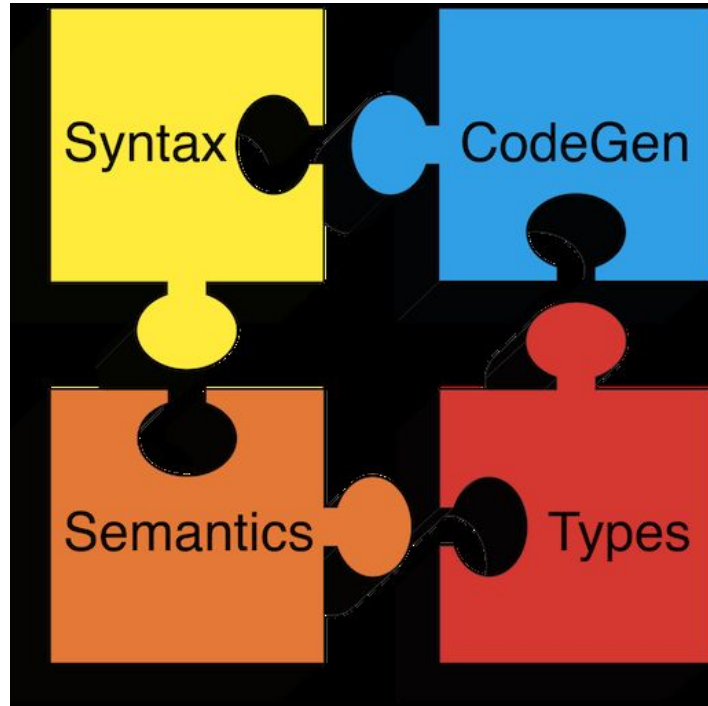


# Programming Languages.

## Part 2. Compilers



# Formal Languages

---

**Def.** Let  $\Sigma$  be a set of characters (an *alphabet*).

A *language over  $\Sigma$*  is a set of strings of characters  
drawn from  $\Sigma$

# Formal Languages

- **Alphabet** =  
English  
characters

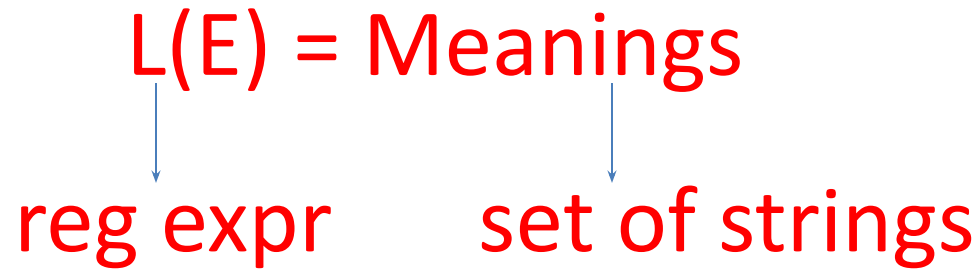
- **Language** =  
English  
sentences

- **Alphabet** = **ASCII**

- **Language** = **C**  
programs

# Meaning function

Meaning function  $L$  maps syntax to semantics



# Meaning function

$$\varepsilon = \{ \text{""} \}$$

$$'c' = \{ \text{"c"} \}$$

$$A + B = \{ a \mid a \in A \} \cup \{ b \mid b \in B \}$$

$$AB = \{ ab \mid a \in A, b \in B \}$$

$$A^* = \bigcup_{i \geq 0} A^i$$

$L : \text{Exp} \rightarrow \text{Sets Strings}$

# Meaning function

$$L(\epsilon) = \{ \text{""} \}$$

$$L('c') = \{ \text{"c"} \}$$

$$L(A + B) = L(A) \cup L(B)$$

$$L(AB) = \{ ab \mid a \text{ from } L(A) \wedge b \text{ from } L(B) \}$$

$$L(A^*) = \cup L(A^i)$$

# Meaning function

- Why use a meaning function?
  - Makes clear what is **syntax**, what is **semantics**.
  - Allows us to consider notation as a separate issue
  - Because expressions and meanings are not 1-1



# Meaning function

0    1    42    107

I    IV    X    XL

# Syntax and Semantic

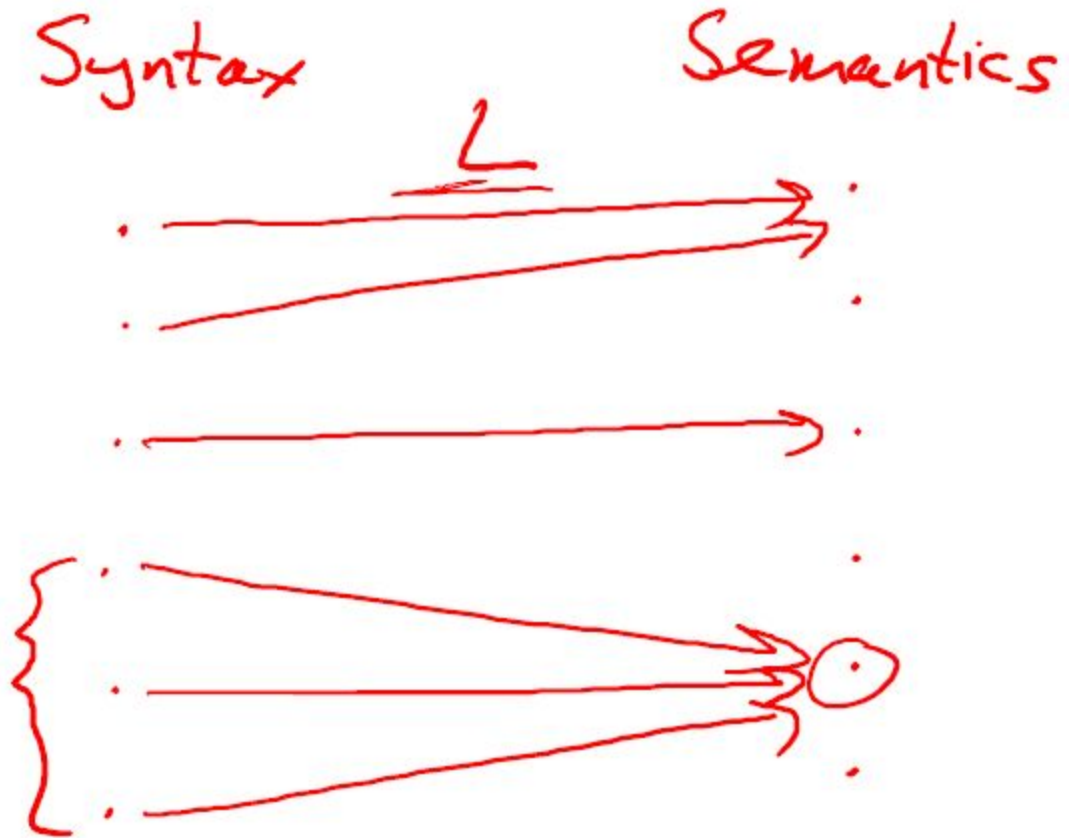
$0^*$

$0 + 0^*$

$E + 00^*$

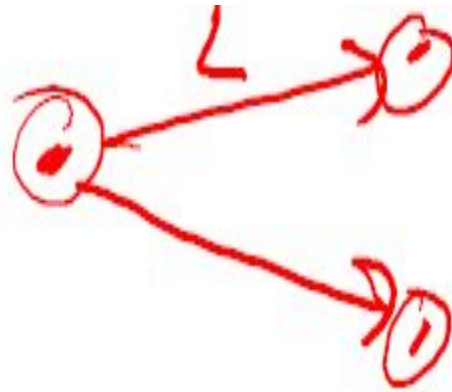
$E + 0 + 0^*$

...



# Syntax and Semantic

- Meaning is many to one
  - Never one to many!



Thank you for your  
attention