



Universidade de Aveiro

Departamento de Electrónica, Telecomunicações e Informática

Linguagens Formais e Autómatos + Compiladores

Exame teórico-prático, parte 2

(Ano Letivo de 2018/2019)

modelo

NºMec:

Nome:

1. Sobre o alfabeto $T_1 = \{t b z w a o v n\}$ considere a gramática G_1 dada a seguir e seja L_1 a linguagem por ela descrita.

$P \rightarrow \varepsilon \mid X I t P \mid X b P z P$
 $X \rightarrow \varepsilon \mid w C$
 $I \rightarrow \varepsilon \mid a$
 $C \rightarrow T \mid C o T$
 $T \rightarrow v \mid n T$

- [1,5] (a) Mostre que a palavra $a t w n v b z$ pertence a L_1 .

Optei por fazer uma derivação à esquerda.

$P \Rightarrow X I t P \Rightarrow I t P \Rightarrow a t P \Rightarrow a t X b P z P \Rightarrow a t w C b P z P$
 $\Rightarrow a t w T b P z P \Rightarrow a t w n T b P z P \Rightarrow a t w n v b P z P$
 $\Rightarrow a t w n v b z P \Rightarrow a t w n v b z$

Mas poderia ser com outro tipo de derivação ou com uma árvore de derivação. A menos que seja pedido de uma forma específica.

$\text{first}(X I t P) = ?$

Aplicando o algoritmo: $h = X$; $w = I t P$

Como h é não terminal e há 2 produções começadas por X

$\text{first}(X I t P) = \text{first}(I t P) \cup \text{first}(w C I t P)$

$\text{first}(I t P) = ?$

Aplicando o algoritmo: $h = I$; $w = t P$

Como h é não terminal e há 2 produções começadas por I

$\text{first}(I t P) = \text{first}(t P) \cup \text{first}(a t P)$

O first de uma expressão começada por um terminal é o conjunto singular com esse terminal. Logo

$\text{first}(w C I t P) = \{w\}$

$\text{first}(t P) = \{t\}$

$\text{first}(a t P) = \{a\}$

Donde:

$\text{first}(I t P) = \{a t\}$

$\text{first}(X I t P) = \{a t w\}$

- [1,5] (b) Considere o conjunto $F = \text{first}(X I t P)$.

Das seguintes afirmações, assinale as que são verdadeiras

☒

$t \in F$

☐

$b \in F$

☒

$w \in F$

☒

$a \in F$

- [1,5] (c) Considere o conjunto $G = \text{follow}(T)$.

Das seguintes afirmações, assinale as que são verdadeiras

☒

$t \in G$

☒

$b \in G$

☐

$w \in G$

☒

$a \in G$

Por definição $\$ \notin \text{follow}(S)$

$\text{follow}(T) = ?$

- pela produção $C \rightarrow T$, $\text{follow}(T)$ contém o $\text{follow}(C)$

$\text{follow}(C) = ?$

- pela produção $X \rightarrow w C$, $\text{follow}(C)$ contém o $\text{follow}(X)$

- pela produção $C \rightarrow C o T$, $\text{follow}(C)$ contém $\text{first}(o T) = \{o\}$

$\text{follow}(X) = ?$

- pela produção $S \rightarrow X I t P$, $\text{follow}(X)$ contém $\text{first}(I t P) = \{a t\}$

- pela produção $S \rightarrow X b P z P$, $\text{follow}(X)$ contém $\text{first}(b P z P) = \{b\}$

Logo:

$\text{follow}(X) = \{a b t\}$, $\text{follow}(C) = \{a b t o\}$, $\text{follow}(T) = \{a b t o\}$

- [2,0] (d) Considere o conjunto $H = \text{predict}(P \rightarrow X I t P)$.

Das seguintes afirmações, assinale as que são verdadeiras.

☒

$t \in H$

☐

$b \in H$

☒

$w \in H$

☒

$a \in H$

$\text{first}(X I t P) = \{a t w\}$ (ver 1b)

Como $\$ \notin \text{first}(X I t P)$

$\text{predict}(P \rightarrow X I t P) = \{a t w\}$

- [2,0] (e) As produções começadas por P e C tornam a gramática G_1 inadequada à implementação de um reconhecedor descendente com *lookahead* de 1. Altere-a de forma a obter uma equivalente que o permita.

não pode ter prefixos comuns
é preciso resolver
 $P \rightarrow X \mid t P \mid X b P z P$

não pode ter recursividade à esquerda
é preciso resolver
 $C \rightarrow T \mid C o T$

Removendo os prefixos comuns

$P \rightarrow \backslash e \mid X Y$
 $Y \rightarrow \mid t P \mid b P z P$

Analisando a recursividade à esquerda

$C \Rightarrow C o T \Rightarrow C o T o T \Rightarrow \dots \Rightarrow C (o T)^* \Rightarrow T (o T)^*$

Permite eliminá-la

$C \rightarrow T Z$
 $Z \rightarrow \backslash e \mid o T Z$

2. Considere o alfabeto $A = \{a, b, c\}$ e seja L_2 o conjunto de todas as expressões regulares definíveis sobre o alfabeto A . L_2 é uma linguagem independente do contexto definida sobre o alfabeto $T_2 = A \cup \{ (,), *, \mid \}$, em que $*$ é o operador de fecho, \mid o operador de escolha e em que o operador de concatenação é implícito. Em termos de precedência, e da mais alta para a mais baixa, estão as operações de fecho, concatenação e escolha. Os parêntesis podem ser usados para alterar a precedência por defeito.

- [3,0] (.) Construa uma gramática independente do contexto que represente a linguagem L_2 .

De forma a evitar confusões entre o operador \mid da linguagem e o usado na gramática, usa-se $+$ no lugar do primeiro

Exemplos de palavras aceites pela linguagem:

a^*
 $a+c$
 ac
 $(ac)^*$
 $((ac|b)^*|aa)^{**}$

Definição da precedência: uma expressão é uma soma de T, um T é uma concatenação de F, um F é K fechado 0 ou mais vezes, e K é uma letra ou uma expressão entre parêntesis

$E = T + T + T \dots$

$T = F F F \dots$

$F = K * * \dots$

$K = N \mid (E)$

Gramática, onde se usa o $+$ em vez do \mid :

$E \rightarrow T \mid E + T$

$T \rightarrow F \mid T F$

$F \rightarrow K \mid F *$

$K \rightarrow N \mid (E)$

$N \rightarrow a \mid b \mid c$

3. Sobre o alfabeto $T_3 = \{\text{NUM}, \text{BOX}, \text{CIRCLE}, \text{THICKNESS}, \text{COLOR}, '\{', '\}', '\}\}$, considere a gramática G_3 dada a seguir e seja L_3 a linguagem por ela descrita.

```

draw  →  seq
seq   →  ε
      |  seq item
item  →  COLOR NUM
      |  THICKNESS NUM
      |  CIRCLE point NUM
      |  BOX point '{' seq '}'
point →  NUM NUM

```

Considere ainda a coleção de conjunto de itens usada na construção de um reconhecedor ascendente parcialmente apresentada a seguir, onde $\delta(Z_i, a)$ representa a função de transição de estado.

```

Z0 = {draw → • seq, seq → •, seq → • seq item}
Z1 = δ(Z0, seq) = {draw → seq •, seq → seq • item, item → • COLOR NUM, item → • THICKNESS NUM,
                    item → • CIRCLE point NUM, item → • BOX point '{' seq '}' }
Z2 = δ(Z1, item) = {seq → seq item •}
Z3 = δ(Z1, COLOR) = {item → COLOR • NUM}
Z4 = δ(Z1, THICKNESS) = {item → THICKNESS • NUM}
Z5 = δ(Z1, CIRCLE) = {•••}
Z6 = δ(Z1, BOX) = {•••}
Z7 = δ(Z3, NUM) = {item → COLOR NUM •}
Z8 = δ(Z4, NUM) = {item → THICKNESS NUM •}
follow(seq) = { $ CO TH CI BO '}'

```

- [2,0] (a) Preencha as linhas da tabela de reconhecimento (*parsing*) para um reconhecedor ascendente relativamente aos estados Z_0 a Z_4 .

	ACTION							GOTO			
	NUM	BOX	CIRCLE	THICKNESS	COLOR	{	}	\$	seq	item	point
Z_0		r2	r2	r2	r2		r2	r2	z1		
Z_1		s, z6	s, z5	s, z4	s, z3			accept		z2	
Z_2		r3	r3	r3	r3		r3	r3			
Z_3	s, z7										
Z_4	s, z8										

- [2,0] (b) Determine os conjuntos de itens definidores dos estados Z_5 , Z_6 e de mais três, além dos apresentados.

Usa-se CI, BO em vez de CIRCLE, BOX

$z5 = \delta(z1, CI) = \{ \text{item} \rightarrow CI \cdot \text{point NUM} \} \cup \{ \text{point} \rightarrow \cdot \text{NUM NUM} \}$

$z6 = \delta(z1, BO) = \{ \text{item} \rightarrow BO \cdot \text{point '{' seq '}' } \} \cup \{ \text{point} \rightarrow \cdot \text{NUM NUM} \}$

$z9 = \delta(z5, \text{point}) = \{ \text{item} \rightarrow CI \text{ point} \cdot \text{NUM} \}$

$z10 = \delta(z5, \text{NUM}) = \{ \text{point} \rightarrow \text{NUM} \cdot \text{NUM} \}$

$z11 = \delta(z6, \text{point}) = \{ \text{item} \rightarrow BO \text{ point} \cdot \text{'{' seq '}' } \}$

este é um adicional, pedido por um aluno

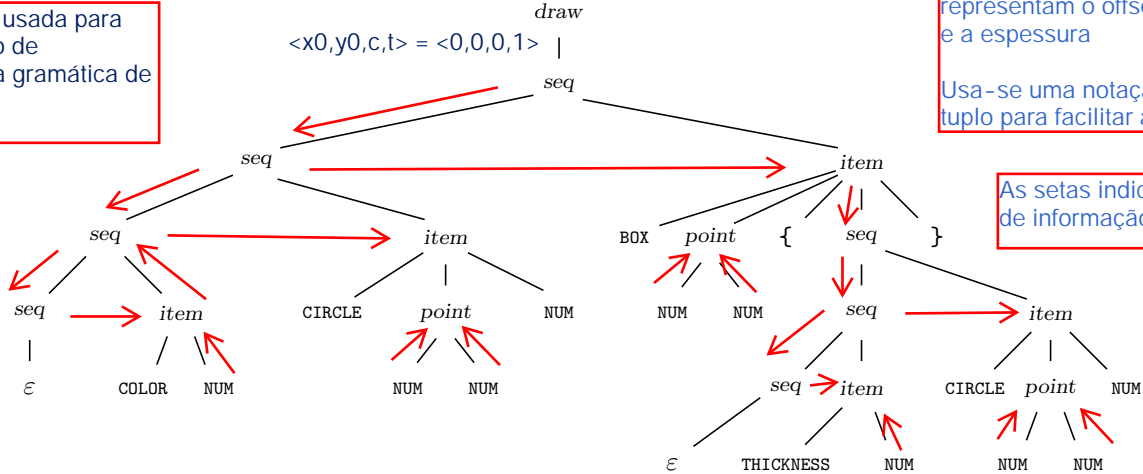
$z12 = \delta(z11, '{') = \{ \text{item} \rightarrow BO \text{ point '{' } \cdot \text{seq '}' } \} \cup \{ \text{seq} \rightarrow \cdot \text{, seq} \rightarrow \cdot \text{seq item} \}$

4. Considere novamente a gramática G_3 dada no exercício anterior. Uma palavra na linguagem dada por G_3 descreve um desenho definido por uma sequência das seguintes operações gráficas (*item*):
- **COLOR NUM**, que permite mudar a cor da caneta de desenho para a dada por NUM.
 - **THICKNESS NUM**, que permite mudar a espessura da caneta de desenho para a dada por NUM.
 - **CIRCLE point NUM**, que desenha um circunferência centrada no ponto dado por *point* e com raio dado por NUM, usando a caneta de desenho ativa.
 - **BOX point '{' seq '}'**, que cria um sub-desenho com um *offset* dado por *point* em relação ao desenho dentro do qual fica. O ponto (0,0) do sub-desenho é o ponto *point* do desenho onde está incluído.

Apenas o símbolo terminal NUM tem um atributo associado, designado *v* e que representa um número. O símbolo não terminal *point* representa as coordenadas X e Y de um ponto. A configuração inicial do sistema é caracterizada por cor 0, espessura 1 e *offset* (0,0). Finalmente, considere que dispõe da função `drawCircle(x, y, r, c, t)` que desenha uma circunferência centrada no ponto (x,y), com raio r, usando uma caneta de desenho com cor c e espessura t.

- [1,5] (a) Trace a árvore de derivação da palavra
 COLOR NUM CIRCLE NUM NUM NUM BOX NUM NUM '{' ' THICKNESS NUM CIRCLE NUM NUM NUM '}'
 Se quiser, ao traçar a árvore, pode abreviar a designação dos símbolos, desde que isso não afete a interpretação da sua resposta.

Esta árvore é usada para ilustrar o fluxo de informação da gramática de atributos



Associam-se a seq e item 4 atributos <x0,y0,c,t> que representam o offset, a cor e a espessura
 Usa-se uma notação em tuplo para facilitar a escrita

As setas indicam o fluxo de informação.

- [3,0] (b) Construa uma gramática de atributos que permita invocar a função `drawCircle` de forma adequada para cada circunferência incluída num desenho.

Produção	Regra semântica
$draw \rightarrow seq$	$seq.<x0,y0,c,t> = <0,0,0,1>$
$seq \rightarrow \epsilon$	
$seq_1 \rightarrow seq_2 item$	$seq2.<x0,y0,c,t> = seq1.<x0,y0,c,t>$ $item.<x0,y0,c,t> = seq2.<x0,y0,c,t>$
$item \rightarrow COLOR\ NUM$	$item.c = NUM.v$
$item \rightarrow THICKNESS\ NUM$	$item.t = NUM.v$
$item \rightarrow CIRCLE\ point\ NUM$	$drawCircle(item.x0+point.x, item.y0+point.y, NUM.v, item.c, item.t);$
$item \rightarrow BOX\ point\ \{ seq \}$	$seq.<x0,y0,c,t> = <item.x0+point.x, item.y0+point.y, item.c, item.t>$
$point \rightarrow NUM_1\ NUM_2$	$point.x = NUM_1.v; point.y = NUM_2.v$

CÁBULA OFICIAL

first:

```
first( $\alpha$ ) {  
  if ( $\alpha \in T \vee \alpha = \varepsilon$ ) then  
    return { $\alpha$ }  
  else if ( $\alpha \in N$ ) then  
    return  $\bigcup_{(\alpha \rightarrow \beta) \in P} \text{first}(\beta)$   
  else  
     $h = \text{head}(\alpha)$     # com  $|h| = 1$   
     $\beta = \text{tail}(\alpha)$   
    if  $\varepsilon \notin \text{first}(h)$  then  
      return first( $h$ )  
    else  
      return (first( $h$ ) - { $\varepsilon$ })  $\cup$  first( $\beta$ )  
}
```

follow:

```
foreach  $X \in N$   
  follow( $X$ ) =  $\emptyset$   
 $\$ \in \text{follow}(S)$   
do  
  if ( $A \rightarrow \alpha B$ )  $\in P$  then  
    follow( $B$ )  $\supseteq$  follow( $A$ )  
  else /* ( $A \rightarrow \alpha B \beta$ )  $\in P$  */  
    if  $\varepsilon \in \text{first}(\beta)$  then  
      follow( $B$ )  $\supseteq$  first( $\beta$ )  
    else  
      follow( $B$ )  $\supseteq$  (first( $\beta$ ) - { $\varepsilon$ })  $\cup$  follow( $A$ )  
while <any set changes>
```

predict:

$$\text{predict}(A \rightarrow \alpha) = \begin{cases} \text{first}(\alpha) & \varepsilon \notin \text{first}(\alpha) \\ (\text{first}(\alpha) - \{\varepsilon\}) \cup \text{follow}(A) & \varepsilon \in \text{first}(\alpha) \end{cases}$$