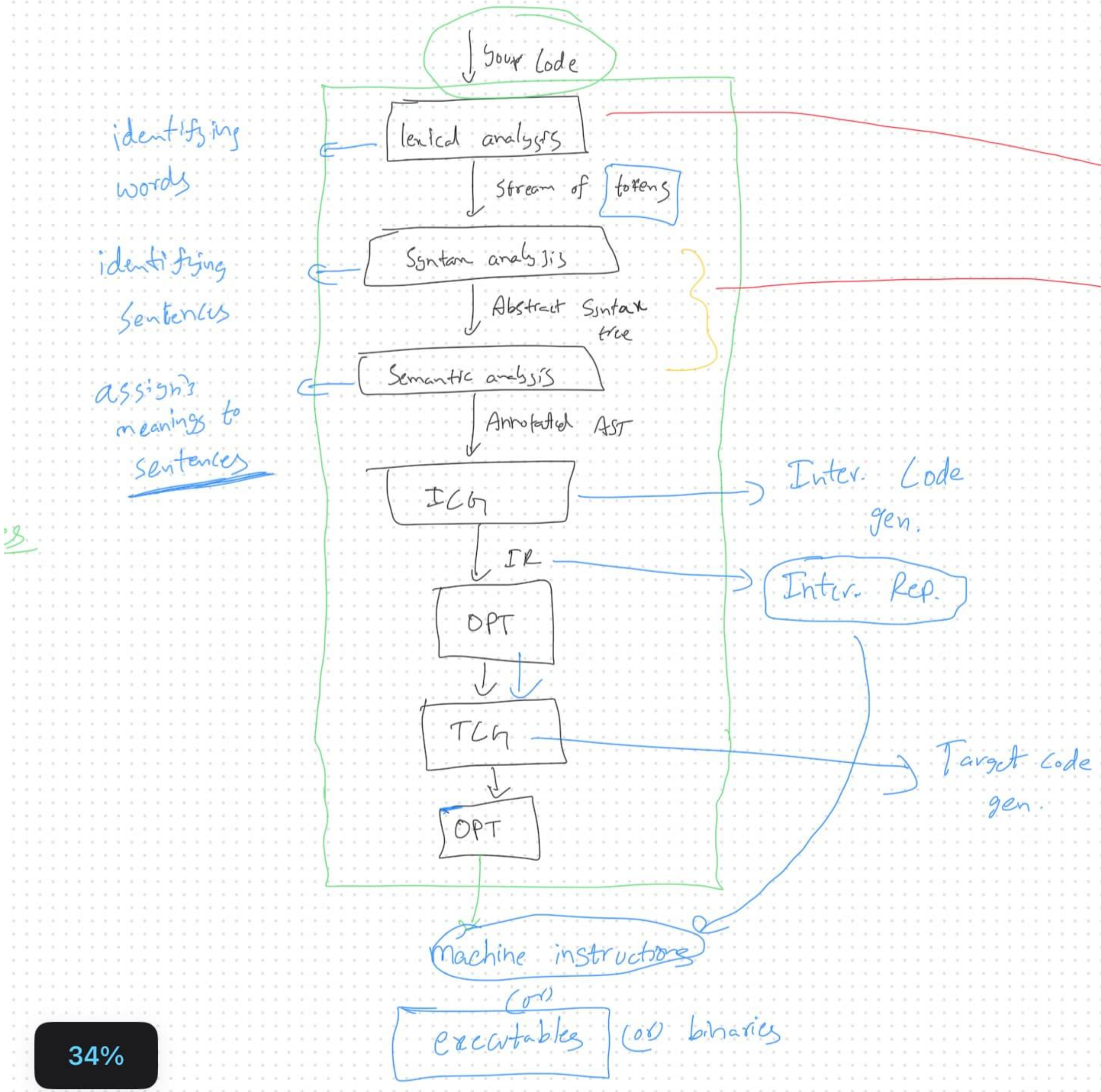
Compiler Design

what is a Compiler?

Prog.c	Prog.Py	Prog.java
gcc Prog.c	Python Prog.Py	javac Prog.java → Prog.class
./a.out		java prog

code/ Program → Compiler → give machine instructions
(or)
Executables (or) Binaries
(or)
bytecode / interpreted.



Theory of Computation

Regular languages

Context free grammars

Turing machines

Theory of Computation → we first need to define "Language" Mathematically

and

Compiler design → understanding language specific code/program → and converting it into machine/CPU instructions

(or) basically an executable

Computers in very early ages → before 1940's.

→ no processers like we have now.

→ used basic transistors to join them

→ instructions like 8 (or) 16 bit are passed Manually by flipping switches (binary)

→ Eventually, programmers recognized the need for making Programming easy, efficient. → Programming languages are then born.

→ First ever Programming Language was FORTRAN I.

→ Now that we have a language and a Compiler, building up became easy.

→ Programmers can use this itself to write further versions of its compiler, or write an OS.

OS's aren't yet well developed till then because it wasn't easy to write an entire OS in Machine instructions.

41%

Theory of Computation

We need to Mathematically, Formally define a "Language" on a paper first before we go write compilers and programs first.

Alphabet :- Σ) It is a finite set of characters

ex :- $\Sigma_1 = \{0,1\}$, $\Sigma_2 = \{a, b, c\}$

String :- A FINITE length string of characters made from a given Σ.

ex :- for a $\Sigma = \{a, b\}$, $x =$ "abba", $y =$ "bbbbbb" are strings.

Empty string :- (ϵ) → It is a string of length '0'. it is empty.

Concatenation of strings

x, y are strings made from Σ

Concatenating x, y is basically joining them

xy.

if $x =$ "ab", $y =$ "ba" → $xy =$ "abba"

if $x =$ "a", $y =$ "b" → $xy =$ "ab"

if $x =$ "ba", $y = \epsilon$ → $xy =$ "ba"

empty string, we saw above

39%

Universal set (U) (or) Σ^* over given Σ:

$\Sigma^* \rightarrow$ it is the set of all the possible strings we can generate from a given alphabet set Σ. including 'ϵ'

$\rightarrow$ if $\Sigma = \{a\}$

$\Sigma^* = \{\epsilon, a, aa, \ldots\ldots\}$

$\Sigma^* \rightarrow$ Set of all possible strings of all finite lengths.

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

$\Sigma^* = \{\overbrace{\epsilon}^{0}, \overbrace{a, b}^{1}, \overbrace{aa, ab, ba, bb}^{2}, \overbrace{aaa, aab, aba, abb, baa, bab, bba, bbb}^{3}, \overbrace{\ldots}^{4}, \overbrace{\ldots}^{5} \ldots\}$

$\rightarrow$ if $\Sigma = \{0, 1\}$

$\Sigma^* = \{\epsilon, \overbrace{0, 1}^{1}, \overbrace{00, 01, 10, 11}^{}, \overbrace{\quad}^{3}, \ldots\ldots\}$

$\hookrightarrow$ Set of all strings possible using Σ.

$\rightarrow$ Σ^* is always an infinite set.

$\hookrightarrow$ Countable infinite

or U (universal set)

Language :- Any subset of Σ^* we will call a language.

→ Do not Panic. This doesnot look anything like a normal language yet but stay with me.

ex :- $\Sigma = \{0,1\}$ $\Sigma^* = \{\epsilon, 0, 1, 01, 10, \ldots\}$

$L_1 = \{\text{"10"}, \text{"01"}\}$, $L_2 = \{\text{"0"}, \text{"000"}, \text{"00000"}, \ldots\}$

→ This is the basic definition of a Language, and again at very basic level, a Language 'L' is just a set, which is a subset of U

$L \subseteq \Sigma^*$ (or) U , for a L constructed on a given Σ.

Computational Problem :-

given a Σ, $L \subseteq \Sigma^*$, $x \in \Sigma^*$

L ↳ a Language x ↳ a String

decide whether $x \in L$ or not.

→ This might look very simple now but please stay

41%

decide whether $x \in L$ or not.

→ This might look very simple now but please stay

- if L is Finite Subset → Trivial
- if L is an Infinite Subset of Σ^* – not so simple.

ex:- $\Sigma = \{0,1\}$

$L_1 = \{\epsilon, 0, 11, 110, 1001, 1100, 1111, ----\}$ → binary representations of all numbers divisible by 3

$\neq \phi$

→ So a finite description must be given for any Computable Language. → in order to build machines to solve CP for that language.

Formal, Mathematical.

→ The language can be an infinite set but describing should be possible finitely.

→ More stronger versions of computability and above statement we will see in Turing Machines part.

$$L = \text{"}\{x \mid x \in \Sigma^*, x \bmod 3 \equiv 0\}\text{"}$$

(T) → another finite set of symbols we use to describe languages formally

Machine:-

$x \in \Sigma^* \to \boxed{L} \to Y/N$

$x \in L$ (or) not

Every computable language $\Longleftrightarrow$ A machine exists

$x \to \boxed{L} \to Y/N$

For every language that we can Finitely describe $\Longleftrightarrow$ there exists a machine capable of solving the computational problems for that Language.

language must be computable

Language's Finite description $\Longleftrightarrow$ Machine

Set of all Subsets of Σ^*

$\uparrow$

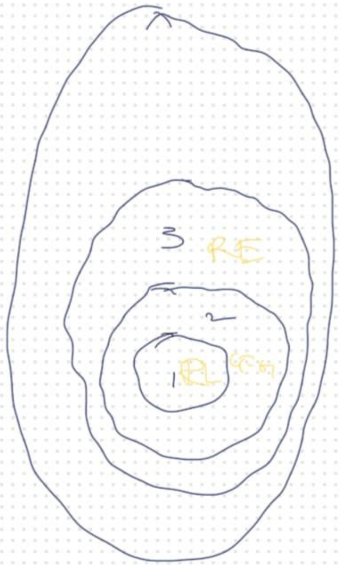
$P(\Sigma^*)$

Types of Languages we will deal with

Machine Models

1) Regular languages $\longleftrightarrow$ DFA, NFA, ε-NFA
2) Context free grammars $\longleftrightarrow$ PDA
3) Recursively enumerable Grammars $\longleftrightarrow$ Turing Machines.

$\downarrow$ Power Incs.



Chomsky hierarchy of Languages.

→ Concatenation of Strings

→ Concatenation of Languages

Σ, $\quad L_1 \subseteq \Sigma^*$, $L_2 \subseteq \Sigma^*$

where xy is Concatenation of Strings x, y

$L_1 \cdot L_2$

$L_1 L_2 = \{ xy \mid x \in L_1, y \in L_2 \}$

$A, B \qquad A \times B = \{ (a,b) \mid a \in A, b \in B \}$

ex:- $A = \{0,1\}$, $B = \{a,b\}$

$A \times B = \{(0,a), (0,b), (1,a), (1,b)\}$

ex:-

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

$L_1 = \{\text{"ab"}, \text{"ba"}\}$, $L_2 = \{\epsilon, \text{"a"}, \text{"b"}\}$

$L_1 \cdot L_2 = \{\text{"ab"}, \text{"aba"}, \text{"abb"}, \text{"ba"}, \text{"baa"}, \text{"bab"}\}$

25%

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

$L_1 = \{"ab", "ba"\}$, $L_2 = \{\epsilon, "a", "b"\}$

$L_1 L_2 = \{"ab", "aba", "abb", "ba", "baa", "bab"\}$

$L_1 = \{"ab", "abb"\}$, $L_2 = \{"b", \epsilon, "a"\}$

$m * n$

L^*

def

$L_1 L_2 = \{"abb", "ab", "aba", "abbb", \cancel{"abb"}, "abba"\}$

$L^0 = \{\varepsilon\}$

L_1

$L^2 = L L$

$L^3 = L \cdot L \cdot L = L^2 \cdot L = L \cdot L^2 = L^0 \cdot L^3$

$L^K = \underbrace{L \cdot L \cdot \_\_\_\_ \cdots L}_{K\text{- times.}} = L^{K-1} L = L \cdot L^{K-1}$

$\boxed{L^* = \bigcup_{K \geq 0} L^K}$ $= L^0 \cup L^1 \cup L^2 \ldots\ldots$

asterate of L

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

$\Sigma \subset \Sigma^*$

finite set

all chars are basically strings of length 1.

$\Sigma^0 = \{\varepsilon\}$ $l=0$

$\Sigma^1 = \{a,b\}$ $l=1$

$\Sigma^2 = \{a,b\}\cdot\{a,b\}$

$= \{aa, ab, ba, bb\}$

↳ set of all length $=2$ strings.

$\vdots$

$\Sigma^k =$ set of all strings of length k.

$L^* \longleftrightarrow \Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \dots$

def $=$ set of strings of all possible lengths.

ToC

Σ, ε, Σ^*, Language $\subseteq \Sigma^*$,

Comp Problem

$x, L \rightarrow x \in L$ or not,

Machine $x \rightarrow \boxed{L} \rightarrow Y/N$.

Result :-

There are ∞ elements in Σ^*.

$\Rightarrow$ ∞ subsets of Σ^*

$\Rightarrow$ ∞ languages exist.

$\rightarrow$ Not all languages have finite descriptions.

$\hookrightarrow$ not all languages have machines to solve comp problems for that.

$x, L \rightarrow x \in L$ or not.

Yes/no.

decision problem.

Functional $\quad f(x) \rightarrow y$

$\rightarrow$ All functional problems can be solved by using small decision problems.

ex: $f(x) = \sqrt{x}$

$f(10) = \sqrt{10}$

DP: $\quad x^2 > 10$ or not

1 $\rightarrow$ $1^2 > 10$ or not NO

2 NO

3 NO

3 $\leftarrow$ 4 YES

Every language for us to build a machine $\Leftrightarrow$ Finite description.

$\downarrow$

$T \rightarrow$ another alphabet set, we use to give language descriptions

How many languages are possible?

$\Sigma \rightarrow \Sigma^* \rightarrow \infty$

$\hookrightarrow$ Countable ∞.

Power set.

$P(A)$ = Set of all subsets of A.

no. of languages $\rightarrow$ no. of subsets of Σ^*.

$2^{|\Sigma^*|}$

$\downarrow$

Size of $P(A)$ where A is Countable ∞.

$\downarrow$

Uncountable ∞ > Countable ∞

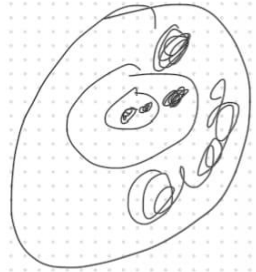
no. of possible lang. = Uncountable ∞

$>$

no. of language descriptions we can give = T^* = Countable ∞

Languages are Subsets of Σ^*.

$\rightarrow$ There exists some languages with no machines.



no. of possible languages > no. of possible lang. descriptions > no. of Computable languages.

$\Downarrow$

there exists languages with no finite descriptions.

$\Downarrow$

no machines for solving CP for those languages