UNIVERSIDADE EDUARDO MONDLANE

FACULDADE DE ENGENHARIA

DEPARTAMENTO DE ENGENHARIA ELECTROTÉCNICA

COMPILADORES

Introdução ao Analisador Sintáctico

Docentes: Ruben Moisés Manhiça
Cristiliano Maculuve

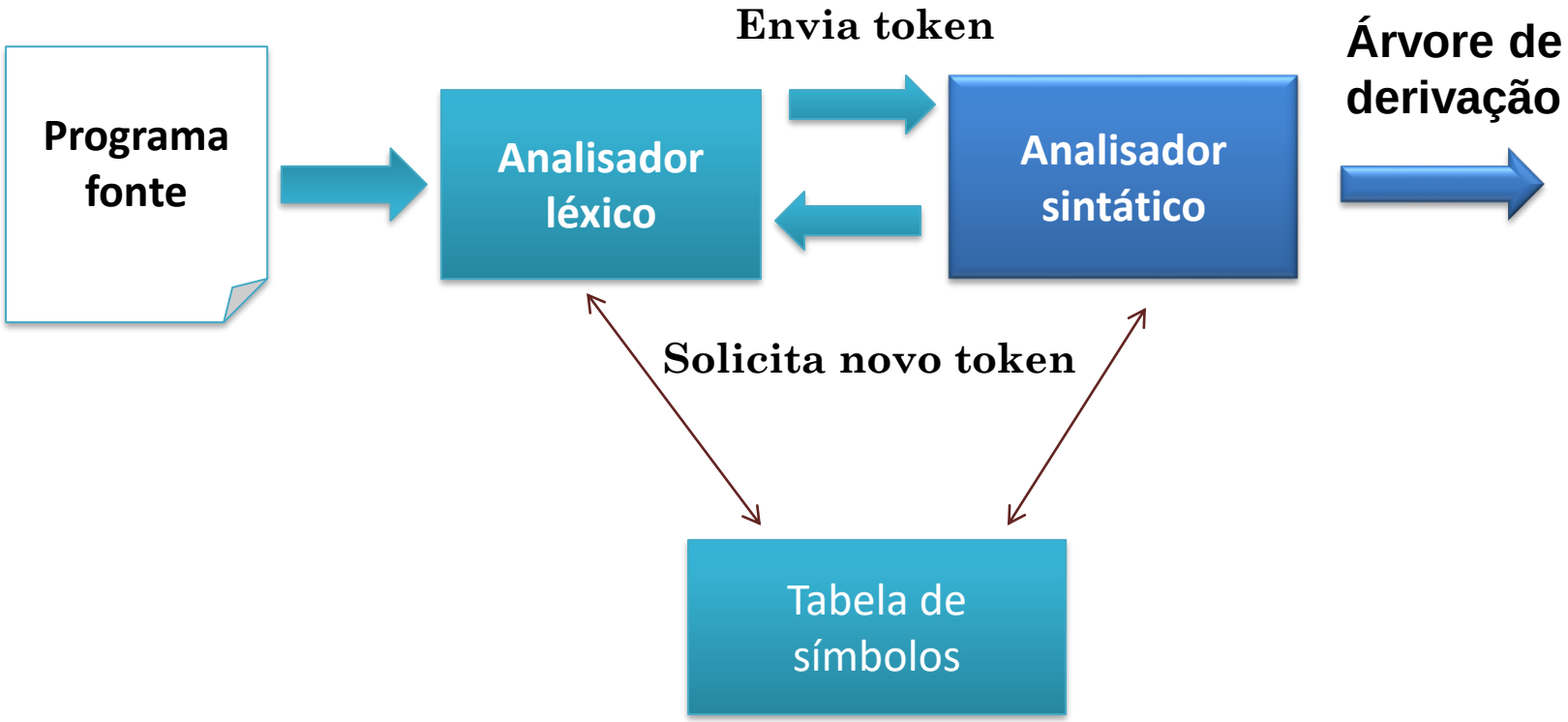
Maputo, 14 de abril de 2023

Conteúdo da Aula

1. Introdução a Analise Sintática
2. Gramáticas Livres de Contexto
3. Árvores de definição

Analise Sintatica

- Analisador sintático (*parser*) é o responsável por verificar se as construções utilizados no programa estão gramaticalmente corretas



Sintaxe

- Define a forma e estrutura de uma linguagem
- Símbolos, palavras, frases e sentenças (estruturas)
- Principal formalismo:
  - Gramáticas Livres de Contexto e Expressões Regulares
  - Notação mais utilizada: BNF (Backus-Naur Form)

Reconhecimento de uma Linguagem

- Toda linguagem tem de ter regras que descrevem sua estrutura sintática (ou sintaxe)
- A sintaxe pode ser descrita através de uma gramática ou pela notação BNF
- Vantagens de se utilizar uma gramática:
  - Fornece uma especificação sintática precisa e fácil de entender
  - Para certas classes de gramáticas, podemos construir automaticamente um analisador sintático e o gerador automático pode certas ambigüidades sintáticas da LP, difíceis de serem identificadas diretamente pelo projeto do compilador
  - Novas construções que surgem com a evolução da linguagem podem facilmente ser incorporadas a um compilador se este tem sua implementação baseada em descrições gramaticais

Descrição de Uma Linguagem Através de uma Gramática

- Linguagens regulares não são capazes de identificar recursões centrais
  - E = x | “(“ E “)”
- Solução: Uso de gramáticas livres de contextos
- Uma Gramática Livre de Contexto é construída utilizando símbolos terminais e não-terminais, um símbolo de partida e regras de produções, onde:
  - Os **terminais** são os símbolos básicos a partir dos quais as cadeias são formadas. Na fase de análise gramatical os tokens da linguagem representam os símbolos terminais. Ex: **if, then, else, num, id,** etc.

Gramática Livre de Contexto

- Os **não-terminais** as variáveis sintáticas que denotam cadeias de caracteres. Impõem uma estrutura hierárquica que auxilia na análise sintática e influencia a tradução. Ex: *cmd, expr*.
- Numa gramática um não terminal é distinguido como símbolo de partida, e o conjunto que o mesmo denota é a linguagem definida pela linguagem. Ex: *program*
- As produções de uma gramática especificam como os terminais e não-terminais podem se combinar para formas as cadeias da linguagem. Cada produção consiste em um não terminal seguido por uma seta (ou ::=), serguido por uma cadeia de não terminais e terminais

Gramática Livre de Contexto

- Ex:

  expr ::= *expr op expr*
  expr ::= (expr)
  expr ::= - expr
  expr ::= **id**
  op ::= +
  op ::= -
  *op ::= **
  op ::= /
- Simbolos terminais
  - **id** + - * / ( )
- Símbolos não-terminais
  - *expr* e *op* , sendo *expr* o símbolo de partida

Convenções Notacionais

- Símbolos Terminais
  - Letras minúsculas do inicio do alfabeto, tais como *a, b c*
  - Símbolos de operadores, tais como +, -, etc
  - Símbolos de pontuação, tais como parênteses e vírgulas
  - Dígitos 0, 1, ..., 9
  - Cadeias em negritos como **id** ou **if**
- Símbolos não-terminais
  - Letras maiúsculas do início do alfabeto, tais como A, B, C
  - A letra S, quando aparecer é usualmente símbolo de partida
  - Nomes em itálico formados por letras minúsculas, como *expr* ou *cmd*
- A menos que seja explicitamente estabelecido, o lado esquerdo da primeira produção é o símbolo de partida

Gramáticas

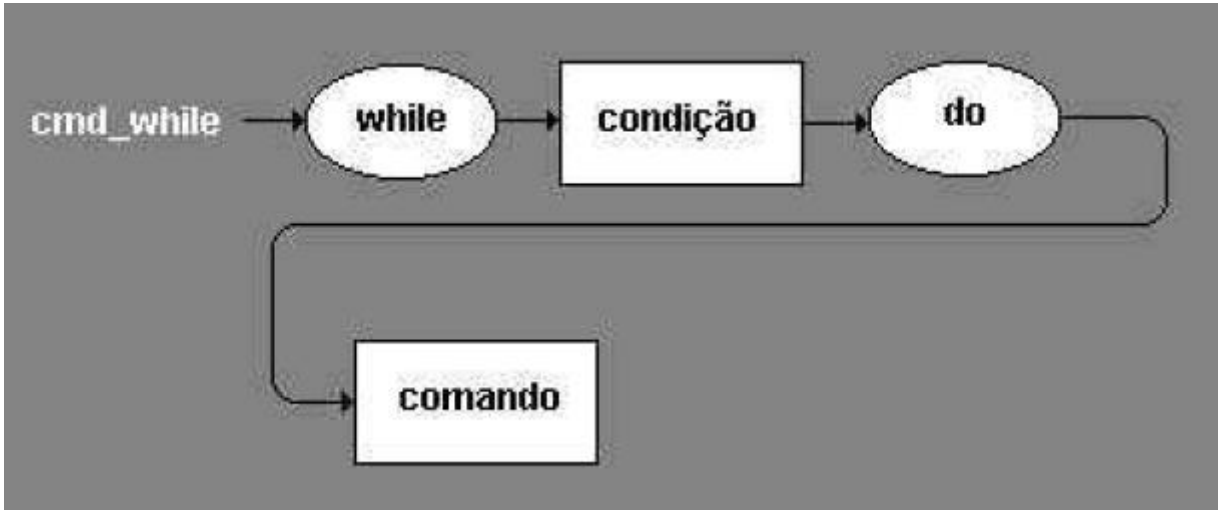
- Produções para o mesmo símbolo não terminal a esquerda podem ser agrupadas utilizando "|". Ex: A::= +|-|...
- Exemplo:

expr ::= *expr op expr*
expr ::= *(expr)*
expr ::= *- expr*
expr ::= **id**
op ::= +
op ::= -
op ::= *
op ::= /

E ::= E A E|(E)|-E| **id**
A ::= +|-|*|/

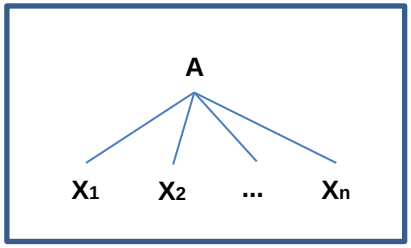
Grafos de Sintaxe

- Grafo direcionado contendo dois tipos de vértices
  - Vértices em elipse para representar os símbolos terminais
  - Vértices retangulares para não terminais



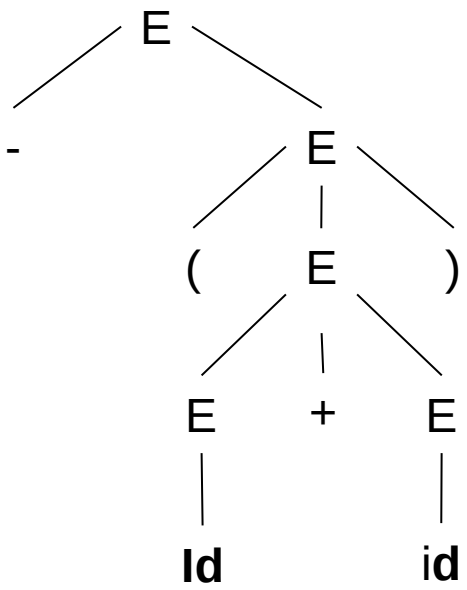
Árvores Gramaticais

- Representação gráfica de uma derivação
- Dá forma explícita a estrutura hierárquica que originou a sentença
- Dada uma GLC, a árvore de derivação é obtida:
  - A raiz da árvore é o símbolo inicial da gramática
  - Os vértices interiores são obrigatoriamente não-terminais. Ex: Se A ::= X1X2...Xn é uma produção da gramática, então A será um vétice interior e X1, X2, ..., Xn serão os filhos (da esquerda para a direita)
  - Símbolos terminais e a palavra vazia são as folhas



Árvores de Derivação

- Exemplo: -(**id** + **id**)



E::=-E

E::=(E)

E::=E+E

E::=**id** E::=**id**

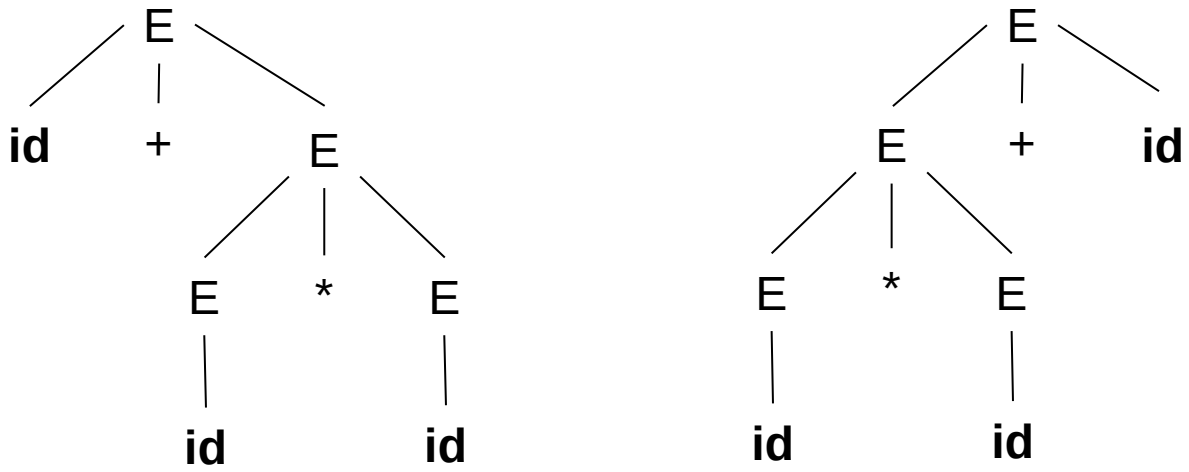
Derivações

- Processo através do qual as regras de produções da gramática são aplicadas para formar uma palavra ou verificar se esta pertence a linguagem
- Símbolo não terminal é substituído pelo lado direito da produção correspondete
- Ex: -(**id** + **id**)

  E => -E => -(E) => -(E+E) => -(**id** + E) => -(**id** + **id**)
- Dois passos:
  - Qual terminal será escolhido para derivar
    - Derivação mais a esquerda
    - Derivação mais a direita
  - Qual regra utilizar

Ambiguidade

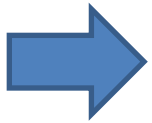
- Se uma gramática possui mais de uma árvore gramatical para uma mesma sentença é dita ambígua
- Parte do significado dos comandos de uma linguagem podem estar especificado em sua estrutura sintática
- Ex: **id** + **id** * **id** possui duas derivações mais a esquerda



Ambiguidade

- Regras de precedência
- Reescrita da gramática

expr ::= *expr op expr*
expr ::= **id**
op ::= +
op ::= -
op ::= *
op ::= /



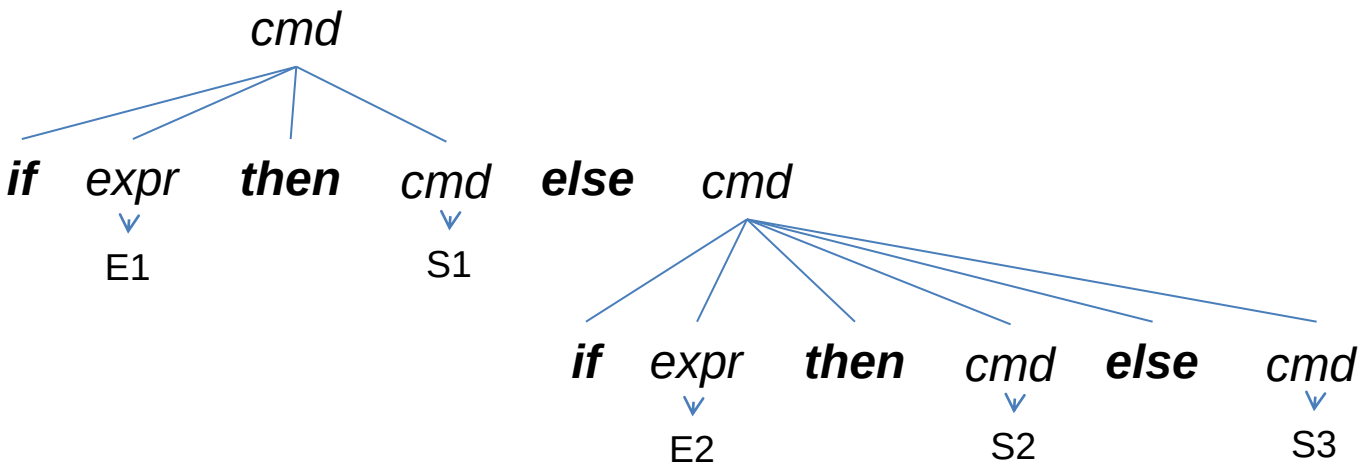
expr ::= *term* | *term op1 term*
term ::= *fator* | *fator op2 fator*
fator ::= **id** | *(expr)*
op1 ::= +
op1 ::= -
op2 ::= *
op2 ::= /

cmd ::= **if** *expr* **then** *cmd*

|**if** *expr* **then** *cmd* **else** *cmd*

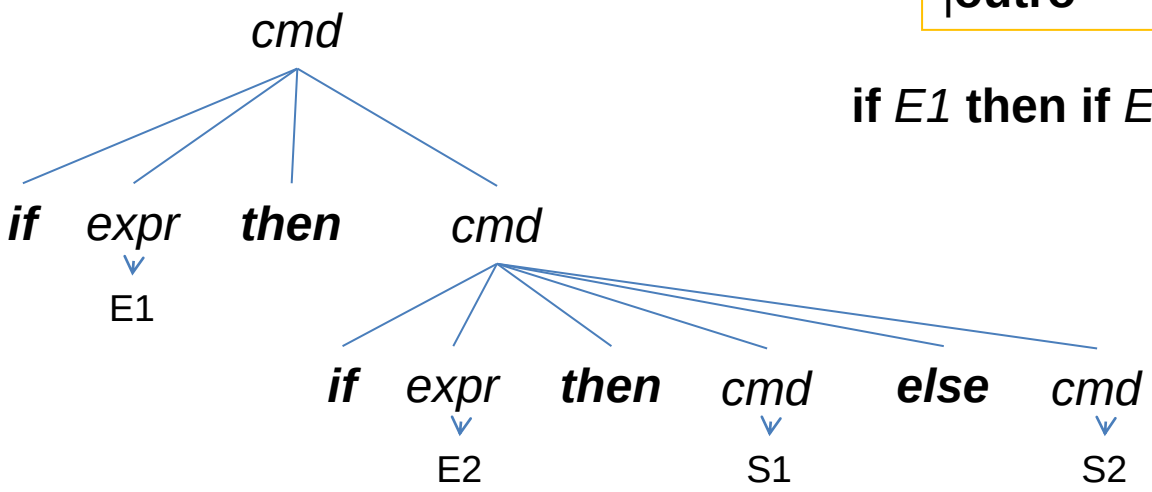
|**outro**

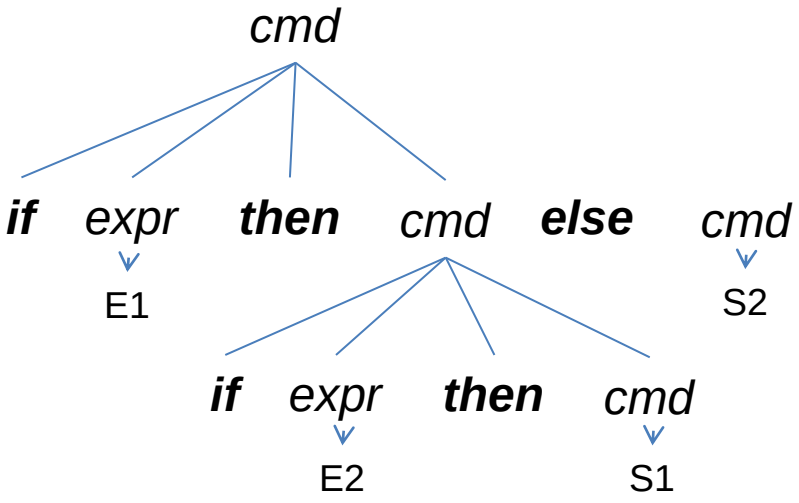
if E1 **then** S1 **else if** E2 **then** S2 **else** S3



cmd ::= **if** *expr* **then** *cmd*
|**if** *expr* **then** *cmd* **else** *cmd*
|**outro**

if *E1* **then if** *E2* **then** *S1* **else** *S2*





<u>Regra geral</u>: associar cada **else** ao **then** anterior mais próximo

Reescrevendo a Gramática

- Todo enunciado entre um **then** e um **else** precisa ser "associado", isto é não pode terminar com um **then** ainda não "associado"
- Um enunciado associado ou é um enunciado **if-then-else** contendo somente enunciados associados ou é qualquer outro tipo de enunciado incondicional

cmd ::= *cmd_associado*
|*cmd_não_associado*
cmd_associado ::= **if** *expr* **then** *cmd_associado* **else** *cmd_associado*
|**outro**
cmd_não_associado ::= **if** *expr* **then** *cmd*
| **if** *expr* **then** *cmd_associado* **else** *cmd_não_associado*

Para Próxima Aula

- Tipos de Analisadores Sintácticos
- Formação de BNF

FIM!!!

Duvidas e Questões?