"Unrestricted" grammars

A grammar is a set of variables (some of which are terminals/letters), a start variable, and some (a finite # of) production rules

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

(V = set of variables
T = set of letters)

E.g.

$$S \to ABC$$

$$AB \to xyBA \mid xyB \mid yx$$

$$BC \to BCax \mid xy \mid Cx$$

So we can have a derivation

$$S \to \underline{AB}C \to \underline{xy}\overline{BC} \to \overline{xyxy}$$

(Note - you don't have derivation trees anymore)

<u>Goal</u>: Prove that every TA language is generated by a grammar, and ~~the set~~ any language generated by a grammar is TE (and therefore TA)

---

1) To show any lang. gen by a grammar is TE - we have to construct a TM that "prints out" all the strings in the grammar.

- we do this by thinking of all possible derivs. as a tree and doing a BFS (breadth-first-search) on this infinitely tall but finitely wide tree.
  - any result w/ no variables, "<u>print</u>"

Any derivation you will finish at some point, so this "prints" every string gen by grammar.

2) Suppose we have a TA language (i.e. a TM that accepts some strings and rejects or runs infinitely on others) How do I build a grammar that generates the language?

Start in the middle of the process.

1 Idea #1 ~~instan~~ changes in instant. descr. based on transitions in a TM look like productions in a grammar.

2 Idea #2 - we want to generate all strings (w/ our grammar), but attach some variable tag that says we haven't checked the string yet - run the string through the TM - if it's accepted, remove the variable so that our grammar actually generates the string.

Annoying problem: Our TM ~~&~~ probably destroys the input string.

Easy sol'n: Imagine a 2-tape TM, where we just leave the input undisturbed on tape 1 and destroy it all we want on tape 2.

<u>One implementation:</u>

$$S \longrightarrow S\, V_{\ell\ell} \mid V_{\ell\ell}^{0}$$

(for every letter ℓ in our input alphabet)

This generates all possible input configs - we should start w/ some string to the right of the head, which is in state 0.

($V_{\ell\ell}$ means the tapes on TM has $\boxed{\begin{matrix}\ell\\ \ell\end{matrix}}$)

($V_{\ell,m}^{k}$ means tapes have $\boxed{\begin{matrix}\ell\\ m\end{matrix}}$, head is over that spot w/ state k)

When we have a transition

$$(q_i, a) \longrightarrow (q_j, b, L),$$

we have productions

$$V_{m,p}\, V^i_{\ell,a} \longrightarrow V^j_{m,p}\, V_{\ell,b} \longleftarrow \text{for every possibility for } \ell, m, p.$$

For a transition

$$(q_i, a) \longrightarrow (q_j, b, R),$$

$$V^i_{\ell,a}\, V_{m,p} \longrightarrow V_{\ell,b}\, V^j_{m,p}$$

This lets us get to any possible computation from any possible input
Now we need to actually recover the input string as a string gen by the grammar <u>if</u> we get to a final state.

the index for a final state

$$V^{f}_{\ell,m} \longrightarrow \ell$$

$$\ell\, V_{p,m} \longrightarrow \ell p$$

$$V_{p,m}\, \ell \longrightarrow p \ell$$

Problem we haven't taken care of - the tape has infinite blanks to the side.