CSE443
Compilers

Dr. Carl Alphonce
alphonce@buffalo.edu
343 Davis Hall

Announcements

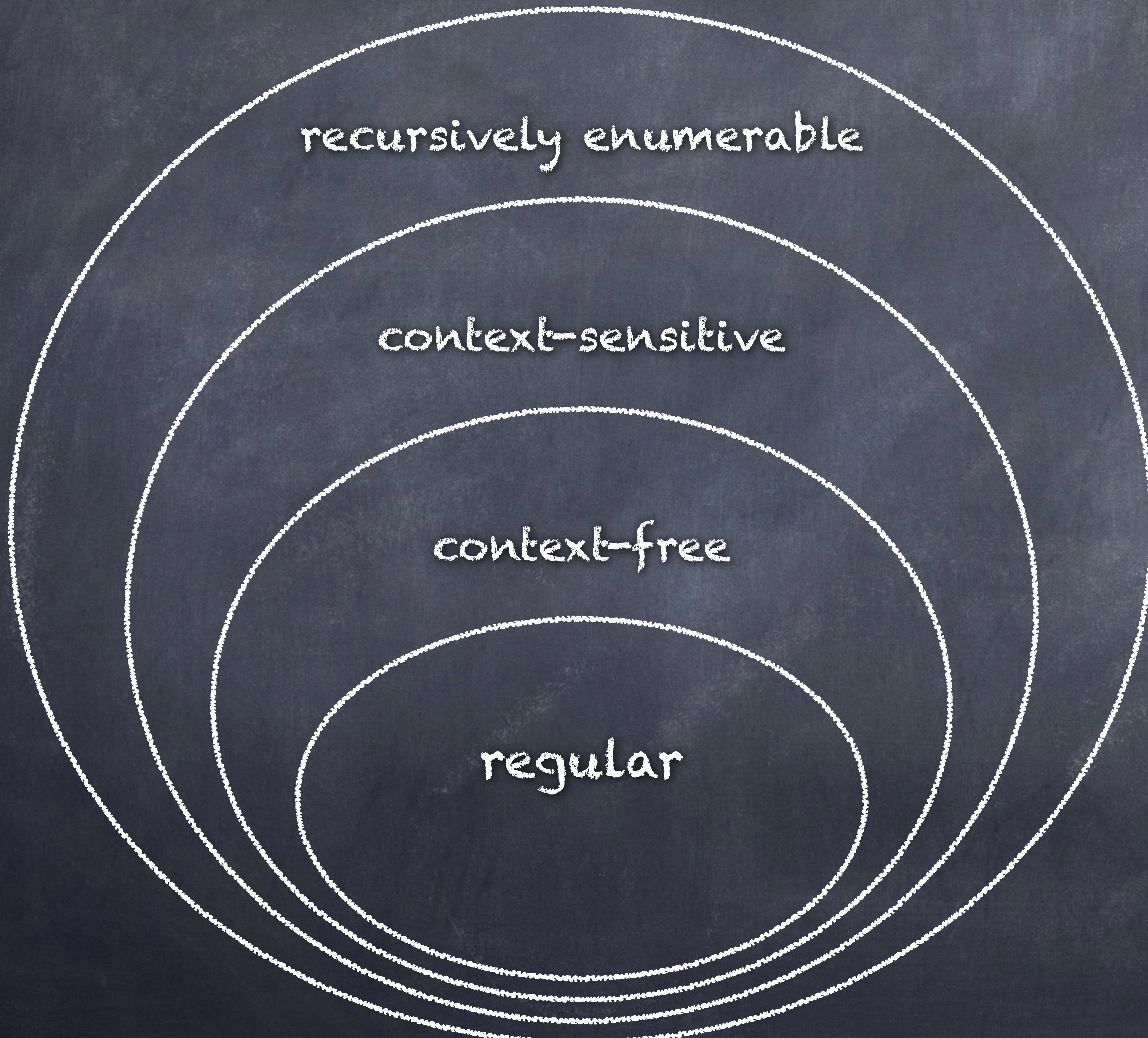
- Project document posted on website
  - Only worry about page 1 - 5 for now
- Team formation
  - We can take a few minutes to do it now
  - Make private Piazza post with UBIT username and corresponding GitHub username.

Syntax and semantics

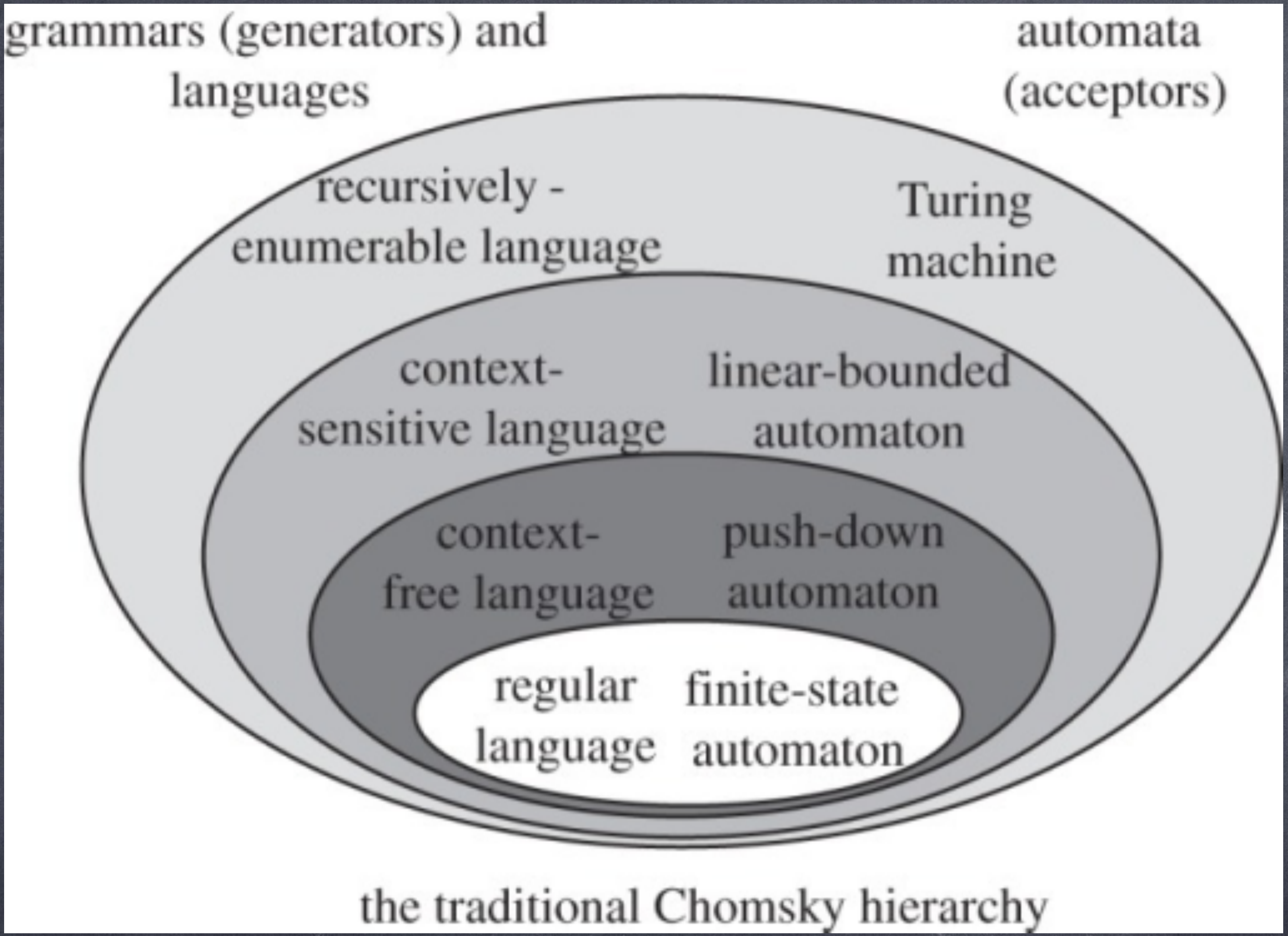
- Syntax: program structure
- Semantics: program meaning
- Semantics are determined (in part) by program structure.

Languages: the Chomsky hierarchy

"On Certain Formal Properties of Grammars" published 1959

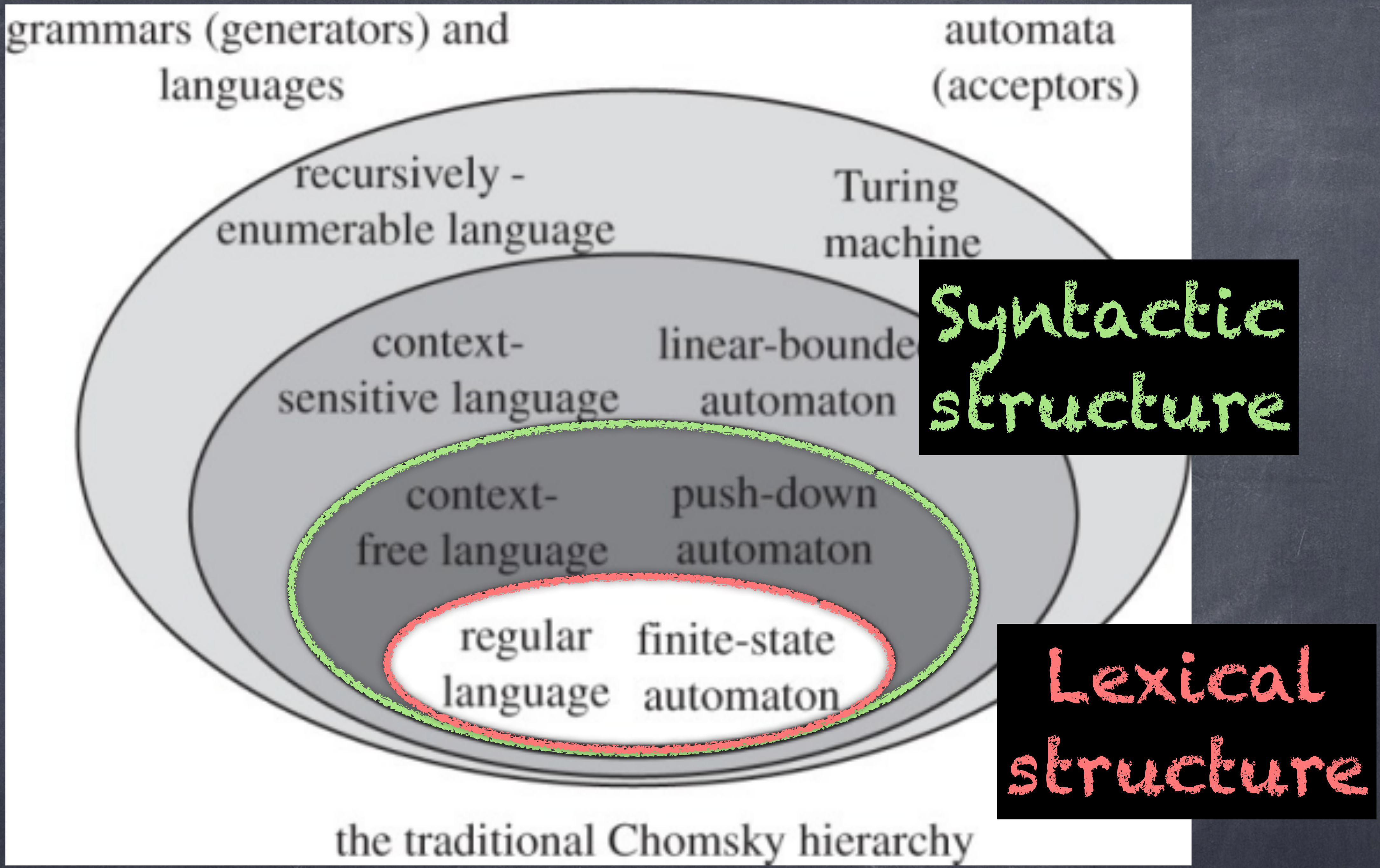


https://upload.wikimedia.org/wikipedia/commons/8/86/Noam_chomsky.jpg



SOURCE: https://openi.nlm.nih.gov/detailedresult.php?img=PMC3367694_rstb20120103-g2&req=4

AUTHORS: Fitch WT, Friederici AD - Philos. Trans. R. Soc. Lond., B, Biol. Sci. (2012)



SOURCE: https://openi.nlm.nih.gov/detailedresult.php?img=PMC3367694_rstb20120103-g2&req=4

AUTHORS: Fitch WT, Friederici AD - Philos. Trans. R. Soc. Lond., B, Biol. Sci. (2012)

Phases of a compiler

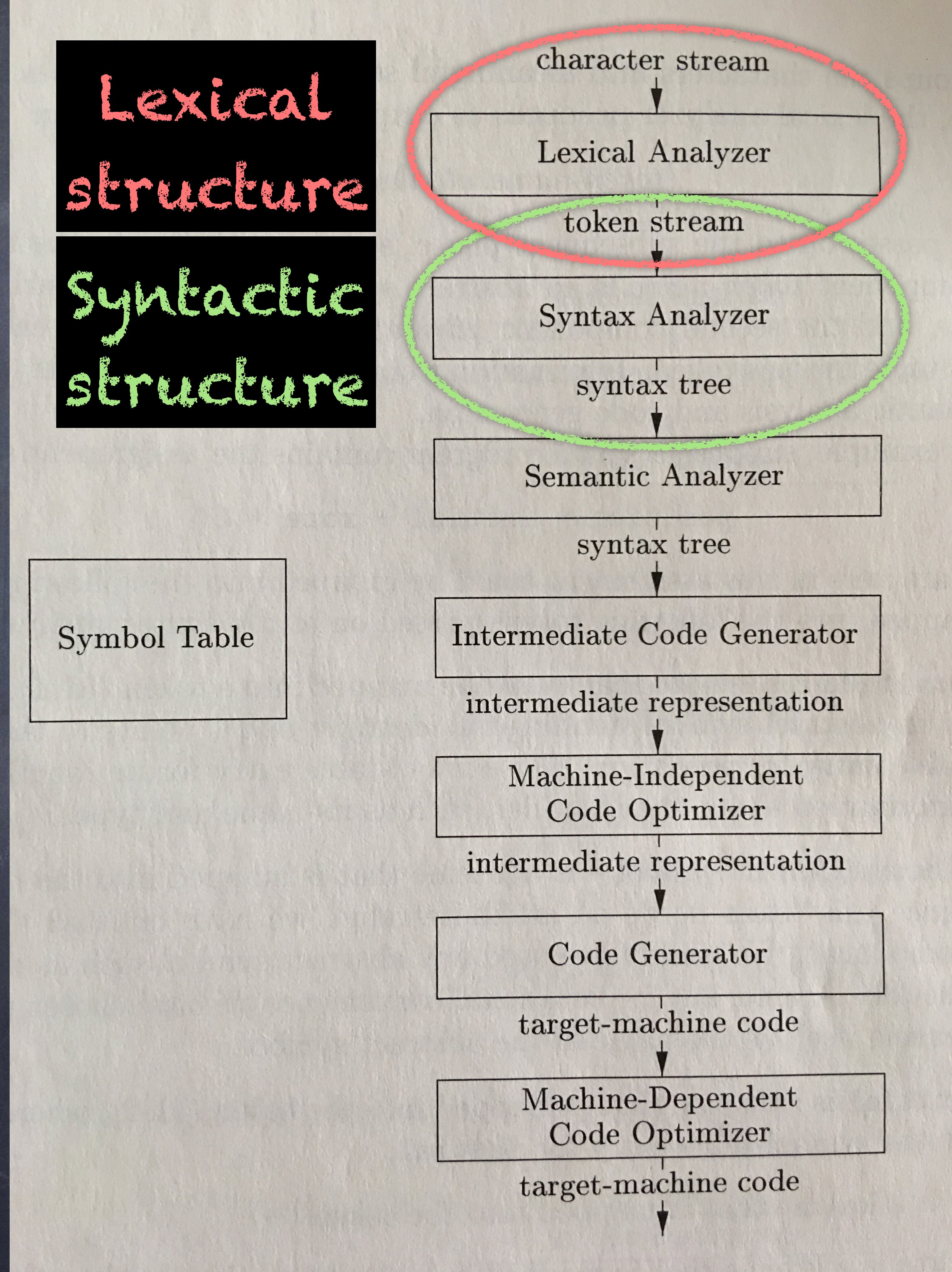


Figure 1.6, page 5 of text

Phases of a compiler

Lexical structure

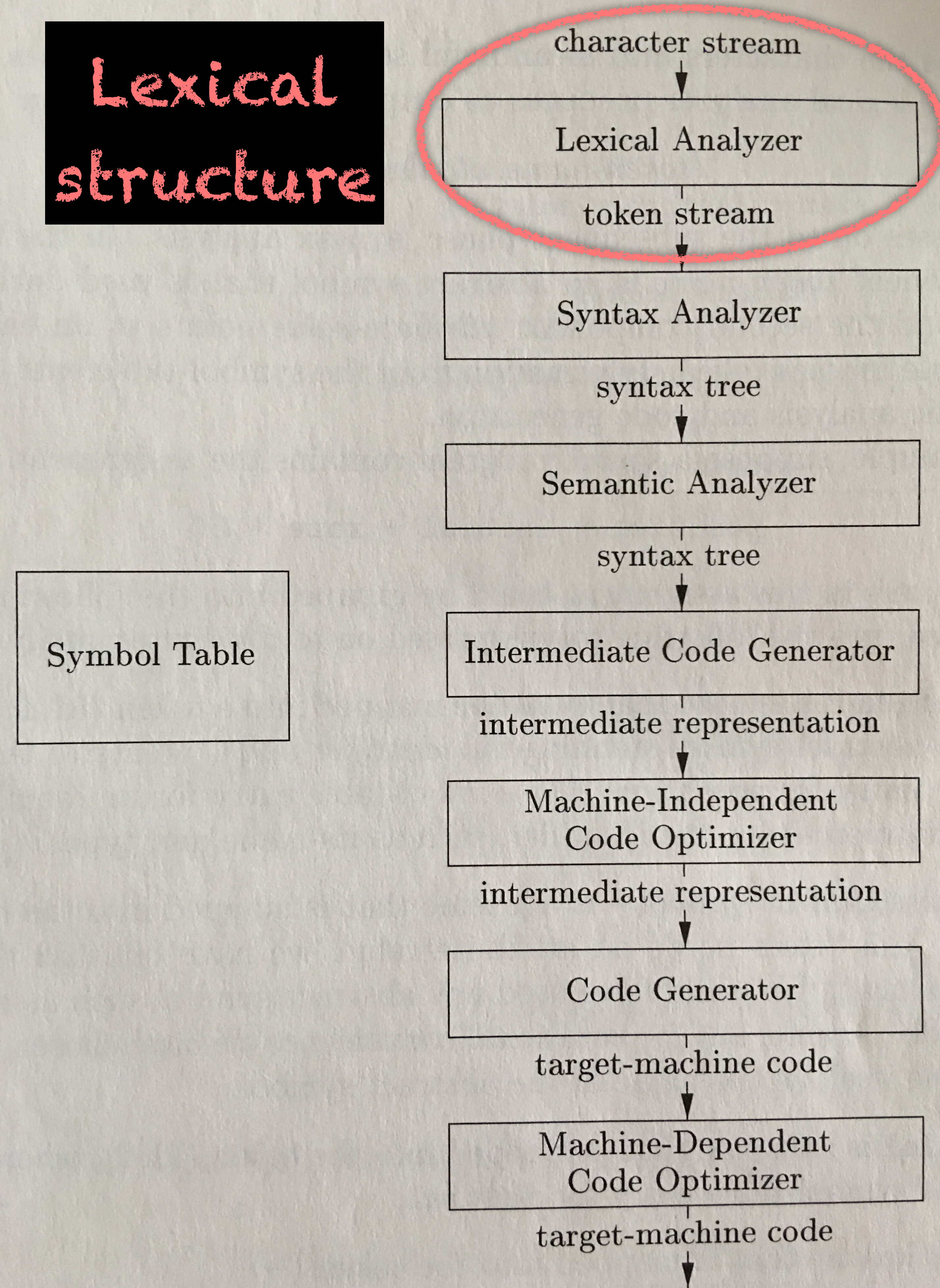


Figure 1.6, page 5 of text

Lexical Structure

```
int main(){
```

Lexical Structure

int main(){

character stream

```
i n t □ m a i n ( ) {
```

Lexical Structure

int main(){

character stream -> token stream

i n t □ m a i n ( ) {   id("int") id("main") LPAR RPAR LBRACE

Lexical Structure

tokens

- keywords (e.g. static, for, while, struct)
- operators (e.g. <, >, <=, =, ==, +, -, &, .)
- identifiers (e.g. foo, bar, sum, mystery)
- literals (e.g. -17, 34.52E-45, true, 'e', "Serenity")
- punctuation (e.g. { , } , (,) , ;)

Describing lexical structure

- We need some formal way of describing the lexical structure of a language.

meta vs object language

- object language: the language we are describing
- meta language: the language we use to describe the object language

meta vs object language

- How do we distinguish between the two?

meta vs object language

- use quotes (meta vs ‘object’)
  - punctuation (e.g. ‘{’ , ‘}’ , ‘(’ , ‘)’ , ‘;’)
- use font or font property (meta vs object)
  - punctuation (e.g. { , } , (,) , ;)

Languages & grammars

- Formally, a language is a set of strings over some alphabet
- Ex. {00, 01, 10, 11} is the set of all strings of length 2 over the alphabet {0, 1}
- Ex. {00, 11} is the set of all even parity strings of length 2 over the alphabet {0, 1}

Languages & grammars

- Formally, a **grammar** is defined by 4 items:
  1. N, a set of non-terminals
  2. Σ, a set of terminals
  3. P, a set of productions
  4. S, a start symbol
- $G = (N, \Sigma, P, S)$

Lexical analysis: a bird's eye view

{ for, while, x, factorial, ... }

language: a set of strings

G = (N, Σ, P, S)

grammar: rules for generating language

regular expression

regex: a form of grammar

finite automaton

a machine for language

C program

generated by FLEX

Languages & grammars

Formally, a grammar $G = (N, \Sigma, P, S)$ is defined by 4 items:

1. N, a set of non-terminals

   $N = \{ X, Y \}$

2. Σ, a set of terminals (alphabet)

   $\Sigma = \{ a , b \}$ <- for example

   $N \cap \Sigma = \{\}$ <- general grammar constraints

3. P, a set of productions of the form (right linear)

   X -> aY

   Y -> bX

   Y -> a <- a right linear grammar describing a regular language

   X -> ε

   $X \in N$, $Y \in N$, $a \in \Sigma$, $b \in \Sigma$, ε denotes the empty string

4. S, a start symbol

   $S = Y$

   $S \in N$

languages & grammars

[...] a regular grammar is a grammar that is right-regular or left-regular:

- all production rules have at most one non-terminal symbol
- that symbol is either always at the end or always at the start of the rule's right hand side.

https://en.wikipedia.org/wiki/Regular_grammar

Languages & grammars

Given a string αA, where $\alpha \in \Sigma^*$ and $A \in N$, and a production $A \rightarrow \beta \in P$

we write $\alpha A \Rightarrow \alpha\beta$ to indicate that αA **derives** $\alpha\beta$ in one step.

$\Rightarrow^k$ and $\Rightarrow^*$ can be used to indicate k or arbitrarily many derivation steps, respectively.