Licenciatura em Engenharia Informática – DEI/ISEP
Linguagens e Programação 2020/2021
Aula Prática-Laboratorial

Ficha PL 2

Gramáticas

Objetivos:

- Aprendizagem dos conceitos apresentados na Ficha TP3, através da realização de exercícios
- Definir árvore de derivação
- Conversão entre AF e gramáticas
- Classificação quanto à ambiguidade

1. Gramáticas

Uma gramática é uma ferramenta poderosa para a descrição e análise de linguagens. É constituída por um conjunto de regras segundo as quais as frases válidas da linguagem são construídas e por vários tipos de palavras (*tokens*) normalmente reconhecidos pelo analisador léxico.

1.1 *Exercícios propostos*

1. Considere a seguinte gramática:

   `S → (L)|a`

   `L → L,S|S`

   a) Identifique os símbolos terminais e não terminais desta gramática;
   b) Determine uma árvore de derivação desta gramática para (`a,a`);
   c) Crie um autómato equivalente a esta gramática;
   d) Caracterize formalmente a linguagem representada por esta gramática.

2. Escreva uma gramática capaz de reconhecer cada uma das seguintes linguagens:
   a) Palavras no alfabeto Σ = {`a`, `b`} que terminam em "`b`" e começam em "`a`"
   b) Palavras no alfabeto Σ = {`a`, `b`, `c`, `d`} em que um "`b`" é sempre precedido de um "`a`"
   c) Palavras no alfabeto Σ= {`a`, `b`, `c`, `d`} que são palíndromas e têm um comprimento maior que 1;

3. Defina uma gramática capaz de representar uma quantia monetária nas moedas apresentadas na tabela seguinte.

Tabela 4.1: Representação de moedas

Moeda	Exemplo
Euro	`e12,23; e1,00; e2,35; 23,50EUR`
Libra	`£12.50; £22.12; £22.99`
Dólar	`$25.13; $5.00; $0.30`
Escudo	`12$50; 25$00; 150$00; 0$50`

4. Considere a seguinte gramática G = (V, Σ,P,S):

S→ TyBT
T→ xT | x
B→ zB | ε

a) Defina formalmente a gramática G.

b) Classifique a gramática G, segundo a hierarquia de Chomsky. Justifique.

c) Caso seja possível, represente um Autómato Finito que reconheça a linguagem gerada pela gramática G. Justifique.

5. Considere a expressão regular **($x^+yz^*x^+$):**
   a) Representa a mesma linguagem que a gramática G do exercício 4? Justifique.
   b) Verifique se palavra **xyyzzxx** é válida no âmbito da expressão regular anterior. Justifique.

6. Considere a gramática G=({S,L,A},{(,),a,+,b},{S→ (L)|a, L→A+S|b, A→a|L}, S):
   a) Considere a palavra **(b+(a+a)+a),** valide se pertence à linguagem gerada pela gramática. Apresente a sequência de derivação mais à direita para a frase.
   b) Classifique a gramática G segundo a hierarquia de Chomsky. Justifique.
   c) Defina uma expressão regular equivalente à gramática G.
   d) Apresente uma gramática equivalente à Expressão Regular **(#|@)*(@|#)#***

7. Considere uma gramática `G` tal que:

`S → aS | Sb | ab | SS`

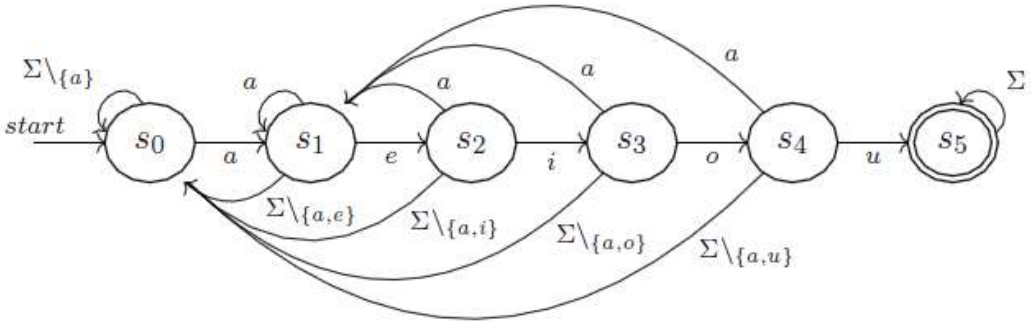
a) Escreva uma expressão regular para reconhecer a linguagem definida pela gramática `G`, ou seja, `L(G)`.
b) Escreva uma sequência de derivação para `aaabb`.
c) Classifique a gramática `G` quanto à ambiguidade.

8. Considere a seguinte gramática:

`S →` `if b then` **`S`** `else` **`S`**
`S →` `if b then` **`S`**
`S →` `a`

a) Mostre que a gramática em questão é ambígua (por exemplo, encontre uma frase que tenha duas árvores sintáticas).
b) Escreva uma gramática equivalente não ambígua.

9. Converta o seguinte autómato numa gramática:



10. De seguida apresentam-se exemplos de declarações válidas.
    - int a, x1=10, y=x1;
    - long int numero;
    - unsigned char c='a';
    - long double real=1.234e-5, pi=3.14159265358979, num1, num2;

    Os tipos válidos são `int`, `char`, `float` e `double`, os *modifiers* que podem aparecer antes do tipo são `short`, `long` e `unsigned`. Os identificadores são uma letra seguida de zero ou mais letras e algarismos. Opcionalmente pode ser efectuada uma atribuição de um valor (constante ou variável).

    Crie uma gramática capaz de reconhecer este tipo de declarações de variáveis. Não é necessário validar a coerência de tipos nas atribuições nem combinações inválidas tipo `short, char`. Use como ponto de partida a seguinte gramática:

    <declaracoes> → **<declaracao><declaracoes>**|ε

    <declaracao> → **<tipo><lista_variaveis>**;

    <tipo> → …

    <lista_variaveis> → …

11. Crie um programa em FLEX, que identifique os *tokens* presentes em expressões lógicas válidas na gramática a seguir descrita, sendo que quando um *token* for identificado, a função yylex deverá devolver o identificador respectivo.

    E → **E** `or` **E**|**E** `and` **E**|**E** `xor` **E**| `not` **E**| `(`**E**`)` | **ID** |**INT**|**REAL**

    NOTA: Um **ID** representa um identificador (uma letra seguida de letras ou algarismos), **INT** um número inteiro e **REAL** um número real. Os espaços, *tabs* e mudanças de linha devem ser ignorados. Qualquer outro carácter deve originar um erro.

1.2 Exercícios complementares

1. Escreva uma gramática capaz de reconhecer cada uma das seguintes linguagens:

    i. Palavras no alfabeto = {a, b, c, d} começam e terminam em a;

    ii. Números romanos menores que 50.

2. Considere uma calculadora básica capaz de processar as quatro operações algébricas fundamentais representadas pelos símbolos +,-, *, /, parêntesis e o '-' unário.
   i. Escreva uma sintaxe para expressões algébricas válidas nesta calculadora;
   ii. Escreva uma árvore de derivação para a expressão: 2 + 3 *4. A árvore de derivação que obteve é única? Que problema está aqui em causa? Explique.
   iii. Adicione as operações: ^ e % à gramática.

3. Considere uma gramática com as seguintes produções (S é a produção inicial):

   S →AB

   A → aB | ε

   B → bAC | ε

   C → c (A+B)

   a) Escreva em FLEX um *parser* em descida recursiva que implemente a gramática anterior;

   b) Diga se as seguintes frases pertencem à gramática, justificando.

      i. abc(+ba)

      ii. abc(a+)

      iii. abc(a+b)

      iv. ac(a+a)a

4. Implemente um analisador léxico para a seguinte gramática:

   S → ID = E|E

   E → E + E|E – E|E * E|E / E| -E| (E) | ID |INT|REAL

5. Converta o seguinte autómato numa gramática: