CSCE 355
4/1/2024

Closure Properties of CFLs (or lack thereof)

CFLs are closed under $\cup$ (union), concat, $*$-operator

[proof mirrors construction of a CFG from a regex]

~~CFLs~~ Prop: CFLs are closed under string reversal:

If L is CFL, then L^R is a CFL.

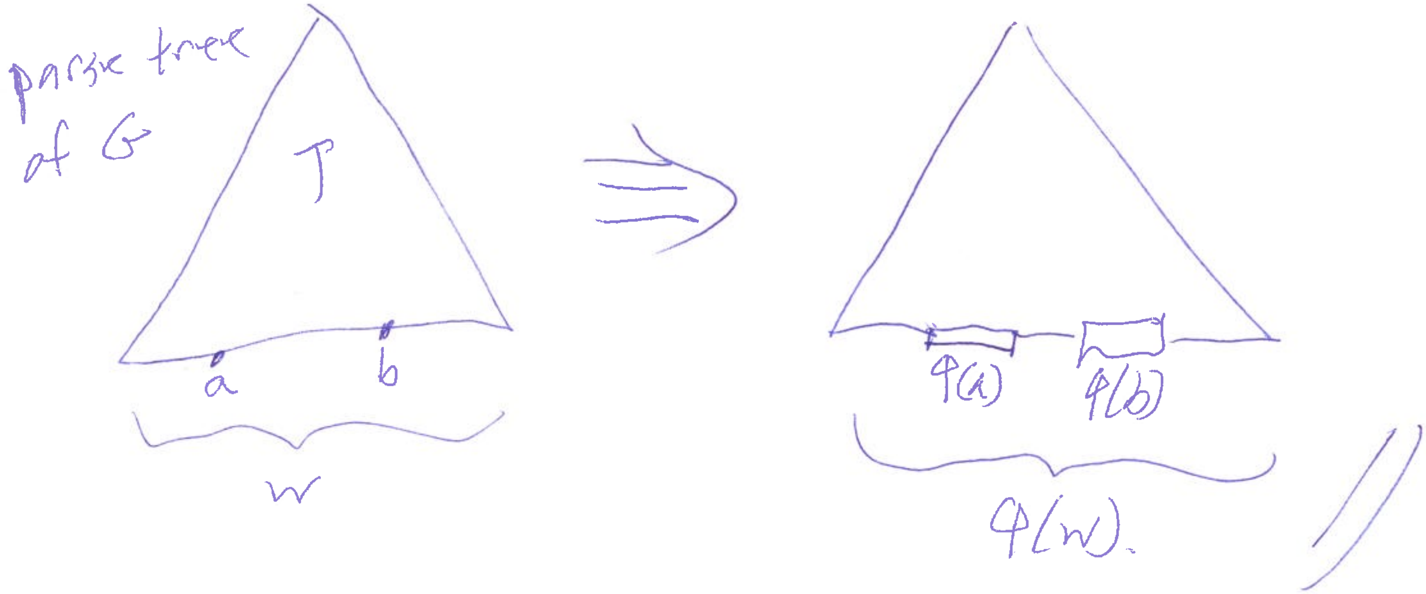
Proof Idea: Given a grammar G for L, form G^R to be the same as G except productions of G^R are of the form $A \to \alpha^R$ for every production $A \to \alpha$ of G.

~~Parses~~ Parse trees of G^R are left-right mirror images of parse trees of G, yielding the reversals of strings in $L(G)$.

$\therefore$ $L(G^R) = \underset{L^R}{\underset{\shortparallel}{L(G)^R}}$ $\quad \therefore L^R$ is a CFL //

Prop: If $L \subseteq \Sigma^*$ is a CFL and $\varphi : \Sigma^* \to \Gamma^*$ is a string homomorphism, then $\varphi(L)$ is a CFL.

Proof Idea: Given a CFG G for L, form a CFG for $\varphi(L)$ by replacing every terminal symbol a in the body of every production with $\varphi(a)$.



<u>Prop</u>: CFLs are closed under inverse homom. images: If $L \subseteq \Gamma^*$ is a CFL and $\varphi : \Sigma^* \to \Gamma^*$ is a string homom., then $\varphi^{-1}(L)$ is a CFL.

[$\varphi^{-1}(L) = \{w \in \Sigma^* : \varphi(w) \in L\}$.

<u>Proof Idea</u>: Given a PDA P for L, construct a PDA P' for $\varphi^{-1}(L)$ that on any symbol $a \in \Sigma \cup \{\varepsilon\}$, mimics what P does on $\varphi(a)$. //

[similar to the proof for reg langs.]

CFLs are not closed under intersection or complement.

For <u>complement</u>: $L := \{ww : w \in \{a,b\}^*\}$ is not CFL-pumpable, $\therefore$ not CFL.

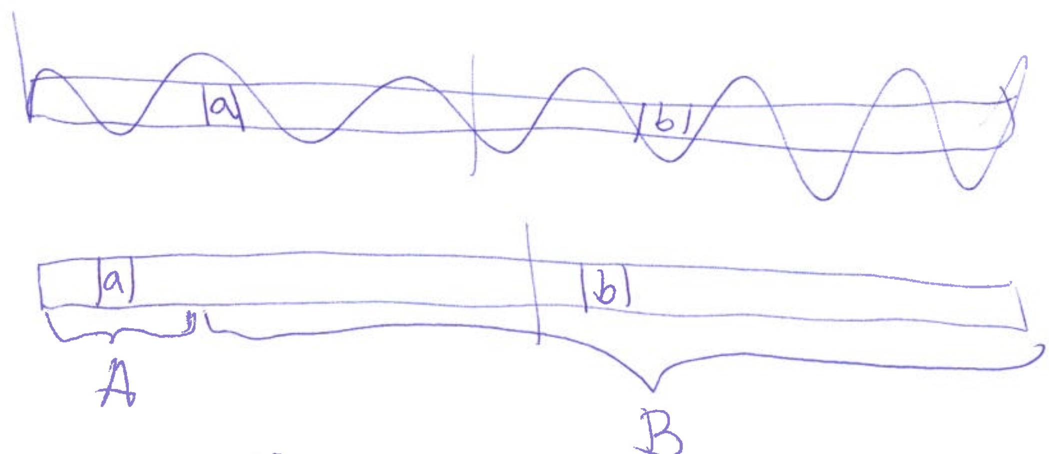
but $\overline{L} := \{x \in \{a,b\}^* : x$ is not of the form $ww\}$ is a CFL:

Here is a grammar for $\overline{L}$:

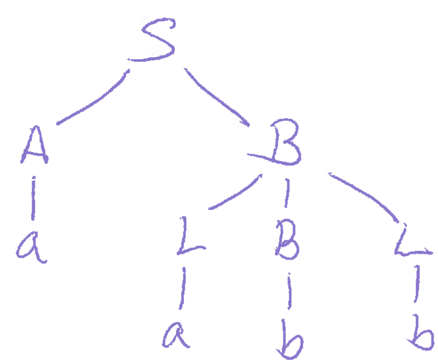
$$
\begin{aligned}
S &\to AB \mid BA \mid O \qquad (O = \text{"odd length"}) \\
L &\to a \mid b \qquad (L = \text{"letter"}) \\
O &\to LLO \mid L \\
A &\to LAL \mid a \\
B &\to LBL \mid b
\end{aligned}
$$

$x \in \overline{L}$

$|x|$ even



aabb :



CFLs not closed under intersection:

Recall: $L := \{a^n b^n c^n : n \geq 0\}$ is not CFL-pumpable, hence not a CFL

$L = L_1 \cap L_2$ for

$$\left.\begin{array}{l} L_1 = \{a^m b^m c^n : m, n \geq 0\} \\ L_2 = \{a^m b^n c^n : m, n \geq 0\} \end{array}\right] \text{CFLs}$$

CFG for L_1:

$$S \to Sc \mid T$$
$$T \to aTb \mid \varepsilon$$

" " L_2

$$S \to aS \mid T$$
$$T \to bTc \mid \varepsilon$$

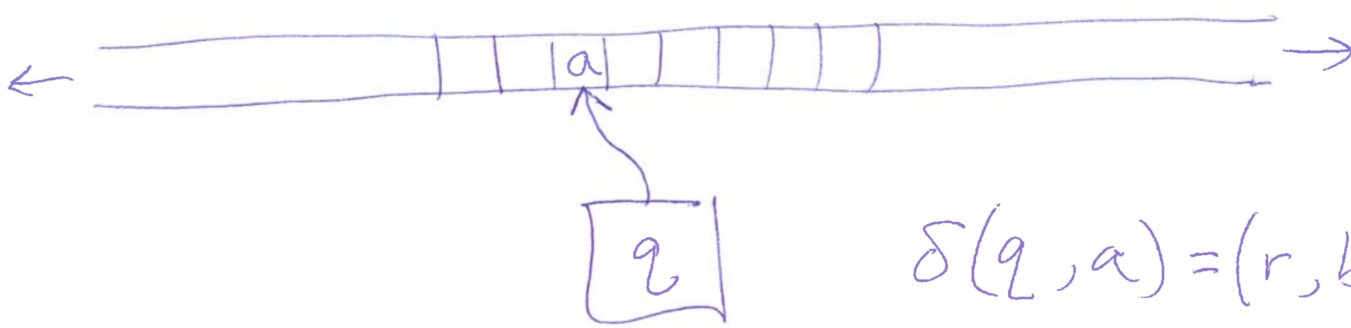
<u>Prop:</u> If $L \subseteq \Sigma^*$ is a CFL and $R \subseteq \Sigma^*$ is a regular language, then $L \cap R$ is a CFL.

<u>Proof idea:</u> The product construction for DFAs also works for a DFA and a PDA.
[the stack for the product is the stack for the PDA component.] //

Turing Machines (TMs)

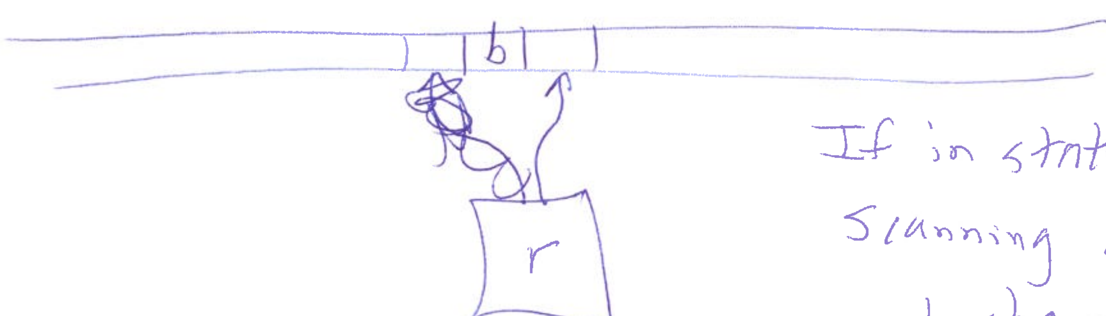
A Turing machine (TM) is a kind of finite-state automaton that can do some more things:

- head can move to right or to the left
- input symbols can be altered
- input is part of an infinite "tape" of writable cells.



$$\delta(q, a) = (r, b, \rightarrow)$$

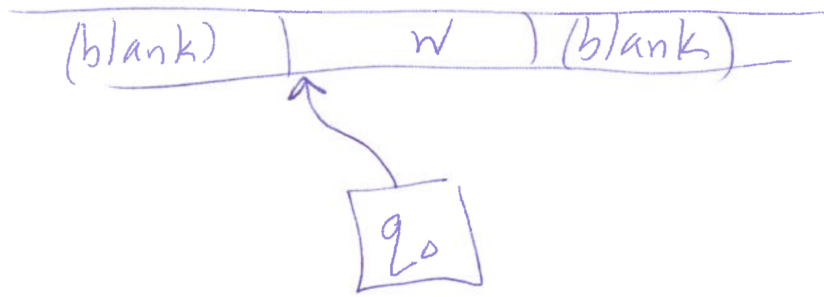
$\Downarrow$



If in state q & scanning a, then

1. change to state r
2. change a to b
3. move head one cell right

Initial conditions: w input string



<u>Def</u>: A <u>Turing machine</u> (<u>TM</u>) is a tuple

$$M := \langle Q, \Sigma, \Gamma, \delta, q_0, B, F \rangle,$$ where

- Q is a finite set (elements are <u>states</u>)
- Σ is an alphabet (the <u>input</u> alphabet; all inputs are strings over Σ)
- Γ is an alphabet (the tape alphabet; possible contents of a cell)

$$\left(\Sigma \subseteq \Gamma \quad \text{and} \quad \Gamma \cap Q = \emptyset \right)$$

- $q_0 \in Q$ (the <u>start</u> <u>state</u>)
- $B \in \Gamma \setminus \Sigma$ (the <u>blank symbol</u>)
- $F \subseteq Q$ (the set of <u>accepting</u> <u>states</u>)

– δ is a *partial* function mapping elements of $Q \times \Gamma$ to elements of $Q \times \Gamma \times \{\leftarrow, \rightarrow\}$

[δ may be *undefined* for some elements of $Q \times \Gamma$

Intended meaning: $\delta(q,a) = (r, b, \rightarrow)$

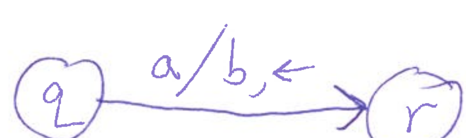
means: if at time t, M's state is q and M's head is scanning a cell containing a, then, at time $t+1$, M's state is r, the contents of the cell becomes b, and head moves ~~one~~ one cell *right*.

Similarly: $\delta(q,a) = (r, b, \leftarrow)$ means

. . . *left*.

Ex:

start state



means $\delta(q,a) = (r, b, \leftarrow)$

A computation ends if $\delta(q, a)$ is undefined. Accepts if $q \in F$ and Rejects otherwise.