Parsing Expression Grammars

Niccolò Piazzesi

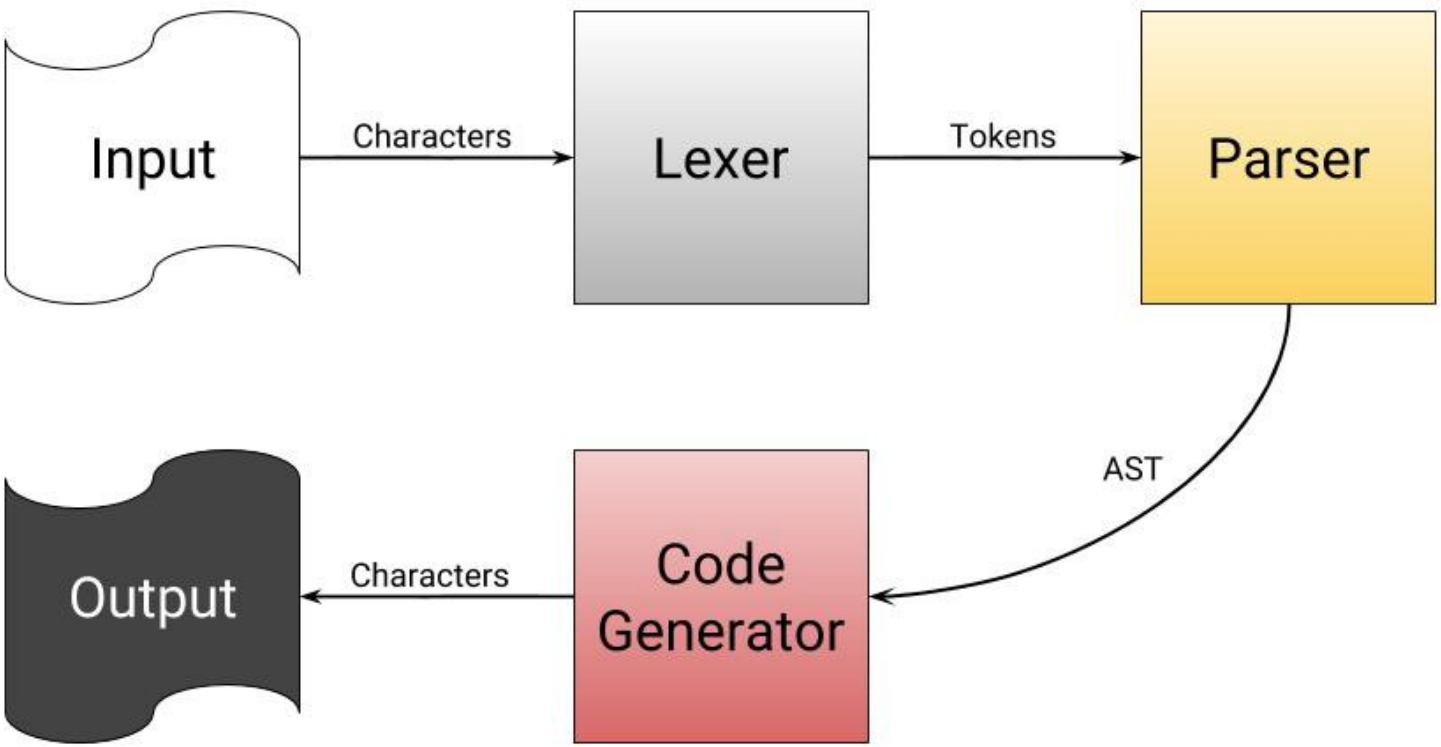
Università degli Studi di Pisa
Anno Accademico 2020-21

January 9, 2021

Parsing

What's a parser?

Part of the compiler. Checks the stream of tokens produced by the *lexer* for syntactical errors and produces an IR representation (usually an abstract syntax tree) of the source that is used in later steps to generate machine code.



Context free grammars

We need a model to describe a language syntax.

Context free grammars

We need a model to describe a language syntax.

Classical problem, solved by using formal language theory.

Context free grammars

We need a model to describe a language syntax.

Classical problem, solved by using formal language theory.

Syntax described by a **Context Free Grammar**.

CFG: definition

Context free grammar defined as:

$$\mathbf{G = (V, \Sigma, P, S)}$$

- V = variables
- Σ = terminals
- P = productions (or rules)
- S = start variables

Productions are in the form A $\rightarrow \alpha$, where A$\in$ V, $\alpha \in (V \cup \Sigma)$

S $\Rightarrow$ A | bb

A $\Rightarrow$ B | b

B $\Rightarrow$ S | a

Given the rules of grammar G, we want to find a sequence of productions that generates a target expression (**derivation**).

Parsing is the process of discovering such sequence.

$S \rightarrow \gamma_1 \rightarrow \gamma_2 \rightarrow \cdots \rightarrow \gamma_n \rightarrow \textit{expression}$

Leftmost derivation: at each step expand the leftmost non-terminal symbol

Rightmost derivation: at each step expand the rightmost non-terminal symbol

1	*Expr*	→	*Expr* + *Term*
2		|	*Expr* - *Term*
3		|	*Term*
4	*Term*	→	*Term* × *Factor*
5		|	*Term* ÷ *Factor*
6		|	*Factor*
7	*Factor*	→	(*Expr*)
8		|	num
9		|	ident

Figure: classic grammar for arithmetic expressions

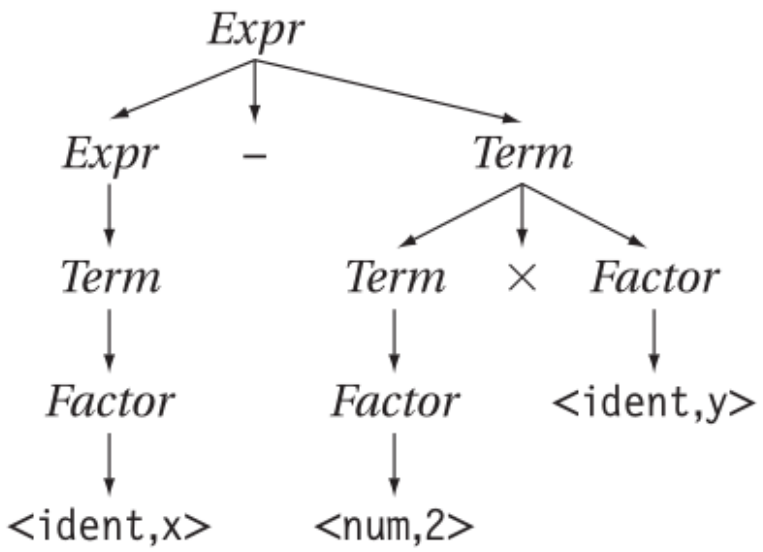


Figure: parse tree for the expression x-2*y

Formal definiton of PEG

A concrete example: *pgen*