

$\overset{*}{\Rightarrow} LT^n aR$ (applying $L \to LT$ n times)

$\overset{*}{\Rightarrow} L_{a^{3^n}} T^n R$ (applying $Ta \to aaaT$ several times)

$\overset{*}{\Rightarrow} L_{a^{3^n}} R$ (applying $TR \to R$ n times)

$\Rightarrow a^{3^n} R$ (applying $L \to \varepsilon$ once)

$\Rightarrow a^{3^n}$ (applying $R \to \varepsilon$ once).

Hence the productions

$\{S \to LaR,\ L \to LT/1,\ Ta \to aaaT,\ TR \to R,\ R \to \varepsilon\}$ generates

$$L_2 = \{a^{3^n} / n \geq 0\}$$

Thus $L(G) = L_1 \cup L_2$

$= \{a^{2^n} / n \geq 0) \cup \{a^{3^n} / n \geq 0)$

5. Describe the language generated by the unrestricted grammar with the productions $\{S \to ABCS/ABC,\ AB \to BA,\ AC \to CA,\ BC \to CB,\ BA \to AB,\ CA \to AC,\ CB \to BC,\ A \to a,\ B \to b,\ C \to c\}$

Solution. $S \Rightarrow ABC$ (applying $S \to ABC$)

$\overset{*}{\Rightarrow} abc$ (applying $A \to a,\ B \to b,\ C \to c$)

Thus $abc \in L(G)$.

$S \Rightarrow ABCS$ (applying $S \to ABCS$ once)

$\Rightarrow ABCABC$ (applying $S \to ABC$ once)

$\overset{*}{\Rightarrow} A^2B^2C^2$ (applying $CA \to AC,\ BA \to AB,\ CB \to BC$)

$\overset{*}{\Rightarrow} a^2b^2c^2$ (applying A $\to a,\ B \to b,\ C \to b$ twice).

In general $S \Rightarrow ABCS$ (applying $S \to ABCS$)

$\overset{*}{\Rightarrow} (ABC)^n$ (applying $S \to$ ABC $(n-1)$ times)

$\overset{*}{\Rightarrow} A^nB^nC^n$ (applying $AB \to BA,\ AC \to CA,\ BC \to CB,\ BA \to AB,\ CA \to AC,\ CB \to BC$ several times)

$\overset{*}{\Rightarrow} a^n b^n c^n$ (applying $A \to a$, $B \to b$, $C \to c$ n times)

Thus $L(G) = \{a^n b^n c^n / n \geq 1\}$.

6. Find the language generated by the unrestricted grammar with productions $\{S \to 0SAB, S \to \varepsilon, BA \to AB, 0A \to 01, 1A \to 11, 1B \to 12, 2B \to 22\}$

Solution. $S \Rightarrow \varepsilon \quad \therefore \quad \varepsilon \in L(G)$

$S \Rightarrow 0SAB$ (applying $S \to 0SAB$ once)

$\Rightarrow 0AB$ (applying $S \to \varepsilon$ once)

$\Rightarrow 01B$ (applying $0A \to 01$ once)

$\Rightarrow 012$ (applying $1B \to 12$ once)

$S \Rightarrow 0SAB$ (applying $S \to OSAB$ once)

$\Rightarrow 00SABAB$ (applying $S \to OSAB$ once)

$\Rightarrow 00ABAB$ (applying $S \to \varepsilon$ once)

$\Rightarrow 001BAB$ (applying $0A \to 01$ once)

$\Rightarrow 001ABB$ (applying $BA \to 13$ once)

$\Rightarrow 0011BB$ (applying $1A \to 11$ once)

$\Rightarrow 00112B$ (applying $1B \to 12$ once)

$\Rightarrow 001122$ (applying $2B \to 22$ once)

Thus $001122 \in L(G)$

In general $n > 2$

$S \overset{*}{\Rightarrow} 0^n S (AB)^n$ (applying $S \to OSAB$ n times)

$\overset{*}{\Rightarrow} 0^n (AB)^n$

$\Rightarrow 0^{n-1} 01 A^{n-1} B^n$ (applying $0A \to 01$ once)

$\overset{*}{\Rightarrow} 0^n 1^{n-1} AB^n$ (applying $BA \to AB$ nC_2 times)

$\Rightarrow 0^n 1^n B^n$ (applying $1A \to 11$ once)

$\Rightarrow 0^n 1^n 2B^{n-1}$ (applying $1B \to 12$ once)

$\overset{*}{\Rightarrow} 0^n 1^n 2^n$ (applying $2B \to 22$ $(n-1)$ times)

Thus $L(G) = \{0^n 1^n 2^n / n \geq 0\}$

7. Prove that the language

$L = \{a^n b^n c^n / n \geq 1\}$ is recursive but not context free.

Solution. We describe the operation of a Turing machine which tests a string for membership in L.

1. Test the string to determine whether it is of the form $a^k b^l c^m$ without changing the string on the tape.
2. If the string has the proper format, rewind the tape, if it does not halt in a rejecting state.
3. For each 'a' found on the tape as the head scans to the right, replace it with a blank; locate one 'b' and one 'c' and replace them with blanks. Repeat this process for each a on the tape. If not enough b's are c's are found.
4. When no more a's can be found, test the tape to see whether it consists of entirely of blanks. If it does, accept the string. If the tape is not empty, halt in a rejecting state.

Hence L is recursive. For the proof of L is not context free see 3.4.3 problem 1.]

8. Let $G = (\{S, A, B\}, \{0, 1\}\ P, S)$ where P consists of $S \to 0AB0$, $A \to 1\ 0AB\ 1$, $B \to A01$, $0A \to 100$, $1B1 \to 0101$. Test whether $100110100011010 \in L(G)$.

Solution. We have to start with an S-production. At every stage we apply a suitable production which is likely to derive w.

Now
$S \Rightarrow 0AB0$ (applying $S \to 0AB0$ once)
$\Rightarrow 100B0$ (applying $0A \to 100$ once)
$\Rightarrow 100A010$ (applying $B \to A01$ once)
$\Rightarrow 10010AB1010$ (applying $A \to 1\ 0AB\ 1$ once)
$\Rightarrow 1001100B1010$ (applying $0A \to 100$ once)
$\Rightarrow 1001100A011010$ (applying $B \to A01$ once)
$\Rightarrow 100110100011010$ (applying $0A \to 11$ once).

Thus $100110100011010 \in L(G)$.

9. Find the language generated by the unrestricted grammar given by the productions

$S \to 0SBA/01A$, $AB \to BA$, $1B \to 11$, $1A \to 10$, $0A \to 00$.

Solution. $S \Rightarrow 01A$

$\Rightarrow 010$

$\therefore \quad 010 \in L(G)$

$S \Rightarrow 0SBA$ (applying $S \to 0SBA$ once)

$\Rightarrow 001ABA$ (applying $S \to 01A$ once)

$\Rightarrow 001BAA$ (applying $AB \to BA$ once)

$\Rightarrow 0011AA$ (applying $1B \to 11$ once)

$\Rightarrow 00110A$ (applying $1A \to 10$ once)

$\Rightarrow 001100$ (applying $0A \to 00$ once)

Thus $001100 = 0^2 1^2 0^2 \in L(G)$.

$S \overset{*}{\Rightarrow} 00SBABA$ (applying $S \to 0SBA$ once)

$\Rightarrow 0001ABABA$ (applying $S \to 01A$ once)

$\Rightarrow 0001BAABA$ (applying $AB \to BA$ once)

$\Rightarrow 00011AABA$ (applying $1B \to 11$ once)

$\overset{*}{\Rightarrow} 00011BAAA$ (applying $AB \to BA$ once)

$\Rightarrow 000111AAA$ (applying $1B \to 11$ once)

$\Rightarrow 0001110AA$ (applying $1A \to 10$ once)

$\overset{*}{\Rightarrow} 000111000$ (applying $0A \to 00$ twice)

In general $S \overset{*}{\Rightarrow} 0^{n-1} S (BA)^{n-1}$ (applying $S \to 0SAB$ $(n-1)$ times)

$\Rightarrow 0^{n-1} 01 A (BA)^{n-1}$ (applying $S \to 01A$ one time)

$\overset{*}{\Rightarrow} 0^n 1^n A^n$ (applying $AB \to BA$ several times and then $1B \to 11$ one time)

$\Rightarrow 0^n 1^n 0A^{n-1}$ (applying $1A \to 10$ one times)

$\overset{*}{\Rightarrow} 0^n 1^n 0^n$ (applying $0A \to 00$ $(n-1)$ times)

Thus $0^n 1^n 0^n \in L(G)$.

Hence $\{0^n 1^n 0^n / n \geq 1\} \subseteq L(G)$.

To prove that $L(G) \subseteq \{0^n 1^n 0^n / n \geq 1\}$, we proceed as follows. If we apply the production $S \to 01A$ first and then $1A \to 10$ we get 010.

Otherwise we have to apply $S \to 0SBA$ once or several times to get $0^{n-1} S (BA)^{n-1}$. To eliminate S we have to apply $S \to 01A$. Thus we arrive at the sentential form $0^n 1A\,(BA)^{n-1}$. To eliminate the variable B we have to apply $AB \to BA$ and $1B \to 11$ several times. Thus we get $0^n 1^n A^n$. Then we have to apply $1A \to 10$ one time and then $0A \to 00$ several times to get $0^n 1^n 0^n$. Thus

$$L(G) \subseteq \{0^n 1^n 0^n / n \geq 1\}$$

Thus
$$L(G) = \{0^n\ 1^n\ 0^n\ / n \geq 1\}.$$

10. Prove that $\{a^{(n+1)^2} / n \geq 1\}$ is generated by the unrestricted grammar with productions $S \to a$, $S \to CD$, $C \to ACB$, $C \to AB$, $AB \to aBA$, $Aa \to aA$, $Ba \to aB$, $AD \to Da$, $BD \to Ea$, $BE \to Ea$, $E \to a$.

Solution. We prove

(a) $S \overset{*}{\Rightarrow} A^n B^n D$

(b) $A^n B^n D \overset{*}{\Rightarrow} a^{n^2} B^n A^n D$

(c) $B^n A^n D \Rightarrow a^{2n+1}$

(a) $S \Rightarrow CD$ (applying $S \to CD$ once)

$\overset{*}{\Rightarrow} A^{n-1} CB^{n-1} D$ (applying $C \to ACB$ $(n-1)$ times)

$\Rightarrow A^n B^n D$ (applying $C \to AB$ once)

(b) $A^n B^n D \Rightarrow A^{n-1} a\, BA\, B^{n-1} D$ (applying $AB \to aBA$ once)

$\Rightarrow A^{n-2} a\, ABAB^{n-1} D$ (applying $Aa \to aA$ once)

$\Rightarrow A^{n-2} a^2\, BA\, AB^{n-1} D$ (applying $AB \to aBA$ once)

$\Rightarrow A^{n-2} a^2\, BA\, a\, B\, AB^{n-2} D$ (applying $AB \to aBA$ once)

$\Rightarrow A^{n-2} a^2\, Ba\, ABAB^{n-2} D$ (applying $Aa \to aA$ once)

$\Rightarrow A^{n-2} a^3\, BA\, BAB^{n-2} D$ (applying $AB \to aB$ once)

$\Rightarrow A^{n-2} a^3\, Ba\, BAAB^{n-2} D$ (applying $AB \to aBA$ once)

$\overset{*}{\Rightarrow} a^4 A^{n-2} B^2 A^2 D^{n-2}$ (applying $Ba \to aB$ once and $Aa \to aA$ several times)

proceeding in a similar way we get

$$A^n B^n D \overset{*}{\Rightarrow} a^{n^2} B^n A^n D$$

Hence (b).

Finally $B^n A^n D \Rightarrow B^n A^{n-1} Da$ (applying $AD \to Da$ once)

$\overset{*}{\Rightarrow} B^n\ Da^n$

$\Rightarrow B^{n-1}\ E\ a^{n+1}$ (applying $AD \to Da$ $(n-1)$ times)

(applying $BD \to Ea$ once)

$\overset{*}{\Rightarrow} Ea^{2n}$

$\Rightarrow a^{2n+1}$ (applying $BE \to Ea$ $(n-1)$ times)

(applying $E \to a$ once)

Hence (c)

Now $S \Rightarrow A^nB^n\ D$ (using (a))

$\overset{*}{\Rightarrow} a^{n^2} B^nA^nD$ (using (b))

$\overset{*}{\Rightarrow} a^{n^2}\ a^{2n+1}$ (using (c))

$= a^{(n+1)^2}$

Thus $L(G) = \left\{a^{(n+1)^2} / n \geq 1\right\}$.

11. Prove that every context sensitive language is recursive.

Proof. Let $G = (N, T, S, P)$ and $w \in T^*$. We have to construct an algorithm to test whether $w \in L(G)$ or not. If $w = \varepsilon$, them $w \in L(G)$ iff $S \to \varepsilon$ is in P. As they are only finite number of productions in P, we have to test whether $S \to \varepsilon$ is in P or not.

Let $|w| = n \geq 1$. The algorithm is based on the construction of a sequence $\{W_i\}$ of subsets of $(N \cup T)^*$. W_i is simply the set of all sentential forms of length less than or equal to n derivable in atmost i steps. The construction is done recursively as follows.

(*i*) $W_0 = \{S\}$

(*ii*) $W_{i+1} = W_i \cup \{\beta \in (N \cup T)^*$ /there exists α in W_i such that $\alpha \Rightarrow \beta$ and $|\beta| \leq n\}$. W_i's satisfy the following.

(*iii*) $W_i \subseteq W_{i+1}$ for all $i \geq 0$

(*iv*) There exists k such that $W_k = W_{k+1}$.

(*v*) If k is the smallest integer such that

$W_k = W_{k+1}$, then

$W_k = \{\alpha \in (N \cup T)^*$ /$S \overset{*}{\Rightarrow} \alpha$ and $|\alpha| \leq n\}$.

Point (*iii*) follows form point (*ii*). To prove (*iv*), we consider the number N of strings over $N \cup T$ of length less than or equal to n. If $|N \cup T| = m$, then

$r = 1 + m + m^2 + \ldots + m^n$ since m^i is the number of strings of length i over $N \cup T$. That is $r = (m^{n+1} - 1) / (m - 1)$ and r is fixed as it depends only on n and m. As any string in W_i is of length atmost n, $| W_i | \leq r$. Therefore, $W_k = W_{k+1}$ for some $k \leq t$. This proves point (*iv*). Form (*ii*) it follows that $W_k = W_{k+1}$ implies

$$W_{k+1} = W_{k+2}$$

$$\{\alpha \in (N \cup T)^* / S \overset{*}{\Rightarrow} \alpha, | \alpha |, \leq n\} = W_1 \cup W_2 \cup \ldots \cup W_k \cup W_{k+1} \cdots$$

$$= W_1 \cup W_2 \cup \ldots \cup W_k$$

$$= W_k \text{ from } (iii).$$

This proves (*v*).

From (*v*) it follows that $w \in L(G)$ (*i.e.* $S \overset{*}{\Rightarrow} w$) if and only if $w \in W_k$. Also, $W_1, W_2, \ldots, W_k$ can be constructed in a finite number of steps. We give the required algorithm as follows.

Algorithm to test whether $w \in L(G)$.

1. Construct $W_1, W_2, \ldots$ using (*i*) and (*ii*). We terminate the construction when $W_{k+1} = W_k$ for the first time.
2. If $w \in W_k$ then $w \in L(G)$. Otherwise $w \notin L(G)$. (As $| W_k | \leq r$, testing whether w is in W_k requires atmost r steps)

12. Let Γ be the grammar $G = (N, T, S, P)$ where $N = \{S, A\}$, $T = \{0, 1, 2\}$ and $P = \{S \to 0SA2, S \to 012, 2A \to A2, 1A \to 11\}$

Test whether (*a*) $00112 \in L(G)$ and (*b*) $001122 \in L(G)$.

Solution.

(*a*) Construct the sets W_0, W_1, W_2 using the problem 11. Here $| w | = 5$.

$$W_0 = \{S\}$$

$$W_1 = \{012, S, 0SA2\}$$

$$W_2 = \{012, S, 0SA2\}$$

As $W_2 = W_1$, we terminate. (Although $0SA2 \Rightarrow 0012A2$, we cannot include $0012A2$ in W_1 as its length is > 5). $00112 \notin W_1$. Hence $00112 \notin L(G)$.

(*b*) To test whether $w = 001122 \in L(G)$. Here $| w | = 6$. We construct W_0, W_1, W_2 etc.

$$W_0 = \{S\}$$

$$W_1 = \{012, S, 0SA2\}$$

$W_2 = \{012, S, 0SA2, 0012A2\}$

$W_3 = \{012, S, 0SA2, 0012A2, 001A22\}$

$W_4 = \{012, S, 0SA2, 0012A2, 001A22, 001122\}$

$W_5 = \{012, S, 0SA2, 0012A2, 001A22, 001122\}$

As $W_5 = W_4$, we terminate. Then $001122 \in W_4$. Thus $001122 \in L(G)$.

13. Prove that there exists a recursive set which is not a context sensitive language over $\{0, 1\}$.

Solution. Let $T = \{0, 1\}$. We write the elements of T^* as a sequence (*i.e.* the elements of T^* are enumerated as the first element, second element etc.). For example one way of writing is ε, 0, 1, 00, 10, 11, 000, 001, 010, ... In this case 010 will be the 10th element.

As every grammar is defined in terms of finite alphabet set and a finite set of productions, we can also write all context sensitive grammars over T as a sequence, say $G_1, G_2, \ldots$.

We define $X = \{w_i \in T^* / w_i \notin L(G_i)\}$. We can show that X is recursive. If $w \in T^*$ then we can find i such that $w = w_i$. This can be done in a finite number of steps (depending on $|w|$). For example, if $w = 01000$, then $w = w_{20}$. As G_{20} is context sensitive, we have an algorithm to test whether $w = w_{20} \in L(G_{20})$ by problem 11. Hence X is recursive.

We prove by contradiction that X is not a context sensitive language. If it is so, then $X = L(G_n)$ for some n. Consider w_n (the nth element in T^*). By definition of X, $w_n \in X$ implies $w_n \notin L(G_n)$. This contradicts $X = L(G_n)$. $w_n \notin X$ implies $w_n \in L(G_n)$ and once again, this contradicts $X = L(G_n)$. Thus $X \neq L(G_n)$ for any n *i.e.* X is not a context sensitive language.

14. Prove that the set $N \times N$ is countable where N is the set of all natural numbers.

Solution.

$$\begin{aligned} N \times N &= \{(x, y)/x, y \geq 0\} \\ &= \{(0, 0), (0, 1), (0, 2), \ldots\} \\ &\cup \{(1, 0), (1, 1), (1, 2), \ldots\} \\ &\cup \{(2, 0), (2, 1), (2, 2), \ldots\} \\ &\cup \ldots \\ &= \bigcup_{t=0}^{\infty} \{(t, y)/y \geq 0\} \\ &= \bigcup_{t=0}^{\infty} (\{t\} \times N) \end{aligned}$$

Since the function $f_t : N \to \{t\} \times N$ defined by $f_t\ (s) = (t, s)$ is a bijection each of the sets $\{t\} \times N$ is countable. (Note the t $\{t\} \times N$ is the set of elements in the tth row of Figure 5.4.). By 5.5.6. it follows that $N \times N$ is countable.

15. Prove that T^* = {set of all strings over a finite set T} is countable.

Solution.
$$T^* = \bigcup_{r=0}^{\infty} T^r$$

where T^r is the set of strings over T of length r.

Since T^r is finite and therefore countable, it follows from 5.5.6 that T^* is countable. [For example when $T = \{0, 1\}$, one way of listing the elements of T^* is to use the canonical order :

$$\{0, 1\}^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, ...\}].$$

16. Prove that any language over a finite set T is countable.

Solution. We know that a language L over T is a subset of T^*. Since every subset of a countable set is countable by problem 15 it follows that L is countable.

17. Prove that the set of all recursively enumerable languages is countable.

Solution. Let T be the set of all Turing machines and R be the set of all recursively enumberable languages. Here we are following our convention that all the states of a Turing machine are elements of the fixed set Q and all the tape symbols are elements of the set S. In particular, the recursively enumerable languages all involve alphabets that are subsets of S. Here we use the encoding function $e : T \to \{0, 1\}^*$.

The only property of e that we need here is that it is one-to-one and there a bijection from T to some subset of $\{0, 1\}^*$. Since $\{0, 1\}^*$ is countable and $T \subseteq \{0, 1\}^*$ we have T is countable.

A recursively enumerable language L can be accepted by some Turing machine. For each L, let t (L) be such a TM. The result is a function t from R to T, and since a Turing machine accepts precisely one language, t is one-to-one. Since T is countable, the same argument we used above shows that R is also countable.

18. Prove that $[0, 1) = \{x \in R\ /0 \leq x < 1$ and R is the set of all real numbers $\}$ is uncountable.

Solution. We use the fact that real numbers in [0, 1) have infinite (possibly repeating) decimal expansions, with no digits before the decimal point. For example, $\frac{1}{2} = .5000\ ...,\ \frac{1}{3} = .333...,\ \frac{\pi}{3} = .7853981...$, and so on. It is not quite correct to say that every number in [0, 1) has exactly one such expansion, because a non-zero number whose expansion terminates (or ends in an infinite

string of 0's) also has an expansion ending with infinitely many 9's ; for example, $\frac{1}{2} = 0.4999 \ldots$. If we agree to disallow expansions that end with strings of 9's, then every number in [0, 1) has exactly one infinite decimal expansion, and every infinite decimal expansion represents a number in this interval.

Suppose that [0, 1) is countable, so that [0, 1) = $\{a_0, a_1, a_2, ..\}$.

For each $i \geq 0$, suppose the decimal expansion of a_i is

$$a_i = a_{i;0}, a_{i;1}, a_{i;2} \ldots$$

The contradiction is obtained by constructing a number $x \in [0, 1)$ that is not in the list. An easy way to guarantee that x is different from a_i for every i is to make the decimal expansion of x differ from that a_i the ith position. Therefore, we define

$$x_i = \begin{cases} a_{i,i+1} \text{ if } a_{i,i} < 8 \\ 7 \text{ if } a_{i,i} \geq 8 \end{cases}$$

and let x be the real numbers with decimal expansion. $x_0\ x_1\ x_2 \ldots$. The two cases in the definition are to make sure that x_i's are single digits and not finally all 9's.

The diagonal aspect of the proof is that we have defined x_i in terms of the ith diagonal term of the infinite two-dimensional matrix

$$\begin{matrix} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \cdots \\ a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \cdots \\ a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \cdots \\ a_{3,0} & a_{3,0} & a_{3,0} & a_{3,3} \cdots \\ \cdots \end{matrix}$$

(the diagonal terms are underlined). It is clear that for every $i \geq 0$, $x_i \neq a_{i,i}$ and therefore $x \neq a_i$. This is a contradiction, because $x \in [0, 1)$. Thus [0, 1) is uncountable.

19. Find an unrestricted grammar to generate the following language.

$$L = \{a^n\ b^n\ a^n\ b^n / n \geq 0\}.$$

Solution. Consider a unrestricted grammar

$$G = (N, T, S, P)$$

where $N = \{S, A, B, C, D\}$

$T = \{a, b\}$

and $P = \{S \to ABCD, AB \to aABb, A \to a, B \to b, CD \to aCDb, C \to a, D \to b, S \to \varepsilon\}$

when $n = 0$

$$S \Rightarrow \varepsilon \quad \therefore \quad \varepsilon \in L(G)$$

when $n = 1$

$$\begin{aligned} S &\Rightarrow ABCD \\ &\Rightarrow aBCD && \text{(applying } A \to a \text{ once)} \\ &\Rightarrow abCD && \text{(applying } B \to b \text{ once)} \\ &\Rightarrow abaDC && \text{(applying } C \to a \text{ once)} \\ &\Rightarrow abab && \text{(applying } D \to a \text{ once)} \end{aligned}$$

Thus $abab \in L(G)$

When $n > 1$

$$\begin{aligned} S &\Rightarrow ABCD && \text{(applying } S \to ABCD \text{ once)} \\ &\Rightarrow aABbCD && \text{(applying } AB \to aABb \text{ once)} \\ &\overset{*}{\Rightarrow} a^{n-1} AB\, b^{n-1} CD && \text{(applying } AB \to aABb \ (n-2) \text{ times)} \\ &\Rightarrow a^n B b^{n-1} CD && \text{(applying } A \to a \text{ once)} \\ &\Rightarrow a^n b^n CD && \text{(applying } B \to b \text{ once)} \\ &\overset{*}{\Rightarrow} a^n b^n a^{n-1} CD\, b^{n-1} && \text{(applying } CD \to a\, CDb \ (n-1) \text{ times)} \\ &\Rightarrow a^n b^n a^n Db^{n-1} && \text{(applying } C \to a \text{ once)} \\ &\Rightarrow a^n b^n a^n b^n && \text{(applying } D \to b \text{ once)} \end{aligned}$$

Thus $a^n b^n a^n b^n \in L(G)$

Hence G generates L.

20. For any languages $L_1 \subseteq \Sigma_1^*$, $L_2 \subseteq \Sigma_2^*$ with $L_1 \leq L_2$, if L_2 is recursive, then L_2 is recursive (or equivalently, if L_1 is not recursive, then L_2 is not). For any decision problems P_1 and P_2 with $P_1 \leq P_2$, if P_2 is solvable then P_1 is solvable. (or, if P_1 is unsolvable, P_2 is unsolvable).

Solution. Suppose M_2 is a TM which decides the language L_2 and M_f be a TM that computes the function $f : \Sigma_1^* \to \Sigma_2^*$ involved in the reduction. Let M_1 be the composite $M_1 = M_f M_2$. On input $x \in \Sigma_1^*$, M_1 first computes $f(x)$, then halts with output 1 or 0, depending on whether $f(x)$ is in L_2 or not. The assumption that f is a reduction of L_1 or L_2 means that the output is 1 if $x \in L_1$ and 0 otherwise, and it follows that M_1 recognizes L_1.

The second part of the problem follows from the discussion after Definition 5.7.17.

21. Whether the decision problem, named *Accepts*.

Accepts : Given a TM M and a string w, is $w \in L(M)$? is undecidable.

Solution. An instance of *Accepts* consists of a Turing machine M and a string w. Using encoding function e represent it as the string $e(M)\ e(w)$.

Let $$L_U = \{e(M)\ e(w) : w \in L(M)\}.$$

To show *Accepts* is undecidable, it is enough to show L_U is not recursive. By Theorem 5.6.7. the result follows.

22. Whether the halting problem abbreviated as *Halts*

Halts: Given a TM M and a string w, does M halts on input w?

is undecidable.

Solution. Consider the corresponding language

$$H = \{e(M)\ e(w) : M \text{ halts on } w\}$$

is undecidable.

It is enough to show that 'Accepts' can be reduced to 'Halts', and from problems 20 and 21 it follows that 'Halts' is unsolvable.

For both problems, an instance is a pair consisting of a TM and a string. For an instance (M, w) of 'Accepts', construct an instance (M', w') of 'Halts', such that the answers are same in both cases. That is, M' halts on w' if and only if M accepts w. In other words, if M accepts w, M' must halts on w', and if M either rejects w or loops forever on w, M' must not halt on w'.

Let $w' = w$. Then M' can be obtained from M by changing move of the form $\delta(p, a) = (h_r, b, D)$ to $\delta(p, a) = (p, a, H)$, where h_r denotes the rejecting state. Other modification in machine M' is as follows. It begins to insert a new symbol say # in cell 0, moving everything else over one cell, and then move to state q_0 with the tape head in cell 1. Additional moves for M' is $\delta(q, \#) = (q, \#, H)$ for all possible states q.

Besides these moves, M' also has all the moves as in M. The effect is that if M ever tries to moves its tape head off the tape, then M' enter an infinite loop with its tape head on cell 0.

This algorithm for obtaining M' from M gives the reduction, hence '*Halts*' is undecidable.

23. The language $E = \{ e(M) : M \text{ is a TM}\}$ is recursive.

Solution. A string x of 0's and 1's is in E if and only if it satisfies the following conditions:

(1) x corresponding to the regular expression $0000^* 1 ((0^+ 1)^5)^+$ so that the

substring following the first 1 can be viewed as a sequence of 5-tuples.

(2) For two distinct 5-tuples y and z in the string x, the first two of the five parts of y cannot be the same as the first two of the five parts of z (A deterministic TM cannot have two distinct moves for the same-symbol combination).

(3) None of the 5-tuples in x can have first part 0 or 00 (A TM cannot move from halt state)

(4) The last part of each 5-tuple must be 0, 00, or 000 (representing one of the three directions)

Any string satisfying these conditions represent a TM, whether or not it carries out a meaningful computation. There is an algorithm to take an arbitrary element of $\{0, 1\}^*$ and determine the truth or falsity of each condition, and it is possible to implement such an algorithm on a TM. Hence the language E is recursive.

24. The decision problem, named 'Accepts-B'

'Accepts-B' : Given a TM M, is M finally reach the accepting state, if it begins with a blank tape B.

is undecidable.

Solution. In order to show 'Accepts-B' is undecidable, it is enough to show 'Accepts' is reduced to 'Accepts-B' (*i.e.*, Accepts $\leq$ Accepts-B)

For both problems instance is a pair consisting of a TM and a string. Suppose (M, w) is a Yes-instance of 'Accepts' (*i.e.* M accepts w) if and only M' is an yes-instance of Accepts-B (*i.e.* M' accepts B).

M' starts with B, but create a tape with x on it by simply writing x. The computation it perform at that point will be exactly same as the computation performed by M on input x. In other words., M' is the composite write (x) M, where write (x) is the TM that halts with tape content B_{q_a} x when started with a blank. Write (x) assumes the tape is initially blank, and thus M' is likely to crash on any input other than B. Hence M' accepts B if and only if M accepts x.

The construction of M' in terms of M and x can be carried out algorithmicaly, it follows that 'Accepts-B' is undecidable.

25. Whether the decision problem,

'Accepts Something' : Given a TM M, is $L(M) \neq \phi$?

is undecidable.

Solution. To prove the result it enough to show **Accepts-*B*** can be reduced to **Accepts something.** For both problems , an instance is a Turing Machine. Start with a TM M and it is easy to construct a TM M' such that

M accepts B if and only if M' accepts atleast one string.

The computation of M' on any input is the one performed by M on B. M' simply erases its input first and then executes M; it is the composite machine M' = *Erase tape* $\rightarrow M$, where *Erase tape* erases its input string and leaves the tape head on cell 0. If M accepts B, then M' will eventually accepts, no matter what's its input is, and otherwise M, will never accepts. Since the function that takes M to M' is computable, the reduction follows.

26. Whether the decision problem.

Accepts everything : Given a TM M, with input alphabet Σ, is $L(M) = \Sigma^*$?

is undecidable.

Solution. The function constructed in the last problem is also a reduction of **Accepts-*B*** to **Accepts Every thing** because the machine M' accepts either everything (if T accepts B) or nothing. Hence **Accepts Everything** is undecidable.

27. Whether the decision problem

subset : Given two TMs M_1 and M_2, is $L(M_1) \subseteq L(M_2)$?

is undecidable.

Solution. An instance of *subset* is a pair (M_1, M_2) of Turing machines, and it is a yes-instance if $L(M_1) \subseteq L(M_2)$. The solution follows by showing Accepts *Everything* can be reduced to **subset.**

For that start with a TM M, and construct a pair (M_1, M_2) so that M accepts every string over its input alphabet if and only if $L(M_1) \subseteq L(M_2)$. The way to proceed here is to ignore everything but the set $A = L(M)$, and find other two B and C so that $A = \Sigma^*$ if and only if $B \subseteq C$. An appropriate choice is to let $B = \Sigma^*$ and $C = A$ (A subset of Σ^* is Σ^* if and only if it contains Σ^*). Therefore M' be the trivial TM with input alphabet Σ that immediately accepts, no matter what the input, so that $L(M_1) = \Sigma^*$ and let $M_2 = M$. This way of construction of (M_1, M_2) form M gives the reduction of **subset** from **Accepts Everything.**

28. The decision problem

Writes symbol Given a TM M, and a symbol 'a' in its tape alphabet, does M ever write 'a' if it is started with a blank tape ?

is undecidable.

Solution. To prove the undecidability of **write symbol,** it is enough to show *Accepts-B* is reduced to *writes Symbol.*

Starting with a TM M; an instance of *Accetps-B,* construct a pair (M', a) in such a way that M accetps B if and if M' eventually writes 'a' when started with a blank. The computation of M' is same as that of M, except that if M halts, M' writes a, and M' never writes a in any other situation. Precisely speaking, M' be a TM with the same input and tape alphabets as M except for one new tape symbol 'a' and the same transitions as M except the move.

$$\delta (p, \sigma) = (q_a, \tau, D)$$

of T becomes

$$\delta (p, \sigma) = (q_a, a, D)$$

Instead. Starting with input B, the two machines perform exactly the same computation until they accepts, and M' writes an 'a' if and only if this happens. Hence the reduction.

29. For any specific recursively enumerable language L_2, such that $L_2 = \{B\}$, the problem

Accepts (L_2): Given a TM M, is $L(M) = L_2$? is undecidable.

Solution. Because L_2 is a non-trivial property, by Rice's theorem the result follows.

30. Does the problem

Equivalent : Given TMs M_1 and M_2, is $L(M_1) = L(M_2)$?

undecidable?

Solution. The problem **Accepts (L_2)** is reducible to **Equivalent**, because if M_2 is a TM accepting L_2, then for any instance M of **Accepts (L_2)**, the pair (M, M_2) is an instance of **Equivalent** having the same answer. Hence **Equivalent** is undecidable.

31. The problem

'Given a TM M, does M make more than 50 moves on input B?

is decidable.

Solution. Given a TM M means given enough information to trace the processing of a fixed string for a certain fixed number of moves . So the given problem is decidable.

32. The decision problem

Writes Non blank : Given a TM M, does it ever write a non-blank symbol in

its tape, when stated with a blank?

is decidable.

Solution. Given a TM M, construct a TM M' as follows. M' has the same tape alphabet as M except that it has one additional symbol #. The states of M_1 are the same as those of M. The transitions of M_1 are the same, except that for any transition of M in which a non-blank symbol is written, the corresponding transition of M_1 writes # and halts. Therefore starting with a blank symbol, M writes a non-blank symbol if and only if M_1 writes the symbol #. Hence **writes Non-blank** is solvable.

33. The decision problem

CFG Generates All: Given a context -free grammar (CFG) G with terminal alphabet Σ, is $L(G) = \Sigma^*$.

is undecidable.

Solution. Accepts Nothing (Given M is $L(M) = \phi$) is reducible to **CFG Generates All.** Starting with a TM M, there is an algorithm to construct a CFG G with terminal alphabet Σ_1, so that $L(G)$ is the complement C'_M of the set of valid computations of M. Saying that M accepts no string is equivalent to saying there are no valid computations of M or that C'_M contains every string over Σ_1. Therefore, M is a yes-instance of **Accepts Nothing** if an only if G is a yes-instance of **CFG Generates All** .

34. The problem

CFG Non-empty Intersection: Given two CFGs G_1 and G_2, is $L(G_1) \cap L(G_2)$ non empty?

is undecidable.

Solution. Accepts Something is reducible to **CFG Non-empty Intersection** For any TM M, the set C_M of valid computations of M can be expressed as $L(M_1) \cap L(M_2)$ for two CFGs G_1 and G_2 (Proof of this statement is left as an exercise). It is easy to construct an algorithm for the grammars G_1 and G_2 from M. Since M is an yes-instance of **Accepts something** if and only if the pair (G_1, G_2) is a yes-instance of **CFG Non-empty Intersection,** the first problem is reducible to the second and the second is therefore unsolvable.

35. Show that the language

$$L_r = \{e(M) : L(M) \text{ is recursive}\}$$

is not recursively enumerable.

Solution. Suppose L_r is recursively enumerable language. Let M_r be a TM accepting L_r. Now construct an algorithm A that takes $e(M)$ $e(w)$ as input and

produces as output a TM M'

such that

$$L(M') = \begin{cases} \phi \text{ if M does not accept } \omega \\ L_U \text{ if M accepts } \omega \end{cases}$$

Note that L_U is not recursive, so M' accepts a recursive language if and only if M does not accepts w. The construction of M' is shown in Fig. 5.27.

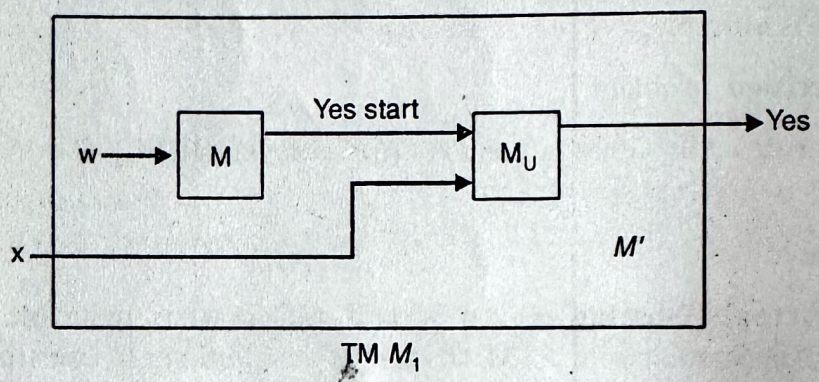

Fig. 5.27.

Given A and M_r, construct a TM accepting $\bar{L}_U$ as shown in Fig. 5.28.

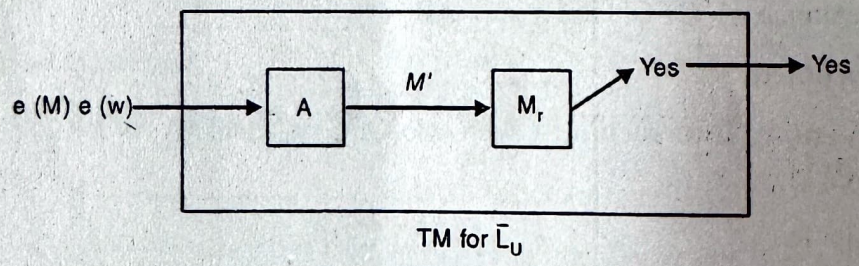

Fig. 5.28.

For an input $e(M)\ e(w)$, the TM uses A to produce M', uses M_r to determine if the set accepted by M' is recursive, and accetps if and only if $L(M')$ is recursive. But $L(M')$ is recursive if and only if $L(M') = \phi$ which means M does not accept w. Thus the TM of Fig. 5.23. accepts $e(M)\ e(w)$ if and only if $e(M)\ e(w)$ is $\bar{L}_U$.

36. Show that the language

$$L_{nr} = \{e(M) : L(M) \text{ is not recursive}\}$$

is not recursively enumerable.

Solution. Suppose there is a TM M_{nr} accepting L_{nr}. Construct an algorithm B that takes $e(M)\ e(w)$ as input and produces a TM M' such that

$$L(M') = \begin{cases} \Sigma^* \text{ if M accepts } \omega \\ L_U \text{ if M does not accept } \omega \end{cases}$$

The construction of M' is shown in Fig. 5.24.

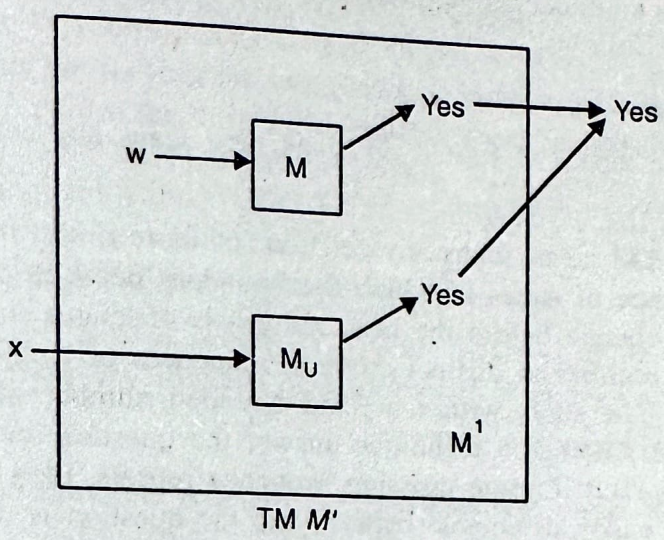

Fig. 5.29.

M' accetps a recursive language if and only if M accepts w. Turing machine to accepts $\overline{L}_U$ is shown in Fig. 5.30.

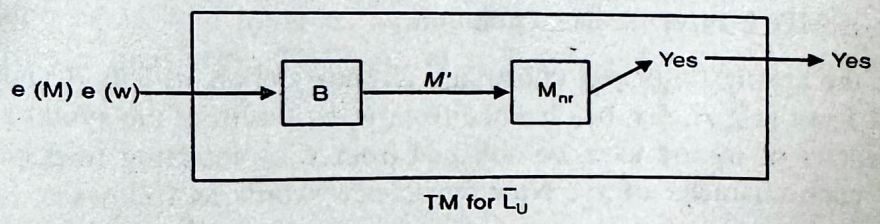

Fig. 5.30.

The TM of Fig. 5.30. accepts $e(M)\ e(w)$ if and only if $L(M')$ is not recursive, or equivalently, if and only if M does not accepts w. That is, the TM Accepts $e(M)\ e(w)$ if and only if $e(M)\ e(w)$ is in $\overline{L}_U$. As no such TM exists, there does not exist any M_{n_r} to accepts L_{n_r}. Hence L_{n_r} is not recursively enumerable.

37. Prove that "whether an arbitrary TM every prints three consecutive 1's " is undecidable.

Solution. For each TM M_i construct a TM $\hat{M}_i$ which on blank tape simulates M_i on blank tape. However $\hat{M}_i$ uses 01 to encode a 0 and 10 to encode a 1. If

M_i's tape has a 0 in cell j, $\hat{M}_i$ has 01 in cells $2j - 1$ and $2j$. If M_i changes a symbol, $\hat{M}_i$ changes the corresponding 1 to 0, then the paired 0 to 1. Easily one can design $\hat{M}_i$ that never, has three consecutive 1's on its tape. $\hat{M}_i$ prints three consecutive 1's and halts iff M_i accepts ε. By Rice theorem, 'whether a TM accepts ε' is an undecidable problem as 'ε is in L' is a non-trivial property. Hence the result follows.

38. The problem

Whether a single tape TM started on blank tape scans any cell four or more times is decidable.

Solution. If the TM never scans any cell four or more times, than every cross sequence (sequence of states in which the boundary between cells is crossed, assuming states change before the head moves) is of length atmost three. But there is a finite number of distinct crossing sequences of length three or less. Thus either the TM stays within a fixed bounded number of tape cells, in which case finite automaton technique answer the question, or some crossing sequence repeats. But if some crossing sequence repeats, then the TM moves right with some easily detectable pattern, and the question is decidable.

39. If PCP is decidable, then MPCP is decidable. That is MPCP reduces to PCP.

Solution. Let A = w_1 , w_2, ..., w_k and $B = x_1, x_2, ..., x_k$ be an instance of MPCP. Convert this instance of MPCP into an instance of PCP that has a solution if and only if MPCP instance has a solution.

Let Σ be the smallest alphabet containing all the symbols in lists A and B, and let ¢ and \$ not in Σ. Let y_i be obtained from w_i by inserting the symbol ¢ after each character of w_i and let z_i be obtained from x_i by inserting trhe symbol ¢ ahead of each character of x_i . Now create new words as follows:

$$y_0 = ¢\, y_1,\ z_0 = z_1$$

$$y_{k+1} = \$,\ z_{k+1} = ¢\,\$$$

Let $C = y_0, y_1, .., y_{k+1}$ and $D = z_0, z_1, ..., z_{k+1}$. For example, the list of C and D constructed form the lists of A and B of Example 5.7.13 are shown in table.

MPCP

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

	List C	*List D*
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	\$	¢ \$

PCP

In general the lists C and D represent an instance of PCP. This instance of PCP has a solution if and only if the instance of MPCP represented by lists A and B has a solution.

Note that if $1, i_1, i_2, \ldots i_r$ is a solution to MPCP with lists A and B, then $0, i_1, i_2, \ldots, i_r, k+1$ is a solution to PCP with lists C and D. Likewise, if $i_1, i_2, \ldots, i_r$ is a solution to PCP with lists C and D, then $i_1 = 0$ and $i_r = k + 1$, since y_0 and z_0 are the only words with the same index that begin with the same symbol, and y_{k+1} and z_{k+1} are the only words with the same index that end with the same symbol. Let j be the smallest integer such that $i_j = k + 1$. Then $i_1, i_2, \ldots, i_j$ is also a solution, since the symbol \$ occurs only as the lost symbol of y_{k+1} and z_{k+1}, and for no l, where $1 \leq l < j$,is $i_l = k + 1$. Clearly $1, i_2, i_3, \ldots, i_{j-1}$ is as solution to MPCP for lists A and B.

If there is an algorithm to decide PCP, there exists an algorithm to decide MPCP by converting any isntance of MPCP to PCP as above.

40. PCP is undecidable.

Solution. By the previous problem it is sufficient to show that if MPCP is decidable, then it is decidable that "whether a TM accepts a given w". That is, for each M and w, construct an instance of MPCP that has a solution if only if M accepts w.

An isntance of MPCP, if it has a solution, starts with $\# q_0 w \# \alpha_1 q_1 \beta_1 \# \alpha_2 q_2 \beta_2 \ldots \# \alpha_k q_k \beta_k$ where strings between successive #'s are successive ID's in a computation of M with inptu w_1 and q_k is a final state. Assume there are no moves from a final state. The pairs of strings forming lists A and B of the instance of MPCP are as follows.

First pair is:

	List A	*List B*	
	#	$\# q_0 w \#$	
Group I.	*List A*	*List B*	
	X	X	for each $X \in \Gamma$
	#	#	

Group II. For each q in $Q - F$, p in Q, and X, Y and Z in Γ:

List A	*List B*	
qX	Yp	if $\delta(q, X) = (p, Y, R)$
ZqX	pZY	if $\delta(q, X) = (p, Y, L)$
$q\#$	$Yp\#$	if $\delta(q, B) = (p, Y, R)$
$Zq\#$	$pZY\#$	if $\delta(q, B, = (p, Y, L)$

Group III. For each q in F, X, Y in Γ

List A	*List B*
XqY	q
Xq	q
qY	q

Group IV

List A	*List B*	
q ##	#	for each q in F.

(x, y) is said to be a *partial solution* of MPCP with lists A and B if x is a prefix of y, and x and y are the concatenation of corresponding strings of lists A and B respectively.

If $xz = y$, then z is said to be the *remainder* of (x, y).

Claim: There is a parital solution (unique)

$(x, y) = (\# q_0 w \# \alpha_1 q_1 \beta_1 \# \ldots, \# \alpha_{k-1} q_{k-1} \beta_{k-1} \# , \# q_0 w \# \alpha_1 q_1 \beta_1 \# \ldots \# \alpha_k q_k w_k \#)$

whose longer string is as long as $| y |$.

The above can be proved by induction on k.

If $k = 0$, the result is trivial, since the pair $(\#, \# q_0 w \#)$ must be chosen first. Suppose the statement is true for some k and $q_k \notin F$. Then it is easy to show the statement for $k + 1$ is also true. The remainder of the pair (x, y) is $z = \alpha_k q_k \beta_k \#$. The next pairs must be chosen so that their strings from list A form z. Whatever symbols appear to the right and left of q_k, there is atmost one pair in Group II that enable the parital solution to be continued past q_k. This pair represents, the move of M from ID $\alpha_k q_k \beta_k$. The other symbols of z force choices from group I. No other choices will enable z to be composed of elements in list A.

Thus a new parital solution $(y, y \alpha_{k+1} q_{k+1} \beta_{k+1} \#)$ is obtained. It is straight forward to see that $\alpha_{k+1} q_{k+1} \beta_{k+1}$ is the one ID that M can reach on one move from $\alpha_k q_k \beta_k$. Also there is no other parital solution whose length of the second string equals $| y \alpha_{k+1} q_{k+1} \beta_{k+1} \#|$

If $q_k \in F$, then it is easy to find pairs from groups I and III which, when preceded by the partial solution (x, y) and followed by the pair in Group IV, provide a solution to MPCP with lists A and B.

Thus if M, started in ID q_0w, reaches an accepting state, the instance of MPCP with lists A and B has a solution. If M does not reach an accepting state, no paris of group III and IV are used. Therefore, there may be partial solutions, but the strings of B exceed the strings of A in length, so no solution is possible.

Hence the instance of MPCP has a solution if and only if M with input w halts

in an accepting state. Since M and w are arbitrary, if there is an algorithm to solve MPCP, then there will be an algorithm to recognize L_U, contradicting that L_U is recursive. Hence MPCP is undecidable, which further implies PCP is undecidable.

41. Let $M = (Q, \Sigma, \Gamma, B, \delta, q_1, F)$ be a TM with $Q = \{q_1, q_2, q_3\}$, $\Sigma = \{0, 1\}$, $\Gamma = \{0, 1, B\}$,

$F = \{q_3\}$ and δ be defined by:

States	*0*	*1*	*B*
q_1	$(q_2, 1, R)$	$(q_2, 0, L)$	$(q_2, 1, L)$
q_2,	$(q_3, 0, L)$	$(q_1, 0, R)$	$(q_2, 0, R)$

Let $w = 01$. Construct an instance of MPCP with lists A and B as follows.

First Pair:

List A	*List B*
#	# q_1 01 #

Group I.

List A	*List B*
0	0
1	1
#	#

Group II.

List A	*List B*	
$q_1 0$	$1q_2$	from $\delta(q_1, 0) = (q_2, 1, R)$
$0q_1$ 1	$q_2 00$	from $\delta(q_1, 1) = (q_2, 0, L)$
$1q_1 1$	$q_2 10$	
$0q_1$ #	$q_2 01$	from $\delta(q_1, B) = (q_2, 1, L)$
1 q_1#	$q_2 11$#	
$0q_2$ 0	$q_3 00$	from $\delta(q_2, 0) = (q_3, 0, L)$
$1q_2 0$	$q_3 10$	
$q_2 1$	$0q_1$	from $\delta(q_2, 1) = (q_1, 0, R)$
q_2#	$0q_2$ #	from $\delta(q_2, B) = (q_2, 0, R)$

Group III.

List A	*List B*
$0q_3 0$	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.

Group IV. *List A* *List B*

q_3 ## #

For the input $w = 01$, the computation is

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

Hence the ID's are

q_101, $1q_21$, $10q_1B$, $1q_2\ 01$, $q_3\ 101$.

For the MPCP constructed as above, the next task is to check whether there is a solution.

The first pair gives a parital soltuion (# , #, $q_1$01 #). That is

#
q_10_1

Fig. 5.31 (a).

Since the length of second string is longer, next choose the pair (q_10, $1q_2$) (see (Fig, 5.31 (*b*))

#	q_10
# q_1 01 #	$1q_2$

Fig. 5.31 (b).

The resulting partial solution is (# q_1 0, # $q_1$01, 01 # q_2)

(That is, Fig 5.31(*c*))

#q_10
q_1 01 # 1 q_2

Fig. 5.31 (c).

The remainder is 1 # $1q_2$. The next pair chosen must be (1, 1). In this case the partial solution is (see Fig. 5.31 (*e*)).

# q_1 0	1
# q_1 01 # $1q_2$ 1	1

Fig. 5.31 (d).

$q_1$01
$q_1$01 # $1q_2$ 1

Fig. 5.31 (e).

The next wo choices are (#, #) and (1, 1). The partial solution becomes (# $q_1$01 # 1, # q_1 01 # 1q_2 1 # 1). The remainder now is q_2 1 # 1 (see Fig. 5.31 (*f*)).

q_1 01 # 1
q_1 01 # 1 q_2 1 # 1

Fig. 5.31 (f).

Constinuing in this way, one can obtain a partial solution, where the length of second string is 14, is (x, x_0 q_1 # 1), where x = # q_1 01 # 1 q_2 1 # 1.

The next choice is (0, 0) or (0q_1 #, q_2 01 #)

x	0
x 0q_1 # 1	0

Fig. 5.31 (g).

or

x	q_1 #
x 0 q_1 # 1	q_2 01 #

Fig. 5.31 (h).

By Fig. 5.31. (*g*), the resulting parital solution is (x0, x0q_1 # 10). The remainder is now q_1 # 10. There is no string in list A starting with q_1 #, it is not possible to make another partial solution. Hence this cannot lead to a solution. By Fig. 5.31 (*h*), the resulting parital solution is (x0q_1#, x0q_1 #1$q_2$01 #). The remainder is now 1 q_2 01 #. The next possible pair is (1q_2 0, q_3 10)

x 0 q_1 #	1$q_2$0
x0q_1 # 1 q_2 01 #	q_3 10

Fig. 5.31. (i).

By Fig. 5.31. (*i*), the resulting partial solution is (x 0 q_1 # 1q_2, x 0 q_1 #1q_2 01 #). Continning in this way, the final parital solution obtained is of the form (y, y1 # q_3 10), where y = # $q_1$01 # 1$q_2$1 # 10 q_1 # 1$q_2$0. Since q_3 is a final stage, use pairs in Groups I, III, IV to find a solution to the instance of MPCP.

The choice of pairs is

(1, 1), (#, #) , ($q_3$1, q_3) , (0, 0) , (1, 1), (# , #) ($q_3$0, q_3), (1, 1), (#, #), (q_3 1, q_3), (#, #) (q_3 ##, #).

Thus the shortest word that can be composed of strings from lists A and *B* starting with pair 1 is

q_1 01 # q_2 1 # 10 q_1 # 1 q_2 01 # q_3 101 # q_3 01 # q_3 1 # q_3

42. The problem 'whether an orbitrary CFG is ambiguous is undecidable.

Solution. Let $A = w_1, w_2, ..., w_n$; $B = x_1, x_2, ..., x_n$ be two lists of strings over a finite alphabet Σ. Let $a_1, a_2, ..., a_n$ be new symbols. Let

$$L_A = \{w_{i_1} w_{i_2} \ldots w_{i_m} a_{i_m} a_{i_{m-1}} \ldots a_{i_1} : m \geq 1\}$$

$$L_B = \{x_{i_1} x_{i_2} \ldots x_{i_m} a_{i_m} a_{i_{m-1}} \ldots a_{i_1} : m \geq 1\}$$

Let G be the CFG such that $G = (\{S_1, S_2, S_3\}, \Sigma \cup \{a_1, a_2, \ldots, a_n\} P, S)$ where contains the productions $S \to S_A$, $S \to S_B$

and $S_A \to w_i S_A a_i$, $S_A \to w_i a_i$,

$S_B \to x_i S_A a_i$, $S_B \to x_i a_i$, for all i, $1 \leq i \leq n$.

If the instance (A, B) of PCP has a soluton, say $i_1, i_2, \ldots, i_m$, then

$$x_{i_1} x_{i_2} \ldots x_{i_m} a_{i_m} a_{i_{m-1}} \ldots a_{i_1} = w_{i_1} w_{i_2} \ldots w_{i_m} a_{i_m} a_{i_{m-1}} \ldots a_{i_1}$$

This string has a leftmost derivation beginning with $S \to S_A$ and another beginning with $S \to S_B$. Hence G is ambiguous.

Conversely, suppose G is ambiguous. since the a' dictate the productions used, it is easy to see that there is only one left most derviation from S_A as well as from S_B. Thus it must be that some word has left most derivations from both S_A and S_B. If this is a word $y\, a_{i_m} a_{i_{m-1}} \ldots a_{i_1}$, $y \in \Sigma^*$, then $i_1, i_2, \ldots i_m$ is a solution of PCP.

Thus G is ambiguous if and only if the instance (A, B) of PCP has a soluton. Thus if there it an algorithm for ambiguity problem for CFG's then there exists an algorithm for PCP. As PCP is undecidable, (no algorithm exists for PCP), the ambiguity problem for CFG's is undecidable.

43. For any arbitrary CFG G, whether

$$L(G) = \Sigma^*$$

is undecidable.

Solution. For a TM M, CFG G can be constructed with terminal alphabet Σ such that

$$L(G) = \Sigma^* \Leftrightarrow L(M) = \phi.$$

That is, by Lemma 5.7.19. There is a CFG G that generates the invalid computations of M. Thus if for arbitrary G, $L(G) = \Sigma^*$ were decidable, then $L(M) = \phi$ is decidable for arbitrary M. Hence a contradiction.

44. Let G_1 and G_2 be arbitrary CFG's and R be an arbitrary regular set. Then the following are undecidable problems.

(1) $L(G_1) = L(G_2)$

(2) $L(G_2) \subseteq L(G_1)$

(3) $L(G_1) = R$

(4) $R \subseteq L(G_1)$

Solution. Fix G_2 to be a grammar generating Σ^*, where Σ is the terminal alphabet of G_1. Then (1) and (2) are equivalent to $L(G_1) = \Sigma^*$.

Fix $R = \Sigma^*$, (3) and (4) are equivalent to $L(G_1) = \Sigma^*$. By previous problem each of these problems is undecidable.

45. It is undecidable for arbitrary CFG's G_1 and G_2 whether (*a*) $\overline{L(G_1)}$ is a CFL (*b*) $L(G_1) \cap L(G_2)$ is a CFL.

Solution.

(*a*) Given an arbitrary TM M, modify without changing, the set accepted, so that M makes atleast two moves on every input. Construct CFG G generating the invalid computations. $\overline{L(G)}$ is a CFL if and only if M accepts a finite set.

(*b*) Proceed as in (*a*), but construct CFG's G_1 and G_2 such that $L(G_1) \cap L(G_2)$ is the set of valid computations of M.

46. Prove that any language generated by an unrestricted grammar is recursively enumerable.

Solution. The grammar in effect defines a procedure for enumerating all strings in the language systematically. For example, we can list all w in L such that

$$S \rightarrow w,$$

that is w is derived in one step. Since the set of the productions of the grammar is finite, there will be a finite number of such strings. Next we list all w in L that can be derived in two steps.

$$S \rightarrow y \rightarrow w,$$

and so on. We can simulate these derivations on a Turing Machine and therefore, have an enumeration procedure for the language. Hence it is recursively enumerable.

EXERCISES

1. Which of the following properties of recursively enumerable sets are themselves recursively enumerable.

   (*a*) L contains atleast two strings

   (*b*) L is infinite

   (*c*) $L = L^R$