CSG is called monotonic, or length increasing.

5/5/22

Theorem 2: Every monotonic grammar G is equivalent to Type-1 grammar.

Proof. By applying the above theorem, let G_1 be obtained.

We construct $G' \equiv G_1$ as follows.

Consider a production

$A_1 A_2 \dots A_m \to B_1 B_2 \dots B_n$ with $n \geq m$ in G_1.

If $m = 1$, then the above production is of Type-1 with left and right context being $\in$. If $n \geq 2$ corresponding to the production

$A_1 A_2 \dots A_m \to B_1 B_2 \dots B_n$

we construct the following Type-1 productions introducing new variables $C_1, C_2 \dots C_m$.

$A_1 A_2 A_3 \dots A_m \to C_1 A_2 \dots A_m$.

$C_1 A_2 \dots A_m \to C_1 C_2 \dots A_m$.

$C_1 C_2 A_3 \ldots A_m \rightarrow C_1 C_2 C_3 A_4 \ldots A_m \ldots$
(as $n \geq m$).

$C_1 C_2 \ldots C_{m-1} A_m \rightarrow C_1 C_2 \ldots C_m B_{m+1} \ldots B_n$.

$B_1 C_2 C_3 \ldots B_n \rightarrow B_1 B_2 C_3 \ldots C_n$.

$B_1 B_2 \ldots C_m B_{m+1} \ldots B_n \rightarrow B_1 B_2 \ldots B_m \ldots B_n$.

The above construction can be explained as:

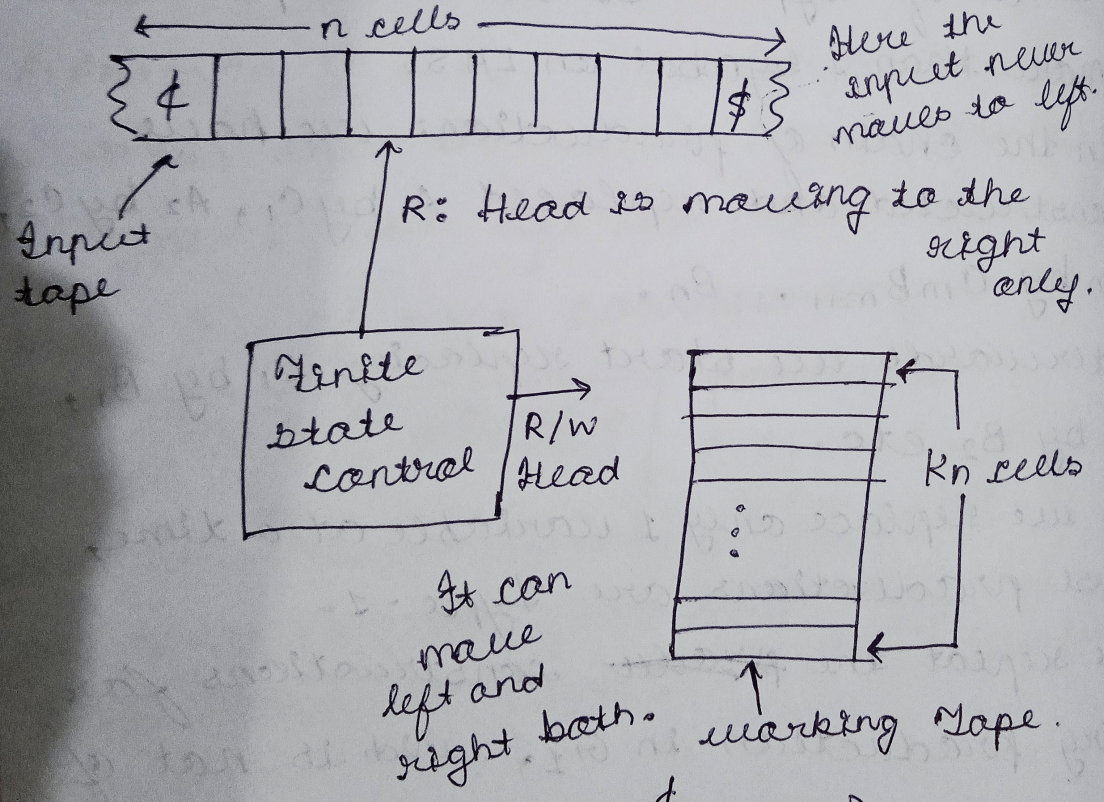
(i) The production $A_1 A_2 \ldots A_m \rightarrow B_1 B_2 \ldots B_n$ is not of Type-1. As we can replace more than 1 symbol on LHS.
In the chain of productions we have constructed and replaced A_1 by C_1, A_2 by C_2, A_m by $C_m B_{m+1} \ldots B_n$.
Afterwards we start replacing C_1 by B_1, C_2 by B_2 etc.
As we replace only 1 variable at a time, these productions are Type-1.
We repeat the ~~procedu~~ constructions for every production in G_1, which is not of type-1. For the new gramar G', the variables are the variables of G_1 together

with the new variables. The productions of G' are the new type-1 productions obtained through the above construction.

T and S of G' and G_1 are same.

G' is context sensitive., and ~~Ø~~ from the above constructions, $L(G_1) = L(G') = L(G)$.

Model of Linear Bounded Automata (LBA)



$M = (Q, \Sigma, \Gamma, \delta, q_0, b, ¢, \$, F)$.

Non-determ
which has
length not

b → blank,

$b \notin \Sigma$.

(q, x) ant

Both end
any other
R/w head
symbol
$|w| = n -$
w can
accepted
tape an

The la
input
prope

Non-deterministic Turing Machine → LBA, which has a single tape and length not infinite.

b → blank, used to indicate no symbol.

$b \notin \Sigma$.

(q, x) onto (q', y, D).
- y → top of stack.
- D → direction of movement of R/W head either left or right.

Both end markers should not appear in any other cell in the input tape.

R/W head should not print any other symbol over the end markers.

$|w| \leq n-2$. w = input string.

w can be recognised by a LBA if it is accepted by a Turing machine of single tape and restricted length.

> The value K does not depend on the input string, but is purely a property of the machine.

Instantaneous Description → 10/5/22

(q, w, k) $q \in Q$, $w \rightarrow$ string, $k \in Z$.

↓ used to denote direction, left / right.

$k - 1 \rightarrow$ right

$k + 1 \rightarrow$ left

The language accepted by LBA is defined by the state $\{w \in (\Sigma - \{¢, \$\})^* |$

$(q_0, ¢w\$, 1) \overset{M}{\vdash^*} (q, \alpha, i)\}$.

A linear bounded automaton M accepts a string w if after starting at initial state with R/w head reading the left end marker, M halts over the right end marker in a final state. Otherwise it is rejected.

A language L is CSL if there exists a CSG, G such that $L = L(G)$ or $L = L(G) \cup \{\in\}$.

LBA is always non-deterministic.

CFG is a subset of CSG.

eg: $L = \{a^m b^n c^{m+n} : m, n \geq 0\}$.

m no. of a's pushed.

n no. of b's pushed.

m+n no. of c's popped.

So this is a CFL.

eg: $L = \{a^n b^n c^n : n \geq 1\}$.

$S \to abc \mid aAbc$

$Ab \to bA$

$Ac \longrightarrow Bbcc$

$bB \to Bb$

$aB \to aa \mid aaA$.

$w = a^3 b^3 a^3$.

$S \Rightarrow aAbc \Rightarrow abAc \Rightarrow abBbcc$

$\Downarrow$

$aBbbcc$

$\Downarrow$

$aaAbbcc$.

$\Downarrow$

$aabAbcc$

$\Downarrow$

$aabbAcc$

$\Downarrow$

$aabbBbccc$

$\Downarrow$

$aabBbbccc$

$\Uparrow$

$aaabbbccc$.

Theorem:

For every CSL, L not including ϵ, there exists some LBA, M such that $L = L(M)$

Proof: If L is a CSL, then there exists a CSG, $L - \{\epsilon\}$. We show that derivations in this grammar can be simulated by a LBA.

The LBA will have two tracks. One containing input string w, other containing sentential forms ~~can write~~ derived using G.

A key point of this argument is that no possible sentential form can have length greater than |w|. ~~LBA~~

LBA is non-deterministic. This is necessary in the argument since we claim that the correct production can always be guessed, and no unproductive alternatives will be there.

Note: Any language generated by unrestricted grammar is recursively enumerable. (Type-0).
It can be carried out without using space except that originally occupied by w, i.e. it can be done by LBA.

11/5/22

Theorem: Every CSL is recursive.

Theorem: There exists a recursive language that is not context-sensitive.

Proof 1: For CSL, L there exists a CSG, G. Consider a string w, for which derivation is as follows:

$S \Rightarrow x_1 \Rightarrow x_2 \Rightarrow \cdots \Rightarrow w.$

The derivations are bound on number of steps and we know $|x_i| \leq |x_{i+1}|$. (since G is non-contracting).
We can check whether w is in L(G) as follows:
i) construct a transition graph whose vertices are the strings of length $\leq |w|$.

ii) paths correspond to derivation in grammars.

iii) add edge from x to y if $x \Rightarrow y$.

iv) w belongs to $L(G)$ iff there is a path from S to w.

v) use path-finding algorithm to find a path.

Proof-2: Consider L is recursive.

create possible CSG, ~~$G_i = \{Q_i, \Sigma_0$~~

$G_i = (Q_i, \{0,1,2,3,4,\ldots,9\}, S_i, P_i)$.

generates numbers

Define language L, which contains the numbers of the grammar which does not generate the number of its position in the list.

$L = \{i \mid i \notin L(G_i)\}$.

We can create a list of all ~~CS~~ context sensitive generating grammars, which generates numbers, we can decide

whether or not the CSG generates its position in the list. So language L is recursive.

~~Assume~~ language L is not CS:

Assume for contradiction that L is a CSL. So there is a CSG, G_K such that $L(G_K) = L$ for some K.

If $K \in L(G_K)$, by def. of L, we have $K \notin L$.

But $L = L(G_K)$.

So here we have a contradiction.

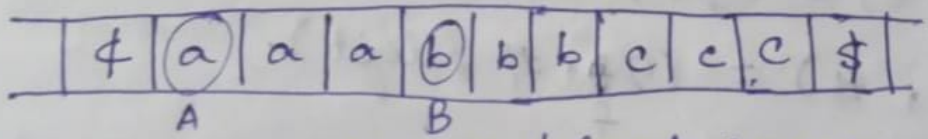
If we consider $K \notin L(G_K)$ then $K \in L$.

This is also a contradiction as

$L = L(G_K)$.

Example of LBA:

$L = \{ a^i b^i c^i \mid i \in \mathbb{N}. \}$.



is used to denote blank space.

$\delta(q_0, a) = (q_1, A, \text{right})$.

$\delta(q_0, \#) = (q_f, \#, \text{right})$.

$\delta(q_1, a) = (q_1, a, \text{right})$.

$\delta(q_1, b) = (q_2, B, \text{right})$.

$\delta(q_2, b) = (q_2, b, \text{right})$.

$\delta(q_2, c) = (q_3, c, \text{left})$.

$\delta(q_0, B) = (q_0, B, \text{left})$.

$\delta(q_0, c) = (q_0, c, \text{left})$

$\delta(q_1, B) = (q_1, B, \text{right})$

$\delta(q_2, c) = (q_2, c, \text{right})$

$\delta(q_3, a) = (q_3, a, \text{left})$

$\delta(q_3, b) = (q_3, b, \text{left})$

$\delta(q_3, c) = (q_3, c, \text{left})$.

$\delta(q_3, B) = (q_3, B, \text{left})$.

$\delta(q_3, A) = (q_0, A, \text{right})$.

$\delta(q_0, A) = (q_4, A, \text{right})$.

$\delta(q_4, B) = (q_4, B, \text{right})$.

$\delta(q_4, c) = (q_4, c, \text{right})$.

$\delta(q_4, \#) = (q_f, \#, \text{left})$.

Theorem: The class of languages accepted by LBA and the class of CSL coincide.

12/5/22

Examples of CSL:

a) $\{a^i b^j a^i b^j \mid i, j \in \mathbb{N}\}$.

b) $\{a^{x^i} \mid i \in \mathbb{N}\}$.

c) $\{a^{n!} \mid n \geq 0\}$.

d) $\{a^n \mid n = m^2, m \geq 1\}$.

e) $\{a^n \mid n \text{ is prime}\}$

f) $\{a^n \mid n \text{ is not prime}\}$.

g) $\{ww \mid w \in (a,b)^+\}$

h) $\{w^n \mid w \in (a,b)^+, n \geq 1\}$.

i) $\{www^R \mid w \in (a,b)^+\}$.

John C. Martin → Automata Language & Theory of computation.

Properties of CSL

i) CSL are closed under union, concatenation, intersection, complement, Kleene's closure, reversal.

But not closed under other operations.

Converting CSL into Kuroda Normal Form.

to prove that two CSL are closed under union.

A grammar is said to be in KNF, when all its productions are of the form

$AB \to CD$ /

$A \to BC$ /

$A \to B$ /

$A \to a$

$G_1 = (V_1, T, S_1, P_1) \Rightarrow L_1$

$G_2 = (V_2, T, S_2, P_2) \Rightarrow L_2$.

$V_1 \cap V_2 = \phi$.

$G_1 \cup G_2 = G$. $\Rightarrow L$ which is accepted by either L_1 or L_2.

$G = (V_1 \cup V_2 \cup \{S\}, T, S, P_1 \cup P_2 \cup \{S \to S_1, S \to S_2\})$

where $S \notin V_1 \cup V_2$.

G_1 generates a language $L_1 \cup L_2$, which is context sensitive.

If $L_1 \cup L_2$ contains empty string, then the production rules will be

$$P_1 \cup P_2 \cup \{S \to S_1, S \to S_2, S \to \epsilon\}.$$

$$\setminus \{S_1 \to \epsilon, S_2 \to \epsilon\}.$$

as these are already included in $P_1 \cup P_2$, so these productions became redundant.

Kleene's closure:

$G_1 = (V_1, T, S_1, P_1) \Rightarrow L_1$

$G_2 = (V_2, T, S_2, P_2) \Rightarrow L_2$

$V_1 \cap V_2 = \phi$.

$G = (V_1 \cup V_2 \cup \{S, S'\}, T, S, P_1 \cup P_2 \cup \{S \to \epsilon, S \to S_1, S \to S_1 S_2, S \to S_1 S_2 S', S' \to S_1, S_2 \to S_1 S_2, S' \to S_1 S_2 S'\} \setminus \{S_1 \to \epsilon, S_2 \to \epsilon\})$.

where $S, S' \notin V_1 \cup V_2$.

$G \Rightarrow L'$.

Using these productions, we can derive all strings in L^* using G^*.

Suppose x is derived by G_1. There are only 3 ways for this to occur.

i) By an application of $S \to \epsilon$, we have $x = \epsilon$ which is in L^*.

ii) By an application of $S \to S_1$, then a derivation using G_1 we have $x \in L \subset L^*$.

iii) By an application of $S \to S_1 S_2$ followed by 0 or more applications of $S_2 a \to S_2 S a$; $a \in T$. Then one application of $S_2 b \to S b$; $b \in T$. The result is a sentential form of G^*, $x_1 x_2 \dots x_n$ where each x_i is a sentential form of G that has the form $a x_1 x_2 \dots x_n$ where $a \in T$, $x_i \in T \cup V$.

This is because of our assumption that terminal symbols only occur on left of the productions of G and that $\{S_1, S_2\} \cap V = \phi$, where $V = V_1 \cup V_2$.

This also means that from $x_1 x_2 \dots x_n$ we can derive the string $b_1 b_2 \dots b_n$ where $b_i \in L$. So $L(G^*) = L^*$.

Suppose L contains ϵ, then $L - \{\epsilon\}$ is CSL. We can simply delete the production $S \to \epsilon$. Since S is not at the RHS of any production, $L - \{\epsilon\}^*$ is context sensitive. $\Rightarrow L^*$ is context sensitive.