Sintaxe

* Forma
* Alfabeto
* Sentenças
* Regras (gramática)
* Tokens
  + Identificadores
  + Operadores
  + ...

Reconhecedores

* Linguagem (L) sob um ∑ (alfabeto)
* Reconhece sentenças corretas
* Análise sintática → reconhecedor

Geradores

* Gera sentenças

BNF

* Formal para reconhecimento
* Gramática
* Regras

Símbolos não-terminais

* São derivados

Símbolos terminais

* Finais
* Pertencem ao ∑

<não-terminais>

:= → terminal

Derivação ∑={a, b}

<A> → a | <A> a | a <B>

<B> → b | b <B>

short → símbolo inicial

Exemplo:

Instrução:

<program> → Begin <ident\_list> end

<ident\_list> → <ident>; | <ident> ; <ident\_list>

<ident> → <var> = <expressão>

<var> → A | B | C

<expressão> → <var> + <var> | <var> - <var> | <var>

Expressão:

begin A=B+C; B=C; end

Avaliação:

<program>

begin <ident\_list> end

begin <ident>; <ident\_list> end

begin <var> = <expressão>; <ident\_list> end

begin A = <expressão>; <ident\_list> end

begin A = <var> + <var>; <ident\_list> end

begin A = B + <var>; <ident\_list> end

begin A = B + C; <ident\_list> end

begin A = B + C; <ident>; end

begin A = B + C; <var> = <expressão>; end

begin A = B + C; B = <expressão>; end

begin A = B + C; B = <var>; end

begin A = B + C; B = C; end

Expressão:

begin A = B \* B; C = C - A; end

Avaliação:

<program>

begin <ident\_list> end

begin <ident>; <ident\_list> end

begin <var> = <expressão>; <ident\_list> end

begin A = <expressão>; <ident\_list> end

Erro. Não existe expressão.