1. Context-free Grammar (CFG)

$E \to E+T \mid T$
$T \to T \times F \mid F$
$F \to (E) \mid a$

$E \to E+T \to T+T \to F+T \to a+T \to a+F \to a+(E) \to a+(T) \to a+(T \times F) \to a+(F \times F) \to a+(a \times F) \to a+(a \times a)$

$E \xrightarrow{*} a+(a \times a)$

Formal Definition of CFG, $G=(V, \Sigma, R, S)$

→ V - Set of Variables (Non-Terminals) } finite
→ Σ - Set of Terminals ($V \cap \Sigma = \{\}$) }
→ R - Set of Rules (Productions)
→ S - Start Variable ($S \in V$)

Def. Language of a grammar = $\{w \mid w \in \Sigma^* \text{ and } S \xrightarrow{*} w\}$

Def. Context-free language is language generated by a CFG.

$\{0^n 1^n \mid n \geq 0\}$ Language = $S \to \varepsilon \mid 0S1$ grammar

Def. Ambiguous grammar – some string can be derived in many ways

Th. Every regular language is context-free.

Proof by construction

state → variable, edge → rule, accept state → EPSILON rule

2. CFG Closure

S_1, S_2 – context-free languages

Union: $S \to S_1 \mid S_2$ => closed

Concatenation: $S \to S_1 S_2$ => closed

Intersection: not necessarily closed } $A \cap B = \overline{\bar{A} \cup \bar{B}}$

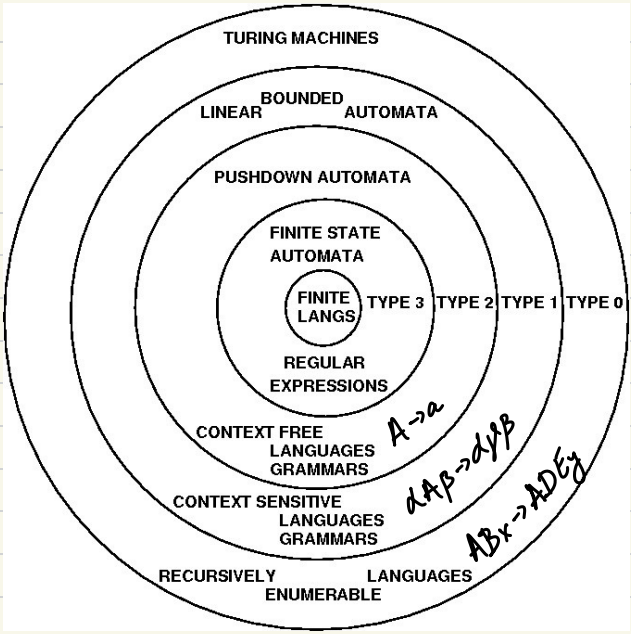
Complement: not necessarily closed }

3. Chomsky Normal Form (CNF)

$A \to BC$ (exactly two variables $\neq S$) or $A \to a$ (terminal); ε only in $S \to \varepsilon$

Th. Every CFG can be generated by a grammar in CNF.

1) no S on right side
2) remove rules with ε on right side ($A \to \varepsilon$)
3) remove unit rules ($A \to B$)
4) get rid of rules with more than 2 symbols
5) make sure right side has either 2 variables or 1 terminal



Chomsky Hierarchy

Pumping Lemma

A-CFL, $|s| \geq p \rightarrow s = uvxyz$ s.t.

1) $uv^i xy^i z \in A$, $\forall i \geq 0$
2) $|vy| > 0$
3) $|vxy| \leq p$

4. Push Down Automaton (PDA) - non-deterministic

$\circ \xrightarrow{a, b \rightarrow c} \circ$ a - input symbol (ε allowed)

b - top of stack (popped); $b = \varepsilon$ - don't read/pop stack

c - pushed onto stack; $c = \varepsilon$ - nothing pushed

Formal definition of PDA, $A = (Q, \Sigma, \Gamma, \delta, q_0, F)$

- Q - Set of states
- Σ - Input alphabet, $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$
- Γ - Stack alphabet, $\Gamma_\varepsilon = \Gamma \cup \{\varepsilon\}$
- $\delta: Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \rightarrow P(Q \times \Gamma_\varepsilon)$
- q_0 - Starting state ($q_0 \in Q$)
- F - Set of accepting sets ($F \subseteq Q$)

5. Equivalence of CFG and PDA

Thm Language is context-free iff some PDA recognizes it.

Proof

CFG ↔ PDA

- CFG → PDA: non-terminals (left-most) → stack
- PDA → CFG: simplify / atomize PDA (single op, 1 start / accept) → CFG (add terminal for state trans that don't touch stack)