M T W T F S S

Page No.: YOUVA

Date:

7/2/22

- Unrestricted grammars

$(V \cup T)^+ \rightarrow (V \cup T)^*$

$T = \{t_1, t_2, t_3\}$ $\quad V = \{S, A, B\}$

$At_1t_2t_3 \rightarrow \varepsilon$ $\quad t_1ABt_2 \rightarrow At_3B$

Note: In context free grammers transition was from $(V)^+ \rightarrow (V \cup T)^+$

eg. $L = \{a^i \mid i \text{ is a power of } 2\}$

a, aa, aaaa, aaaa aaaa, ... accepted ones

$S \rightarrow ACaB$
$Ca \rightarrow aaC$
$CB \rightarrow DB$
$CB \rightarrow E$
$aD \rightarrow Da$
$AD \rightarrow AC$
$aE \rightarrow Ea$
$AE \rightarrow \varepsilon$

$S \Rightarrow ACaB$
$\Rightarrow AaaCB \Longrightarrow AaaE$ (will see later)
$\Rightarrow AaaDB$
$\Rightarrow AaDaB$
$\Rightarrow ADaaB$
$\Rightarrow ACaaB$
$\Rightarrow AaaCaB$
$\Rightarrow AaaaaCB$
$\Rightarrow AaaaaE$
$\Rightarrow AaaaEa$
$\overset{3}{\Rightarrow} AEaaaa$
$\Rightarrow aaaa$

Date:

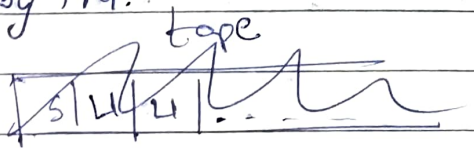
To prove
Unrestricted grammar ~~have~~ has same computational power as TMs. or they compute partial recursive functions.

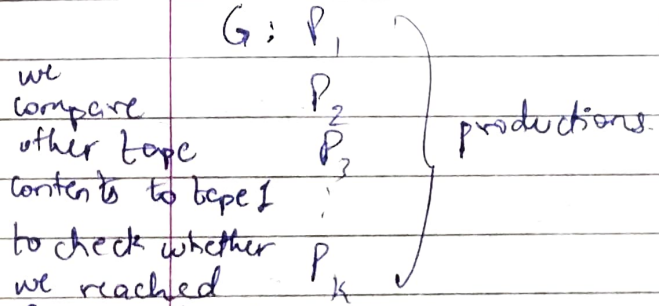
Part 1

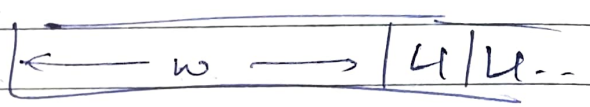
~~Using Unrestricted grammar, can we simulate TM.~~
Given a language, we will come up with turing machine doing the same

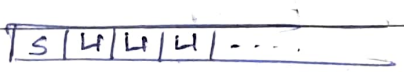
Objective: Using 0 or more production generating a $w \in L(G)$. such that w is accepted by TM.
OR w must be accepted by TM.

Starting with S



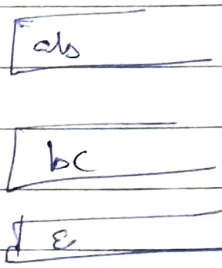




first we will search for S in left hand side in productions. (non deterministically)
Let there be 3 of them
$S \to ab \mid bc \mid \varepsilon$
meaning, apply all possible productions in different tapes

We don't know which will lead us to final answer, so non-deterministically we will guess them generating 3 different instances.




~~Now let~~ The same rule for threads applies, if one dies, then others continue, and if one is accepted then input is accepted and all threads are killed

If ~~the~~ at some point we have.

tape is

| a | a | B | a | C | d | a | C | d | ... |
|---|---|---|---|---|---|---|---|---|---|

$P_1$
$P_2$
$P_3$
$P_4$

We will go through this string searching for $P_1$ if possible, then $P_2$, and so on, and generate new threads if possible

Lets say $P_2$ is $Cd \rightarrow$ (something)
So it will also search for positions non deterministically. In above string we have 2 Cd's so one more thread will fork and both cases will be followed

So ~~not only~~ there are 2 non-determinisms here
i) searching for production
ii) searching for different possible production sites available in the substring

When we are replacing in TMs, we are eventually simulating the productions.

$$\delta(q, aCd) \rightarrow (\quad)$$

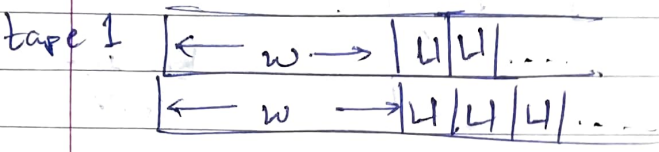
Here ~~only~~ its final transistion I wrote, but in some 2,3 or in any finite number of steps we can achieve it.

So ultimately we will have a set of transistions for a TM.
Thus ~~we~~ given a language, we came up with a turing machine which can accept that language.

Part 2

Given a DTM, we will use unrestricted grammar to simulate it.

Suppose we had input in tape 1 (in DTM)




Copy its contents to tape 2.

Let us try to generate the corresponding unrestricted grammar

Note: I will denote all productions in boxes. And we will list others as and when necessary

$S \to q_0 A_1$

$A_1 \to [a_i, a_i] A_2$

$A_2 \to \varepsilon$

condition → for all $a_i$ → $\forall\, a_i \in \Sigma$

for having concrete simulation.

let tape 1 with input be

| $a_1$ | $a_2$ | . . | $a_n$ | U | .. |
|---|---|---|---|---|---|
| $a_1$ | $a_2$ | - - | $a_n$ | U | -- |

$S \Rightarrow q_0 A_1$

$\Rightarrow q_0 [a_1, a_1] A_1$

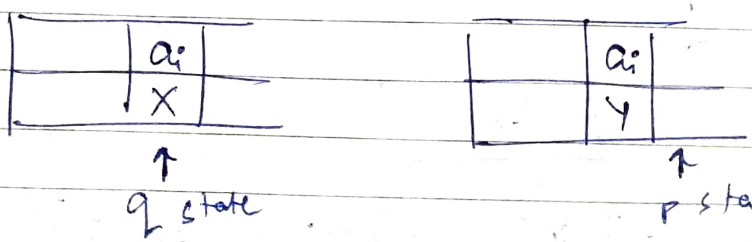
$\Rightarrow q_0 [a_1, a_1][a_2, a_2] A_1$

$\overset{*}{\Rightarrow} q_0 [a_1, a_1][a_2, a_2][a_n, a_n]$

$q_0$ here is starting state

We ~~made~~ produced this much

Now on tape, if it has transition



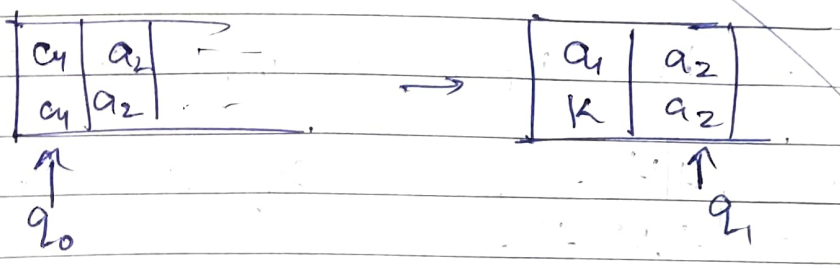

$\delta(q, [a_i, X]) = (p, [a_i, Y], R)$

(We will simulate this now)

$$q[a_i, X] \rightarrow [a_i, Y]_p$$

with changing symbols on tape we are also changing, mentioning the states.

let initial transition be




by this rule,

so in our grammar model.

$$q_0[a_1, a_1][a_2, a_2] \ldots [a_n, a_n] \Rightarrow [a_1, K]\, q_1\, [a_2, a_2] \ldots [a_n, a_n]$$

or (another eg).

at some pt let our production be

$$[a_1, X_1][a_2, X_2] \ldots q[a_i, X] \ldots [a_{n+1}, aX_{n+1}] \ldots [a_s, X_s]$$

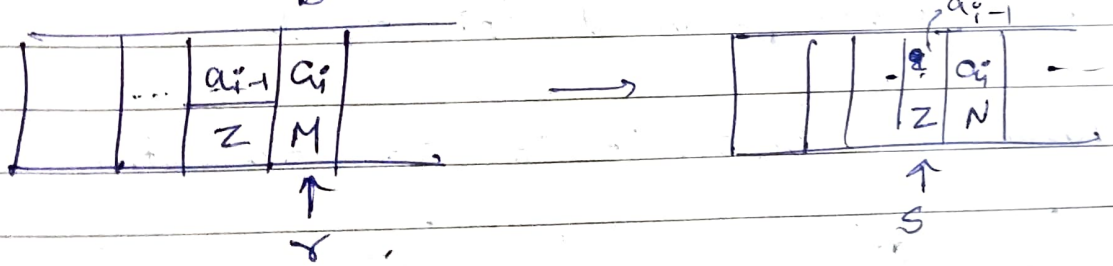
$$\Downarrow$$

$$[a_1, X_1][a_2, X_2] \ldots [a_i, Y]_p \ldots [a_{n+1}, aX_{n+1}] \ldots [a_s, X_s]$$

---

Let if there be left transition i.e. tape head moves to left.

$$\delta(\gamma, [a_i, M]) = (t, [a_i, N], L)$$




Our production will be

$$[a_{i-1}, Z]\, \gamma\, [a_i, M] \rightarrow s\, [a_{i-1}, Z][a_i, N]$$

Subsequent Corresponding effect to our string or produced string.

$$[a_1, X_1][a_2 X_2] \cdots [q_{i-1}, Z][a_i, X] \cdots [a_{n+1}, X_{n+1}] \cdots [a_s, X_s]$$

$$\Downarrow$$

$$[a_1, X_1][a_2, X_2] \cdots [q_{i-1}, Z][a_i, Y] \cdots [a_{n+1}, X_{n+1}] \cdots [a_s, X_s]$$

Lets say while playing with states ~~and t~~ $p, q, r, s$ we come upto an accept state $q'$, ~~so~~ ~~We~~ How we will deal with ~~it is~~ it, will now be done.

$$[a_i, X] q' \rightarrow q' a_i q'$$

$$q' [a_i, X] \rightarrow q' a_i q' \quad \forall a_i \in \Sigma$$

$$q' \rightarrow \epsilon$$

At some point we have

$$[a_1, Y_1] \cdots [a_{t-1}, Y_{t-1}] q' [a_t, Y_t] \cdots [a_s, Y_s]$$

$$\Rightarrow [a_1, Y_1] \cdots [a_{t-1}, Y_{t-1}] q' a_t q' [a_{t+1}, Y_{t+1}] \cdots [a_s, Y_s]$$

$$\overset{*}{\Rightarrow} [a_1, Y_1] \cdots [a_{t-2}, Y_{t-2}] q' a_{t-1} q' a_t q' a_{t+1} q' [a_{t+2}, Y_{t+2}] \cdots [a_s, Y_s]$$

(* denotes 0 or more productions applied)

$$\overset{*}{\Rightarrow} [a_1, Y_1] \cdots [a_{t-2}, Y_{t-2}] q' a_{t-1} a_t a_{t+1} q' [a_{t+2}, Y_{t+2}] \cdots [a_s, Y_s]$$

$$\overset{*}{\Rightarrow} a_1 a_2 \cdots a_n \cdots a_s$$

$a_1 a_2 \cdots a_n$ — our string of interest

$a_n \cdots a_s$ — extra part that may be generated while productions. With additional productions we can prune them. like $\bar{a}_n a_j \rightarrow a_n$ for $j = n+1$ to $j = s$

If we have reached an accept state $q'$, then we will always end up with "$a_1 a_2 \cdots a_n$" with help of subsequent productions.

If we dont end up at "$a_1 a_2 \cdots a_n$", then we can say that w doesn't belong to our grammar And that $w \notin L(M)$, as last state isn't ~~accept~~ is ~~ed~~ reject state or ~~does~~ machine does not halt. ~~state.~~

Thus we proved both parts and can say TMs and unrestricted grammars have same power.

# CS205

M T W T F S S
Page No.:
Date:
YOUVA

- Regular Expressions

$\Sigma$ is the alphabet
regular expression over $\Sigma$ and the sets they denote are defined recursively.

reg. exp is just easy representation for like $\{\alpha\}$ to $\alpha$.

$\emptyset$ is RE and denotes empty set
$\varepsilon$ is RE and denotes $\{\varepsilon\}$
$\forall \alpha \in \Sigma$, $\alpha$ is a RE and denotes $\{\alpha\}$

For any r, s regular expression denoting set R and S, then.

- $(r \cup s)$ or $(r+s)$ is RE that denotes $R \cup S$
  - ∪ union
- $rs$ is a RE that denotes RS
  - ↱ concatenation

$RS = \{xy \mid x \in R, y \in S\}$.

eg $R = \{1, 2, 3\}$ $S = \{a, b, c\}$
$RS = \{1a, 1b, 1c, 2a, \ldots, 3c\}$

* = zero or more
\+ = one or more

- $r^*$ is a RE denoting the set $R^*$

$R^*$ : kleene closure of L

$L^0 = \{\varepsilon\}$
$L^i = L \cdot L^{i-1}$ eg $L^3 = L L^2$

eg $L = \{1, 2\}$
$L^0 = \{\varepsilon\}$
$L^1 = L L^0$
$= \{1, 2\}.\{\varepsilon\} = \{1, 2\} = L$
$L^2 = \{1, 2\}\{1, 2\} = \{11, 12, 21, 22\}$
$L^3 = \{1, 2\}\{11, 12, 21, 22\}$
$= \{111, 112, 121, 122, 211, 212, 221, 222\}$

$$L^* = \bigcup_{i=0}^{\infty} L^i$$

$$= L^0 \cup L^1 \cup L^2 \cup \ldots$$

$r^+$ is a regular exp. denoting the set $R^+$ ] positive closure

$$R^+ = \bigcup_{i=1}^{\infty} R^i$$

$$= R \cup R^2 \cup \ldots$$

eg. $00$ is RE of $\{00\}$.

eg. $(0+1)^*$ $\Sigma = \{0,1\}$

$= (\{0\} \cup \{1\})^*$

$= (\{0,1\})^*$

Kleene closure of $\{0,1\}$

$\{0,1\}^0 = \{\varepsilon\}$ $\{0,1\}^1 = \{0,1\}$

$\{0,1\}^2 = \{00,01,10,11\}$ $\{0,1\}^3 = \{0,1\}\{0,1\}^2$

so $(0+1)^*$ denotes all strings of $\Sigma$.

eg. $(0+1)^* 00 (0+1)^*$

$(0+1)^*$ denotes all strings of $\Sigma$

$\Sigma^* \{0\}\{0\} \Sigma^*$

$\Sigma^* \{00\} \Sigma^*$

$= \{ xyz \mid x \in \Sigma^*, y \in \{00\}, z \in \Sigma^* \}$

{ any string of $\Sigma$ which has 00.

$= \{ x\,00\,z \mid x, z \in \Sigma^* \}$

eg. $(1+10)^*$ $= (\{1\} \cup \{1\}\{0\})^*$

$= (\{1\} \cup \{10\})^*$

$= (\{1\} \cup \{10\})^0 \cup (\{1\} \cup \{10\})^1 \cup (\{1\} \cup \{10\})^2 \cup \ldots\ldots$

$= \{\varepsilon\} \cup \{1, 10\} \cup \{11, 110, 101, 1010\} \cup \ldots\ldots$

observation : strings always start with 1, never has 00 as a substring.

eg. $(0+\varepsilon)(1+10)^*$

$= (\{0\} \cup \{\varepsilon\})(\{1\} \cup \{10\})^*$

$= (\{0, \varepsilon\})(\{1\} \cup \{10\}^*$

effective expansion

$= 0x, 0y, \ldots, x, y, \ldots$

So by this concatenation, we included strings starting with 0 and retained starting with 1 with the help of $\varepsilon$

eg. $0^*1^*2^*$ : $\{0\}^*\{1\}^*\{2\}^*$

$= \left(\bigcup_{i=0}^{\infty} \{0\}^i\right)\left(\bigcup_{i=0}^{\infty} \{1\}^i\right)\left(\bigcup_{i=0}^{\infty} \{2\}^i\right)$

$= (\{\varepsilon\} \cup \{0\} \cup \{00\} \cup \{000\} \cup \ldots\ldots)$
$(\{\varepsilon\} \cup \{1\} \cup \{11\} \cup \{111\} \cup \ldots\ldots)$
$(\{\varepsilon\} \cup \{2\} \cup \{22\} \cup \{222\} \cup \ldots)$

$= (\{\varepsilon\}\{\varepsilon, 0112, 000112, 001122 \ldots\}$