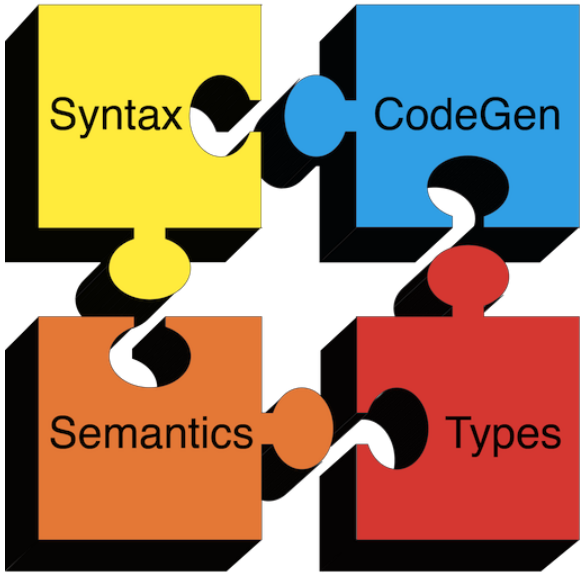


Compilers

Formal Languages

Formal Languages

Def. Let Σ be a set of characters (an *alphabet*).

A *language over* Σ is a set of strings of characters drawn from Σ

Formal Languages

- Alphabet = English characters
- Language = English sentences

- Alphabet = ASCII
- Language = C programs

Meaning function *L* maps syntax to semantics

$L(e) = M$

e: reg exp

M: set of strings

Formal Languages

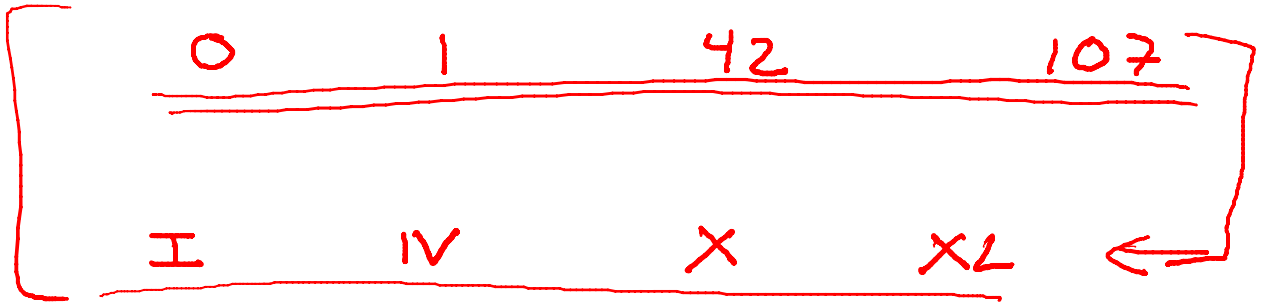
$L : Exp \rightarrow Sets\,Strings$

$$
\begin{aligned}
L(\varepsilon) &= \{ "" \} \\
L('c') &= \{ "c" \} \\
L(A + B) &= L(A) \cup L(B) \\
L(A\ B) &= \{ ab \mid a \in L(A) \wedge b \in L(B) \} \\
L(A^*) &= \bigcup_{i \geq 0} L(A^i)
\end{aligned}
$$

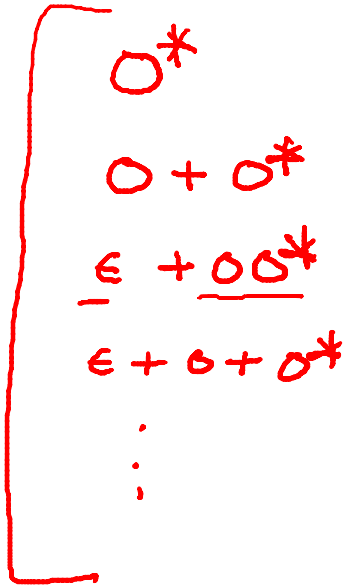
Formal Languages

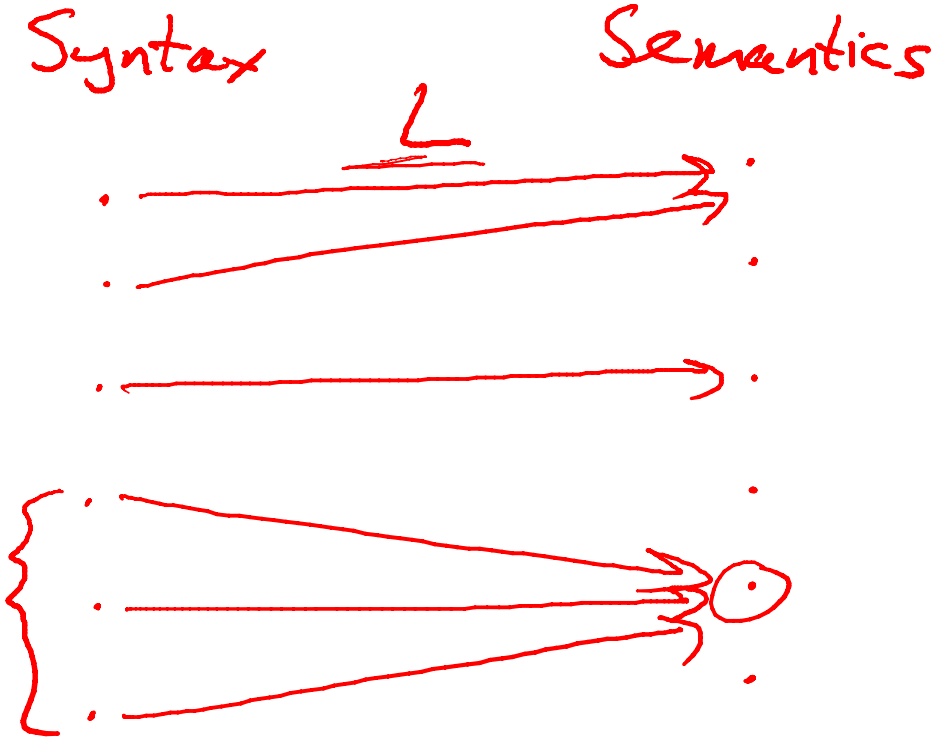
- Why use a meaning function?
  - Makes clear what is syntax, what is semantics.
  - Allows us to consider notation as a separate issue
  - Because expressions and meanings are not 1-1

Formal Languages



Formal Languages





Formal Languages

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