chapter 02

Context free Languages.

what did we learn so far?

- DFA / NFA computers with extremely limited memory
- Regular expressions
  - Recipe to describe languages that are regular.

How about creating slightly powerful computational model?

$$A = \{ 0^n 1^n \mid n \geq 0 \}$$

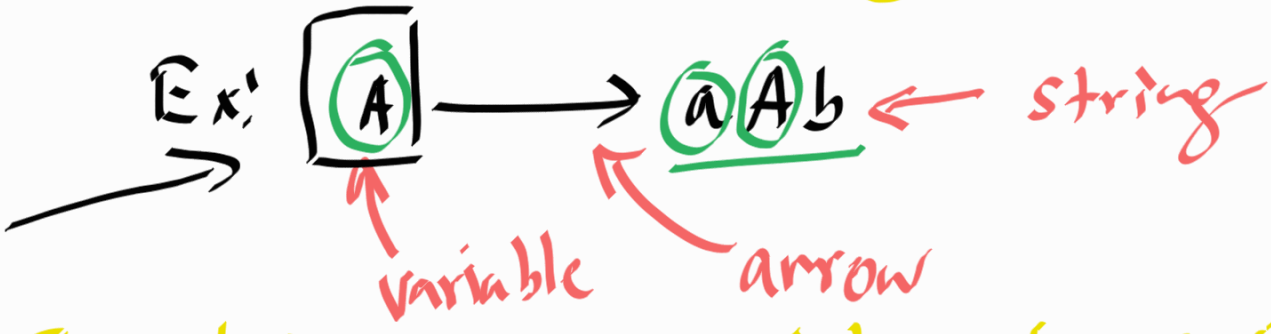
Instead going directly to the new computation model, let's try to learn about something called context free grammar.

Context free grammar to context free languages is what regular expression are to regular languages.

* context free grammar is a collection of substitution rules. also known as productions.

* Each rule appears as a line in the grammar.

* Each line comprises of a symbol, arrow, and a string.



* The string can consist of variables, and other symbols called terminals.

* Variables are often represented using capital letters, terminals are represented using lowercase letters, numbers or special symbols

A ⟶ oAb ⟵ terminal

* One variable is designated starting variable.

What we know now

$F = \{ 0^n 1^n \mid n \geq 0 \}$ is not regular.

starting variable.

1. $A \rightarrow 0A1$
2. $A \rightarrow B$
3. $B \rightarrow \varepsilon$ //

sometimes we use λ, #

$A \Rightarrow 0A1 \Rightarrow 00A11$

$00A11 \Leftarrow 000A111$

$000B111 \Rightarrow 000\varepsilon111 \Rightarrow 000111$

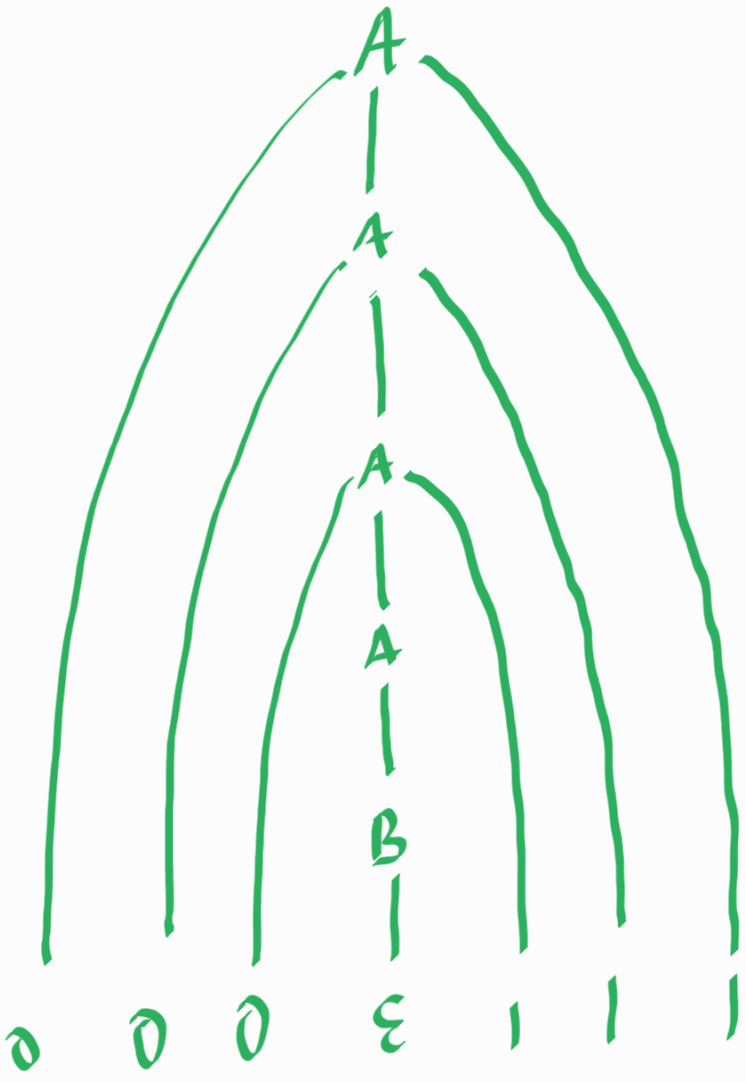
we can represent this info using parse tree as well.

Parse tree....?

$A \rightarrow 0A1$

$A \rightarrow B$

$B \rightarrow \varepsilon$



Any language that can be generated by some context-free grammar is called a **context-free Language**.

Just like regular expressions and regular languages.

CFG is the recipe, CFL is the meal.

Language created by grammar G is denoted as **L(G)**.

Definition

A context-free grammar is a 4-tuple (V, Σ, R, S), where

1. V is the set of variables.
2. Σ is a finite set, called terminals.
3. R is a set of Rules, with each rule being a variable and a string of variables & terminals
4. S is the starting variable.

Example

$G_3 = (\{S\}, \{a, b\}, R, S)$

R:

1. $S \rightarrow aSb$
2. $S \rightarrow SS$
3. $S \rightarrow \varepsilon$

$$S \longrightarrow aSb \mid SS \mid \varepsilon$$

$$S \Rightarrow aSb \Rightarrow aSSb \Rightarrow aaSbSb \Rightarrow aaSb \Rightarrow aabb$$

$$S \overset{*}{\Rightarrow} aabb$$

$a = ($ $b =)$

$)($

aaSb ⇒ aabSb ⇒ aabaSbb

language of all strings with proper nested parenthesis

$G_4 = (V, \Sigma, R, \langle Expr \rangle)$ $\langle EXPR \rangle = E$

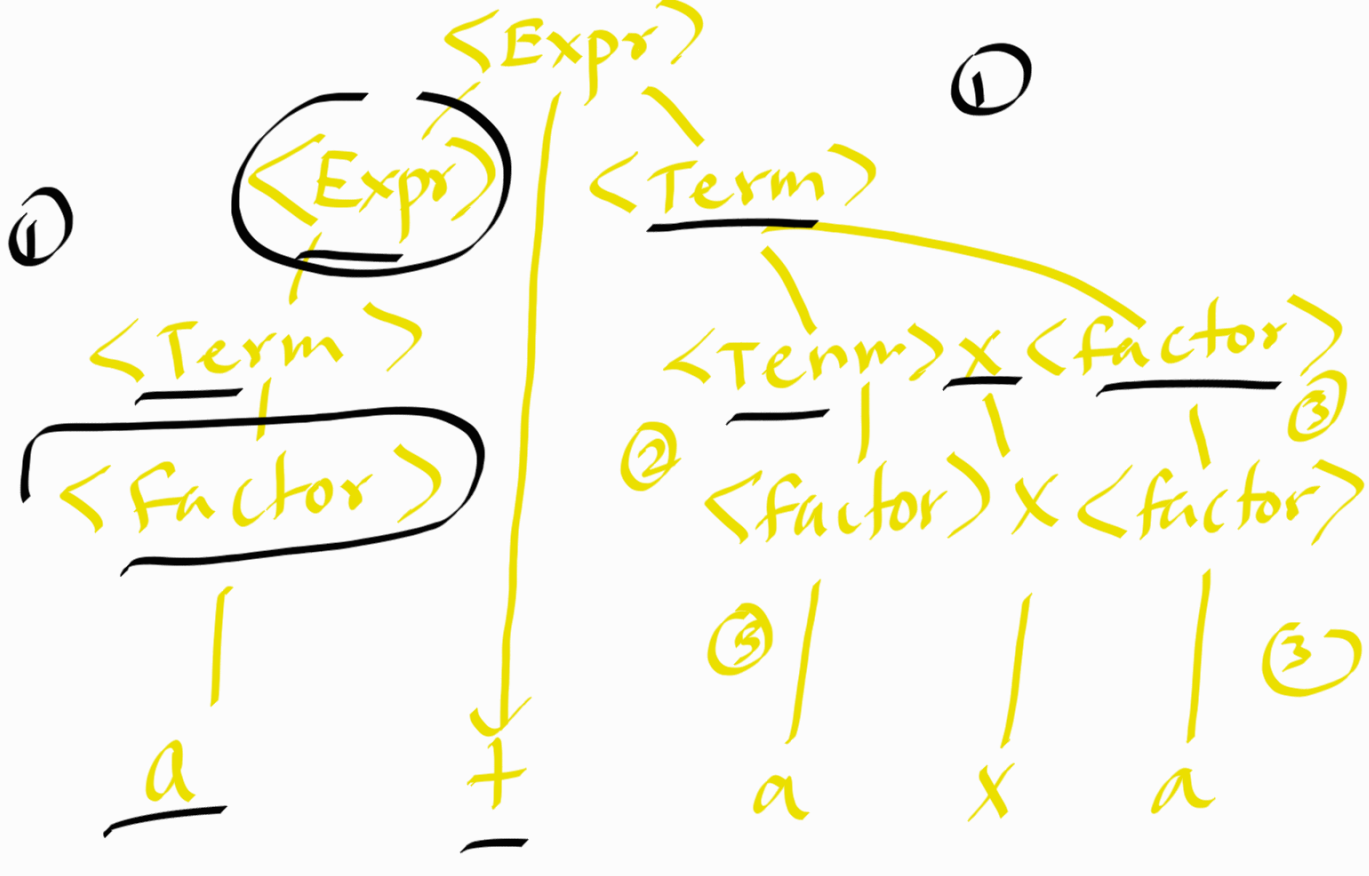
$V = \{\langle Expr \rangle, \langle Term \rangle, \langle Factor \rangle\}$

$\Sigma = \{a, +, x, (,)\}$

1. $\langle Expr \rangle \longrightarrow \langle Expr \rangle + \langle Term \rangle \mid \langle Term \rangle$

2. $\langle Term \rangle \longrightarrow \langle Term \rangle x \langle Factor \rangle \mid \langle Factor \rangle$

3. $\langle Factor \rangle \longrightarrow (\langle Expr \rangle) \mid a$

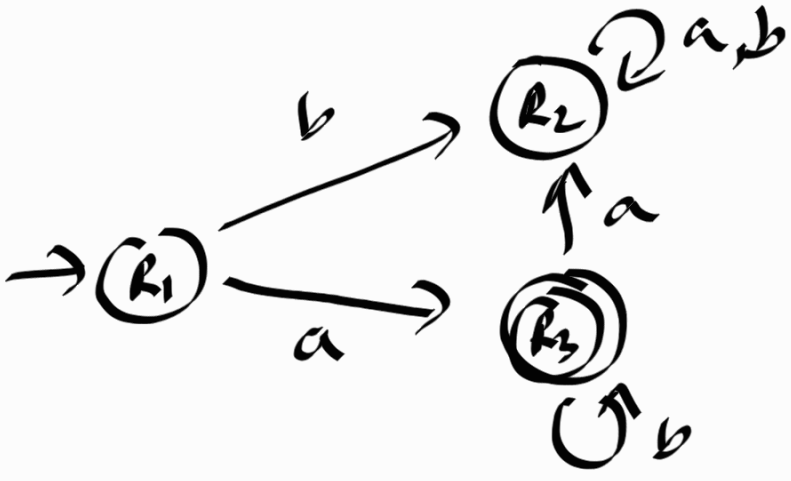


Side Note:

we can easily represent any regular language using context free grammar.

If you have a DFA, you can easily convert it to a grammar

Ex: DFA → Grammar



Regular Grammar

$R_1 \longrightarrow a R_3$

$R_1 \longrightarrow b R_2$

$R_2 \longrightarrow a R_2 \mid b R_2$

$R_3 \longrightarrow b R_3 \mid a R_2$

$R_3 \longrightarrow \varepsilon$

- Designing context free grammar is tricky

- You need to be able to identify recursive structure in your language.
- Some CFL are union of simpler CFLs
- We can start by constructing CFG for smaller CFL and then merging them together.

Designing context free grammer

- $\Sigma = \{a, b\}$

Ex 01: $A = \{w \mid$ the length of w is even $\}$

$S \rightarrow aSa \mid aSb \mid bSa \mid bSb \mid \varepsilon$

$\rightarrow aaS \mid abS \mid baS \mid bbS$ ✓

Ex 02: $B = \{ w \mid w$ starts and ends with the same symbol $\}$

$\Sigma = \{a, b\}$

$\varepsilon, a, b, aa, bb, \cancel{ab}, \cancel{ba}$

$\underbrace{a \ldots\ldots a}$ $\quad$ $\underbrace{b \ldots\ldots b}$

$S \rightarrow \varepsilon \mid a \mid b \mid aTa \mid bTb$

$T \rightarrow aT \mid bT \mid \varepsilon$

Ex 03!

$C = \{ w \mid w$ contains twice as many a's as b's $\}$

✓ ε ~~aaa~~ aab ~~abb~~

$S \rightarrow SaSaSbS \mid SaSbSaS \mid SbSaSaS \mid \varepsilon$

Remember

All regular languages can be expressed by context free grammar.

Esentially, CFG is more powerful than regular expressions.