Processamento de Linguagens I
LESI + LMCC (3º ano)

4º Ficha Prática

Ano Lectivo de 05/06

1 Objectivos

Este ficha prática contém exercícios para serem resolvidos nas aulas teórico-práticas com vista a sedimentar os conhecimentos relativos a

- Análise de Gramática para construção de Tabelas de Parsing LL (*Top-Down*); verificação da Condição LL(1).
- Análise de Gramática para construção de Tabelas de Parsing LR (*Bottom-Up*); verificação da Condição LR(0)/SLR(1).

2 Enunciados

No contexto do desenvolvimento de Compiladores, ou mais genericamente de Processadores de Linguagens, a primeira tarefa a realizar é a criação de uma GIC que especifique a linguagem que se quer processar, visto que todo o desenvolvimento vai assentar sobre a dita gramática.
Em particular, o **reconhecedor sintáctico (parser)** é construído a partir de uma algoritmo iterativo genérico, muito eficiente e independente da gramática, que é guiado por uma **tabela de decisão**, a qual é específica de cada linguagem e se cria sistematicamente a partir da GIC concreta.
Nessas condições, propõe-se para esta aula a construção da tabela LL(1) e da tabela LR(0), ou SLR(1), a partir da gramática indicada em cada exercício, a verificação das condições de reconhecimento para cada uma das estratégias (TD ou BU) e a eventual transformação da GIC.

2.1 Gramática simples

Considere a seguinte gramática independente de contexto $G_1 = (T, N, S, P)$ com $T = \{a, b, c\}$, $N = \{S, A\}$ e,

$$
\begin{array}{rcll}
P & = & \{ & p_1 : \; S \;\; \rightarrow \;\; A \; B \; c \\
 & & , & p_2 : \;\;\;\; | \;\; B \\
 & & , & p_3 : \; A \;\; \rightarrow \;\; A \; a \\
 & & , & p_4 : \;\;\;\; | \;\; a \\
 & & , & p_5 : \; B \;\; \rightarrow \;\; b \\
 & & \} &
\end{array}
$$

Como provar que, por exemplo, $abbc \in \mathcal{L}_{G_1}$?
Prova-se que uma frase pertence à linguagem de uma dada gramática se for possível encontrar uma sequência de derivação para essa dada frase ou se for possível construir uma árvore de parsing.
Utilizando uma sequência de derivação LL (cuja re-escrita se faz sempre à esquerda) temos:

$$S \overset{\#1}{\Rightarrow} ABc \overset{\#1}{\Rightarrow} AaBc \overset{\#1}{\Rightarrow} aaBc \overset{\#1}{\Rightarrow} aabc$$

Utilizando uma sequência de derivação LR (cuja re-escrita se faz sempre à direita) temos:

$$S \overset{\#1}{\Rightarrow} ABc \overset{\#1}{\Rightarrow} Abc \overset{\#1}{\Rightarrow} Aabc \overset{\#1}{\Rightarrow} aabc$$

Utilizando um reconhecedor Bottom-up, a árvore de derivação de uma frase f é construída começando pelas folhas (bottom) e subindo pela raiz S.

$$f \overset{\#1}{\Rightarrow} S$$

Assim, informalmente, o reconhecimento da frase *aabc* faz-se da seguinte forma:

aabc	p_4
Aabc	p_3
Abc	p_5
ABc	p_1
S	

Note-se que a sequência de derivação conseguida utilizando um reconhecedor bottom-up é igual à sequência de derivação LR apresentada anteriormente.

No entanto, esta sequência apesar de provar que a frase *aabc* pertence à linguagem, não mostra como é que o algoritmo bottom-up funciona. Uma versão simplificada desse algortimo é apresentado a seguir utilizando uma tabel de três entradas: stack, frase de entrada e a acção a executar.

As acções pode ser de dois tipos: desloca (que remove um símbolo da frase para a stack) e reduz (na qual uma regra de produção é utilizada).

Stack	f	acção
	aabc	descola
a	abc	reduz p/p_4
A	abc	desloca
Aa	bc	reduz p/p_3
A	bc	desloca
Ab	c	reduz p/p_5
AB	c	desloca
ABc		reduz p/p_1
S		aceite

Ainda assim, no exemplo acima representado nada nos é dito qual a escolha das acções que temos que executar, e quando se faz a acção reduz qual a produção escolhida.

Para tal, vai-se introduzir o algortimo SLR ou LR(0).

Uma gramática diz-se SLR ou LR(0) se e só se for possível construir para ela um reconhecedor SLR.

Para tal, é necessário construir duas tabelas: uma tabela de transição de estados e uma tabela de acções.

Tabela de transição de estados:

$$\delta = array[\delta \times (T \cup N)] \ de \ Q \cup \{erro\}$$

Tabela de acções:

$$TA = array[Q, T] \ de \ P \cup \{desloca, ok, erro\}$$

construida segundo as seguintes regras que serão explicadas mais à frente:

$$\begin{aligned} A \rightarrow\ \delta\ .\ a\ \beta \quad &\in I_i \quad \Rightarrow ta[i,a] = desloca \\ A \rightarrow\ \alpha\ . \quad &\in I_i \quad \Rightarrow \forall_a \in Follow(A), ta[i,a] = A \rightarrow \alpha \\ S \rightarrow\ S\ .\ \$ \quad &\in I_i \quad \Rightarrow ta[i,a] = ok \end{aligned}$$

Começa-se por um passo de preparação em que se verifica se a gramática inclui o símbolo de fim-de-ficheiro \$. Como não exemplo dado, tal não acontece, a gramática vai ser extendida através da adição de um novo símbolo não-terminal:
$G = (T, N, S, P)$, com $T = \{a, b, c, \$\}$, $N = \{S', S, A\}$ e,

$$\begin{aligned} P \ = \ \{ \ & p_0 : \ S' \ \rightarrow \ S\ \$ \\ , \ & p_1 : \ S \ \rightarrow \ A\ B\ c \\ , \ & p_2 : \ \ \ \ | \ B \\ , \ & p_3 : \ A \ \rightarrow \ A\ a \\ , \ & p_4 : \ \ \ \ | \ a \\ , \ & p_5 : \ B \ \rightarrow \ b \\ \} & \end{aligned}$$

O primeiro passo para a construção das tabelas SLR é a definição de um autómato finito não determinista a partir da gramática.

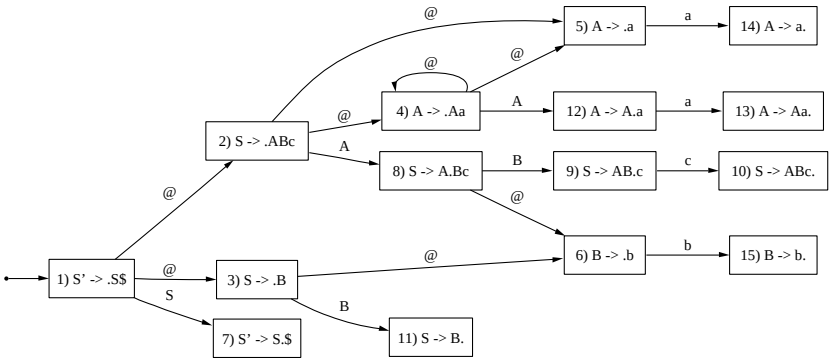


Figura 1: NDFA da gramática independente do contexto.

Cada estado do automáto representa uma produção e o símbolo que já foi processado. O resultado desse autómato pode ser visualizado na Figura 1.

Por exemplo o estado do autómato 1)$S' \rightarrow \ .\ S\ \$$, representa o axioma da gramática no qual nenhum estado foi processado. Este estado é o estado inicial do autómato. Processando o símbolo S, vai-se para o estado 7, representado por 7)$S' \rightarrow \ S\ .\ \$$. Note-se que o ponto indica que o símbolo S foi processado. As transições por ϵ representadas pelo símbolo @ existem quando a seguir de um ponto aparece um símbolo não-terminal, e fazem-se para todas as produções desse símbolo colocando um ponto antes dos símbolos do lado direito.

Depois de construído o NDFA que representa a gramática, é necessário convertê-lo para DFA.

Calculando as transições dos símbolos tem-se:

$$
\begin{array}{lcll}
I_0 & = & \epsilon - closure(1) & = \{1,2,3,4,5,6\} \\
I_1 & = & \delta(I_0, S) & = \{7\} \\
I_2 & = & \delta(I_0, a) & = \{14\} \\
I_3 & = & \delta(I_0, b) & = \{15\} \\
I_4 & = & \delta(I_0, A) & = \{6,8,12\} \\
I_5 & = & \delta(I_0, B) & = \{11\} \\
I_6 & = & \delta(I_4, a) & = \{13\} \\
I_7 & = & \delta(I_4, B) & = \{9\} \\
I_8 & = & \delta(I_4, b) & = \{15\} \\
I_9 & = & \delta(I_7, c) & = \{10\}
\end{array}
$$

O que permite definir a seguinte tabela de conversão:

estados	S	A	B	a	b	c
{1,2,3,4,5,6}	{7}	{6,8,12}	{11}	{14}	{15}	
{7}						
{6,8,12}			{9}	{13}	{15}	
{11}						
{14}						
{15}						
{9}						{10}
{13}						
{10}						

O autómato finito determinista equivalente pode ser visualizado na Figura 2.

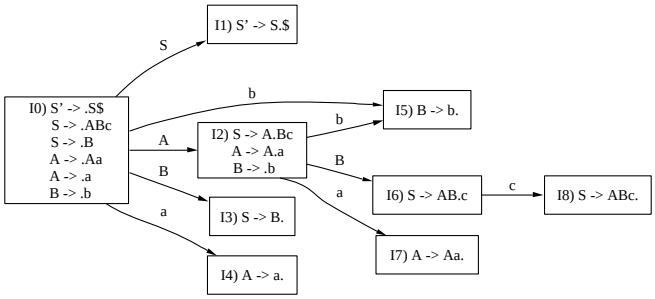


Figura 2: DFA equivalente da gramática independente do contexto.

Através deste autómato, utilizando as regras para cálculo da tabela de acções obtém-se o lado esquerdo da tabela apresentada a seguir. Da função δ desse automáto, deriva-se o lado direito da tabela.

	a	b	c	$	S'	S	A	B
0	shift4	shift5				1	2	3
1				ok				
2	shift7	shift5						6
3				p_2				
4	p_4							
5			p_5	p_5				
6			shift8					
7	p_3	p_3						
8				p_1				

Agora tendo a tabela de acções e de estados o algoritmo de aceitação bottom-up torna-se um pouco mais simples. Provando que $aabc \in \mathcal{L}_{G_1}$ tem-se:

Stack de parsing	Entrada	Acções
0	aabc$	$TA(0, a) = desloca$
0a	abc$	$\delta(0, a) = 4$
0a4	abc$	$TA(4, a) = p_4 = A \rightarrow a$, $\|a\| = 1$, retira 1×2 símbolos, acrescenta A
0A	abc$	$\delta(0, A) = 2$
0A2	abc$	$TA(2, a) = desloca$
0A2a	bc$	$\delta(2, a) = 7$
0A2a7	bc$	$TA(7, b) = p_3 = A \rightarrow A\ a$, $\|Aa\| = 2$, retira 2×2 símbolos, acrescenta A
0A	bc$	$\delta(0, A) = 2$
0A2	bc$	$TA(2, b) = desloca$
0A2b	c$	$\delta(2, b) = 5$
0A2b5	c$	$TA(5, c) = p_5 = B \rightarrow b$, $\|b\| = 1$, retira 1×2 símbolos, acrescenta B
0A2B	c$	$\delta(2, B) = 6$
0A2B6	c$	$TA(6, c) = desloca$
0A2B6c	$	$\delta(6, c) = 8$
0A2B6c8	$	$TA(8, \$) = p_1 = S \rightarrow A\ B\ c$, $\|ABc\| = 3$, retira 3×2 símbolos, acrescenta S
0S	$	$\delta(0, S) = 1$
0S1	$	$TA(1, \$) = OK$!!!

2.2 Horto de Braga

Determinado horto desta cidade faz propostas de fornecimento de plantas (árvores e arbustos) para construir, ou reconstruir, jardins exteriores, públicos ou particulares. Nesse contexto pretende-se desenvolver um processador de linguagens para implementar diversas operações associadas à gestão corrente do Horto.
Analise atentamente a seguinte gramática independente de contexto, que está simplificada por razões óbvias. O Símbolo Inicial é `Flores` e os Símbolos Terminais são escritos em minúsculas (pseudo-terminais), ou em maiúscula (palavras-reservadas), ou entre apostrofes (sinais de pontuação). A string nula é denotada por `&` e o caracter `$` representa o fim-de-ficheiro (do texto de entrada).

```
p1:  Flores    -->  FsExt  FsInt
p2:  FsExt     -->  FEXTERIOR  Fs
p3:  FsInt     -->  &
p4:               |  FINTERIOR
p5:  Fs        -->  Flor  MaisFs
p6:  MaisFs    -->  &
p7:               |  ","  Fs
p8:  Flor      -->  Cod  NomVulgar  Preco
p9:  NomVulgar-->  str
p10: Preco     -->  num
p11: Cod       -->  pal
```

Calcule então as Tabelas aLL e LR acima pedidas.

Resolução – 1.parte

Comecemos por construir a Tabela de Decisão LL(1), o que nos leva a calcular os *lookahead* de cada uma das produções em P[1].
Para isso consideraremos que são *anulavéis* apenas os símbolos FsInt e MaisFs.

- p1

$$\begin{aligned} lookahead(Flores \rightarrow \text{FsExt FsInt}) &= First(FsExt) \\ &= First(FEXTERIOR) \\ &= \{FEXTERIOR\} \end{aligned}$$

- p2

$$\begin{aligned} lookahead(FsExt \rightarrow \text{FEXTERIOR Fs}) &= First(FEXTERIOR) \\ &= \{FEXTERIOR\} \end{aligned}$$

- p3

[1] Recorde as fórmulas de cálculo no documento com as definições formais em anexo.

$$
\begin{aligned}
lookahead(FsInt \rightarrow \epsilon) &= First(\epsilon) \bigcup Follow(FsInt) \\
&= \emptyset \bigcup Follow(FsInt) \\
&= First(\epsilon) \bigcup Follow(Flores) \\
&= \{\$\}
\end{aligned}
$$

- p4

$$
\begin{aligned}
lookahead(FsInt \rightarrow FINTERIOR) &= First(FINTERIOR) \\
&= \{FINTERIOR\}
\end{aligned}
$$

- p5

$$
\begin{aligned}
lookahead(Fs \rightarrow FlorMaisFs) &= First(Flor) \\
&= First(Cod) \\
&= \{pal\}
\end{aligned}
$$

- p6

$$
\begin{aligned}
lookahead(\text{MaisFs} \rightarrow \epsilon) &= First(\epsilon) \bigcup Follow(MaisFs) \\
&= \emptyset \bigcup Follow(Fs) \\
&= Follow(FsExt) \bigcup Follow(MaisFs) \\
&= First(FsInt) \bigcup Follow(Flores) \\
&= \{FINTERIOR, \$\}
\end{aligned}
$$

- p7

$$
\begin{aligned}
lookahead(MaisFs \rightarrow \text{","Fs}) &= First(\text{","}) \\
&= \{\text{","}\}
\end{aligned}
$$

- p8

$$
\begin{aligned}
lookahead(Flor \rightarrow \text{Cod NomVulgar Preco}) &= First(Cod) \\
&= \{pal\}
\end{aligned}
$$

- p9

$$\begin{aligned} lookahead(NomVulgar \rightarrow str) &= First(str) \\ &= \{str\} \end{aligned}$$

- p10

$$\begin{aligned} lookahead(Cod \rightarrow pal) &= First(pal) \\ &= \{pal\} \end{aligned}$$

- p11

$$\begin{aligned} lookahead(Preco \rightarrow \text{num}) &= First(num) \\ &= \{num\} \end{aligned}$$

A partir dos resultados obtidos no cálculo dos *lookahead*, podemos passar, então, à construção da tabela LL(1), seguindo sistematicamente o algoritmo recordado no dcumento anexo.
Obtém-se, assim, a tabela seguinte:

	FEXTERIOR	FINTERIOR	","	str	pal	num	$
Flores	p1						
FsExt	p2						
FsInt		p4					p3
Fs					p5		
MaisFs		p6	p7				p6
Flor					p8		
NomVulgar				p9			
Cod					p10		
Preco						p11	

Podemos, finalmente, concluir que **a gramática é LL(1)**, uma vez que:

$$\forall_{A\rightarrow\alpha_1, A\rightarrow\alpha_2} : lookahead(A \rightarrow \alpha_1) \bigcap lookahead(A \rightarrow \alpha_2) = \emptyset$$

Esta conclusão, que se pode tirar de imediato após o cálculo dos *lookahead*, está também patente na tabela construida uma vez que esta não apresenta nenhum conflito em alguma das suas entradas.

Resolução – 2.parte

Para obter a tabelas de decisão BU, ACTION e GOTO, tem de se começar por construir o autómato de reconhecimento (DFA) LR(0), seguindo o método sistemático recordado no documento anexo.
Começando no estado 0 com o item

[S' → •Flores $]

obtemos o a DFA com 16 estados, que se mostra na figura 3.
A partir da função de transição, δ, do autómato LR(0) calculado, é possível derivar as tabelas de decisão BU seguintes.

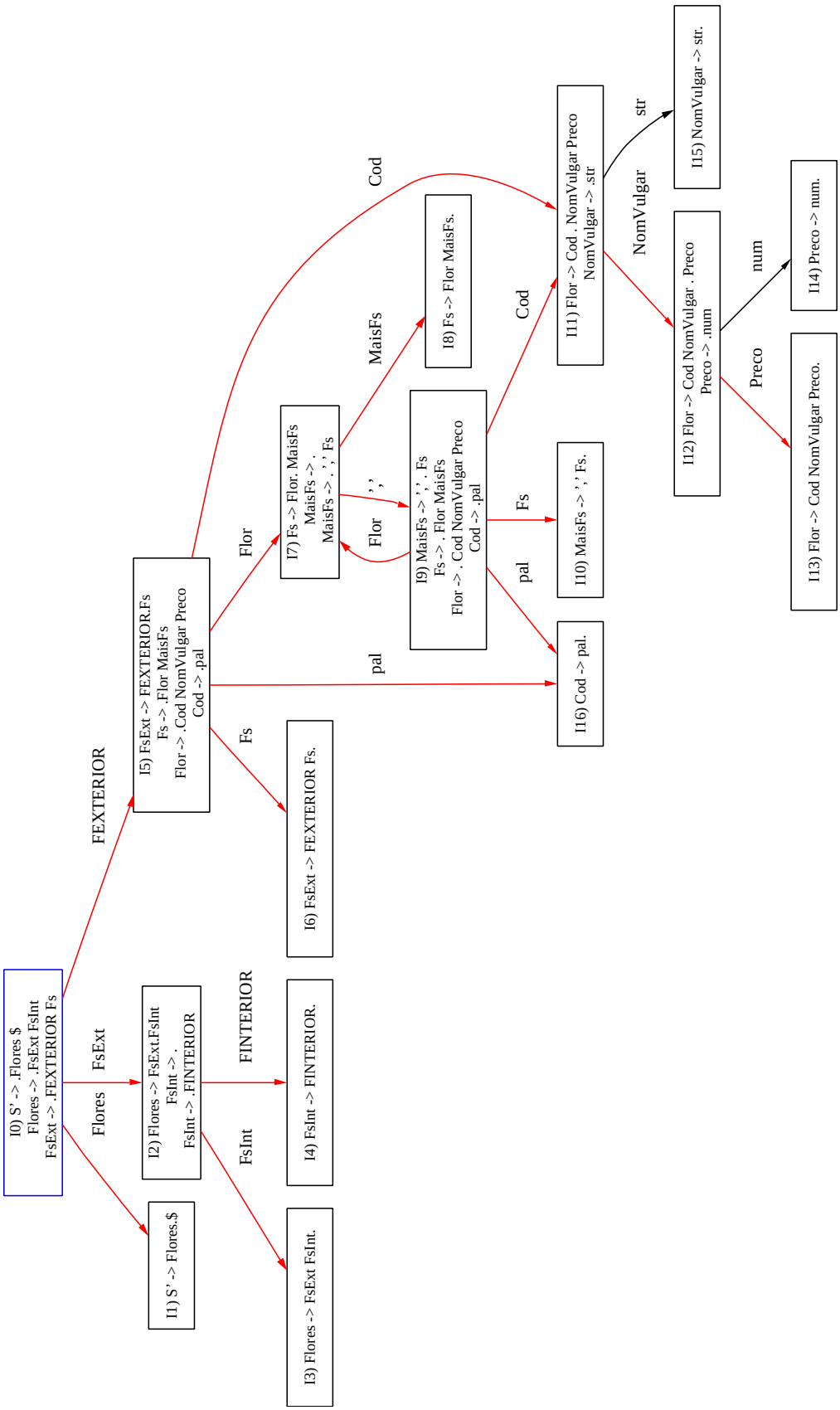


Figura 3: DFA da gramática independente do contexto "Horto de Braga".

Tabela ACTION:

	FEXTERIOR	FINTERIOR	","	str	num	pal	$
0	s5						
1							OK
2	#3	**#3/s4**	#3	#3	#3	#3	#3
3	#1	#1	#1	#1	#1	#1	#1
4	#4	#4	#4	#4	#4	#4	#4
5						s16	
6	#2	#2	#2	#2	#2	#2	#2
7	#6	#6	**#6/s9**	#6	#6	#6	#6
8	#5	#5	#5	#5	#5	#5	#5
9						s16	
10	#7	#7	#7	#7	#7	#7	
11				s15			
12					s14		
13	#8	#8	#8	#8	#8	#8	#8
14	#10	#10	#10	#10	#10	#10	#10
15	#9	#9	#9	#9	#9	#9	#9
16	#11	#11	#11	#11	#11	#11	#11

Tabela GOTO:

	Flores	FsExt	FsInt	Fs	MaisFs	Flor	NomVulgar	Preco	Cod
0	1	2							
1									
2			3						
3									
4									
5				6		7			11
6									
7					8				
8									
9				10		7			11
10									
11							12		
12								13	
13									
14									
15									
16									

Olhando para o DFA da figura 3, ou para a Tabela ACTION acima, é imediato concluir que a gramática em causa **não é LR(0)**, pois existem conflitos *redução/transicção* (*shift/reduce*) nos estados 2 e 7.

Fazendo a redução apenas nos símbolos terminais que pertencem ao *follow* do símbolo à esquerda(LHS) da produção pela qual se quer reduzir—estratégia SLR(1)—e recorrendo aos cálculos que foram efectuados na 1ª parte da resolução, verifica-se que:

- no estado 2 só se reduz pelo símbolo terminal "$" ($Follow(FsInt)$) pelo que se elimina o conflito na coluna $FINTERIOR$
- no estado 7 só se reduz pelo símbolos terminais $\{FINTERIOR, "\$"\}$ ($Follow(MaisFs)$) pelo que se elimina o conflito na coluna ","

concluindo-se, portanto, que a gramática em causa **é SLR(1)**.

2.3 Documento anotado em XML

Neste exercício retoma-se o problema 2.3 da Ficha de Prática nu.º 2, em que se pedia para desenvolver um processador para Documentos Anotados.
Considere a seguinte gramática independente de contexto como uma possível solução para os requisitos impostos nesse enunciado quanto ao conceito de Documento XML.

```
p1:  DocXML     -->  MarcaAbr  Conteudo  MarcaFec
p2:  MarcaAbr   -->  "<"  EleXML  ">"
p3:  MarcaFec   -->  "<"  "/"  id  ">"
p4:  EleXML     -->  id  Atribs
p5:  Atribs     -->  &
p6:  Atribs     -->  Atribs  id  "="  str
p7:  Conteudo   -->  &
p8:  Conteudo   -->  Conteudo  Componente
p9:  Componente-->  pcdata
p10: Componente-->  DocXML
```

Admita que: o Símbolo Inicial é `DocXML`; os Símbolos Terminais são escritos em minúsculas (pseudo-terminais), ou em maiúscula (palavras-reservadas), ou entre aspas (sinais de pontuação); a string nula é denotada por `&` e o caracter `$` representa o fim-de-ficheiro (do texto de entrada).
Calcule então as Tabelas aLL e LR acima pedidas.

Resolução – 1.parte

Comecemos por construir a Tabela de Decisão LL(1), o que nos leva a calcular os *lookahead* de cada uma das produções em P[2].
Para isso consideraremos que são *anulavéis* apenas os símbolos Atribs e Conteudo.

- p1

$$\begin{aligned} lookahead(DocXML \rightarrow MarcaAbrConteudoMarcaFec) &= First(\text{MarcaAbr Conteudo MarcaFec}) \\ &= First(MarcaAbr) \\ &= First('<'\ EleXML'>') \\ &= First('<') \\ &= \{'<'\} \end{aligned}$$

- p2

$$\begin{aligned} lookahead(MarcaAbr \rightarrow '<'\ EleXML'>') &= First('<'\ EleXML'>') \\ &= First('<') \\ &= \{'<'\} \end{aligned}$$

- p3

[2]Recorde as fórmulas de cálculo no documento com as definições formais em anexo.

$$lookahead(MarcaFec) \rightarrow '<' \ '/'\text{id}' >') \quad = \quad First('<') \\ \{'<'\}$$

- p4

$$\begin{aligned} lookahead(EleXML \rightarrow idAtribs) &= First(idAtribs) \\ &= First(id) \\ &= \{id\} \end{aligned}$$

- p5

$$\begin{aligned} lookahead(Atribs \rightarrow \epsilon) &= First(\epsilon) \bigcup Follow(Atribs) \\ &= (First(\epsilon) \bigcup Follow(EleXML)) \bigcup First(id \ ' =' \ str) \\ &= First('>') \bigcup First(id) \\ &= \{'>', id\} \end{aligned}$$

- p6

$$\begin{aligned} lookahead(Atribs \rightarrow Atribsid \ ' =' \ str) &= First(Atribsid \ ' =' \ str) \\ &= (First(\epsilon) \bigcup First(Atribsid' =' \ str)) \bigcup First(id \ ' =' \ str) \\ &= (First(Atribs) \bigcup First(id)) \bigcup First(id) \\ &= \{id\} \end{aligned}$$

- p7

$$\begin{aligned} lookahead(Conteudo \rightarrow \epsilon) &= First(\epsilon) \bigcup Follow(Conteudo) \\ &= First(MarcaFec) \bigcup First(Componente) \\ &= First('<' \ '/'\text{id}' >') \bigcup (First(DocXML) \bigcup First(pcdata)) \\ &= First('<') \bigcup \{pcdata\} \bigcup \{'<'\} \\ &= \{'<', pcdata\} \end{aligned}$$

- p8

$$\begin{aligned} lookahead(Conteudo \rightarrow \ldots) &= First(\text{Conteudo Componente}) \\ &= (First(\epsilon) \bigcup First(\text{Conteudo Componente})) \bigcup First(Componente) \\ &= (First(Conteudo) \bigcup First(Componente)) \\ &\quad \bigcup (First(DocXML) \bigcup First(pcdata)) \\ &= \{'<', pcdata\} \end{aligned}$$

- p9

$$\begin{aligned} lookahead(Componente \rightarrow pcdata) &= First(pcdata) \\ &= \{pcdata\} \end{aligned}$$

- p10

$$\begin{aligned} lookahead(Componente \rightarrow DocXML) &= First(DocXML) \\ &= First(\text{MarcaAbr Conteudo MarcaFec}) \\ &= First(MarcaAbr) \\ &= First('<' \; EleXML' >') \\ &= First('<') \\ &= \{'<'\} \end{aligned}$$

A partir dos resultados obtidos no cálculo dos *lookahead*, podemos passar, então, à construção da tabela LL(1), seguindo sistematicamente o algoritmo recordado no dcumento anexo.
Obtém-se, assim, a tabela seguinte:

	<	/	>	=	id	pcdata	str	$
DocXML	p1							
MarcaAbr	p2							
MarcaFec	p3							
EleXML					p4			
Atribs			p5		**p5/p6**			
Conteudo	**p7/p8**					**p7/p8**		
Componente	p10					p9		

Como podemos ver pelos resultados obtidos no cálculo dos *lookahead*, a gramática **não é LL(1)**, uma vez que:

- lookahead(p5) $\bigcap$ lookahead(p6) = {id}

- lookahead(p7) $\bigcap$ lookahead(p8) = {pcdata, "<"}

Esta conclusão também está patente na tabela LL(1) acima, onde se identifica 1 conflito na linha relativa ao símbolo não-terminal *Atribs* (coluna *id*) e 2 conflitos na linha *Conteudo* (colunas $'<'$ e *pcdata*).

Resolução – 2.parte

Para obter a tabelas de decisão BU, ACTION e GOTO, tem de se começar por construir o autómato de reconhecimento (DFA) LR(0), seguindo o método sistemático recordado no documento anexo. Começando no estado 0 com o item

$$[\text{S'} \rightarrow \bullet\text{DocXML '\$'}]$$

obtemos o a DFA com 20 estados, que se mostra na figura 4.
A partir da função de transição, δ, do autómato LR(0) calculado, é possível derivar as tabelas de decisão BU cuja construção não é aqui incluida ficando ao cuidado do leitor.

Olhando para o DFA da figura 4, é imediato concluir que a gramática em causa **não é LR(0)**, pois existe um conflito *redução/transição* (*shift/reduce*) no estado 12 (redução pela produção $p4$ e transição pelo símbolo terminal *id*.
Neste exemplo é importante observar que os estados 2 e 11 do autómato LR(0) são enganadores, pois apesar de conterem um item de redução (pela produção $p7$ no estado 2, ou $p5$ no estado 11) não há conflitos *redução/transição* nesses estados visto que deles também não há transições por símbolos terminais (só há transições por um não-terminal em cada caso, o que não causa conflito). Fazendo a redução apenas nos símbolos terminais que pertencem ao *follow* do símbolo à esquerda(LHS) da produção pela qual se quer reduzir—estratégia SLR(1)—e recorrendo aos cálculos que foram efectuados na 1ª parte da resolução, verifica-se que:

- no estado 12 só se reduz pelo símbolo terminal $'>'$ ($Follow(ElemXML)$) pelo que se elimina o conflito relativo ao terminal *id*

concluindo-se, portanto, que a gramática em causa **é SLR(1)**.

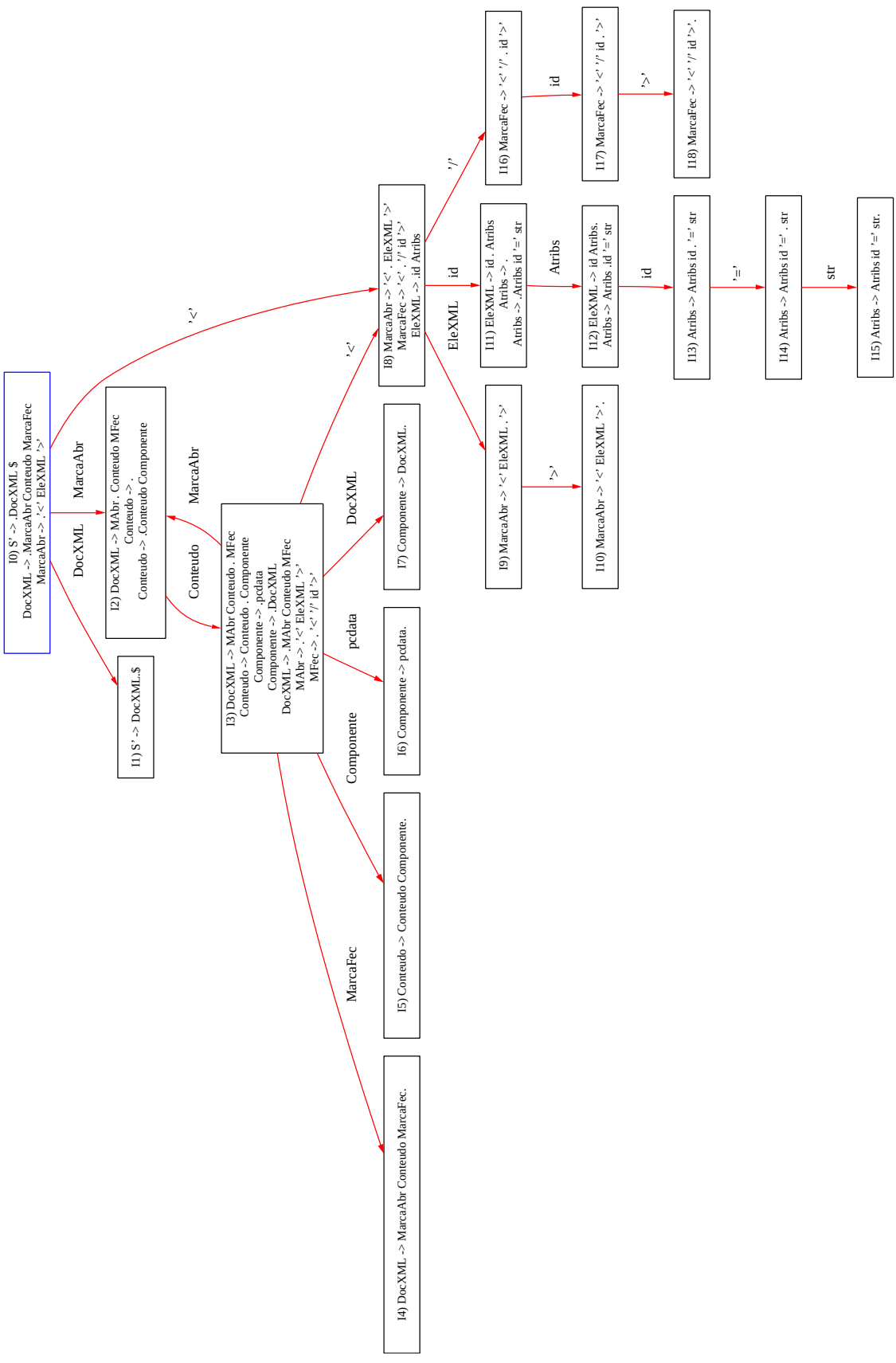


Figura 4: DFA da gramática independente do contexto "Documento XML".

2.4 Documentos sobre o Clero Catedralício

De modo a facilitar o trabalho de uma equipa de historiadores, que está a levantar informação, nos vários arquivos do País, sobre individualidades do Clero Catedralício português, definiu-se uma linguagem que permite processar automaticamente (no sentido de validar, normalizar e criar uma base de dados) as notas extraídas de cada documento consultado.
Mostra-se abaixo a respectiva gramática independente de contexto, na qual: o Símbolo Inicial é `Documentos`; os Símbolos Terminais são escritos em minúsculas (pseudo-terminais), ou em maiúscula (palavras-reservadas), ou entre aspas (sinais de pontuação).

```
p1:  Documentos -> Doc
p2:             |  Documentos Doc
p3:  Doc        -> "{" IdentDoc  IdentClerigos  Evento "}"
p4:  IdentDoc   -> Arq  CotaDoc  DataRedaccao  DataConsulta  Redactor  Tipo
p5:  IdentClerigos -> Clerigo
p6:             |  IdentClerigos ";" Clerigo
p7:  Clerigo    -> Nome  Apelido Diocese CategEcles  Papel
p8 a
p13: Evento, Arq, Redactor, Nome, Apelido, Diocese -> str
p14 a
p16: CotaDoc, Tipo, Papel  -> pal
p17: CategEcles -> BISPO
p18:            |  CONEGO
p19:            |  PRESBITERO
p20: DataRedaccao -> data
p21: DataConsulta -> data
```

Assumindo que o caracter `$` representa o fim-de-ficheiro (do texto de entrada), calcule então as Tabelas aLL e LR acima pedidas.

Resolução – 1.parte

Comecemos por construir a Tabela de Decisão LL(1), o que nos leva a calcular os *lookahead* de cada uma das produções em P[3].
Para isso note-se que nesta gramática, e ao contrário dos exercícios anteriores, não há símbolos *anulavéis*.

- p1

$$\begin{aligned} lookahead(Documentos \rightarrow Doc) &= First(Doc) \\ &= First(\text{"\{"}) \\ &= \{\text{"\{"}\} \end{aligned}$$

- p2

$$\begin{aligned} lookahead(Documentos \rightarrow DocumentosDoc) &= First(Documentos) \\ &= First(Doc) \bigcup First(\text{Documentos}) \end{aligned}$$

[3]Recorde as fórmulas de cálculo no documento com as definições formais em anexo.

$$\begin{aligned} &= First(\text{"\{"}) \\ &= \{\text{"\{"}\} \end{aligned}$$

- p3

$$\begin{aligned} lookahead(Doc \rightarrow \{\text{IdentDoc IdentClerigos Evento}\}) &= First(\text{"\{"}) \\ &= \{\text{"\{"}\} \end{aligned}$$

- p4

$$\begin{aligned} lookahead(IdentDoc \rightarrow \text{Arq CotaDoc}\ldots) &= First(Arq) \\ &= \{str\} \end{aligned}$$

- p5

$$\begin{aligned} lookahead(IdentClerigos \rightarrow Clerigo) &= First(Clerigo) \\ &= First(Nome) \\ &= \{str\} \end{aligned}$$

- p6

$$\begin{aligned} lookahead(IdentClerigos \rightarrow \text{IdentClerigos ";" Clerigo}) &= First(IdentClerigos) \\ &= First(Clerigo) \bigcup First(IdentClerigos) \\ &= First(Nome) \\ &= \{str\} \end{aligned}$$

- p7

$$\begin{aligned} lookahead(Clerigo \rightarrow \text{Nome Apelido Diocese CategEcles Papel}) &= First(\text{Nome}) \\ &= \{str\} \end{aligned}$$

- p8

$$lookahead(Evento \rightarrow str) = \{str\}$$

- p9

$$lookahead(Arq \rightarrow str) \quad = \quad \{str\}$$

- p10

$$lookahead(Redactor \rightarrow str) \quad = \quad \{str\}$$

- p11

$$lookahead(Nome \rightarrow str) \quad = \quad \{str\}$$

- p12

$$lookahead(Apelido \rightarrow str) \quad = \quad \{str\}$$

- p13

$$lookahead(Diocese \rightarrow str) \quad = \quad \{str\}$$

- p14

$$lookahead(CotaDoc \rightarrow pal) \quad = \quad \{pal\}$$

- p15

$$lookahead(Tipo \rightarrow str) \quad = \quad \{pal\}$$

- p16

$$lookahead(Papel \rightarrow pal) \quad = \quad \{pal\}$$

- p17

$$lookahead(CategEcles \rightarrow BISPO) \quad = \quad \{BISPO\}$$

- p18

$$lookahead(CategEcles \rightarrow CONEGO) \quad = \quad \{CONEGO\}$$

- p19

$$lookahead(CategEcles \rightarrow PRESBITERO) \quad = \quad \{PREBISTERO\}$$

- p20

$$lookahead(DataRedaccao \rightarrow data) \quad = \quad \{data\}$$

- p21

$$lookahead(DataConsulta \rightarrow data) \quad = \quad \{data\}$$

A partir dos resultados obtidos no cálculo dos *lookahead*, podemos passar, então, à construção da tabela LL(1), seguindo sistematicamente o algoritmo recordado no dcumento anexo.
Obtém-se, assim, a tabela seguinte:

	{	}	;	str	pal	data	BISPO	CONEGO	PRESBITERO	$
Documentos	**p1/p2**									
Doc	p3									
IdentDoc				p4						
IdentClerigos				**p5/p6**						
Clerigo				p7						
Evento				p8						
Arq				p9						
Redactor				p10						
Nome				p11						
Apelido				p12						
Diocese				p13						
CotaDoc					p14					
Tipo					p15					
Papel					p16					
CategEcles							p17	p18	p19	
DataRedaccao						p20				
DataConsulta						p21				

Como podemos ver pelos resultados obtidos no cálculo dos *lookahead*, a gramática **não é LL(1)**, uma vez que:

- lookahead(p1) $\bigcap$ lookahead(p2) = { "{"}

- lookahead(p5) $\bigcap$ lookahead(p6) = { str }

Esta conclusão também está patente na tabela LL(1) acima, onde se identifica 1 conflito na linha relativa ao símbolo não-terminal *Documentos* (coluna "{") e 1 conflito na linha *IdentClerigos* (coluna *str*).

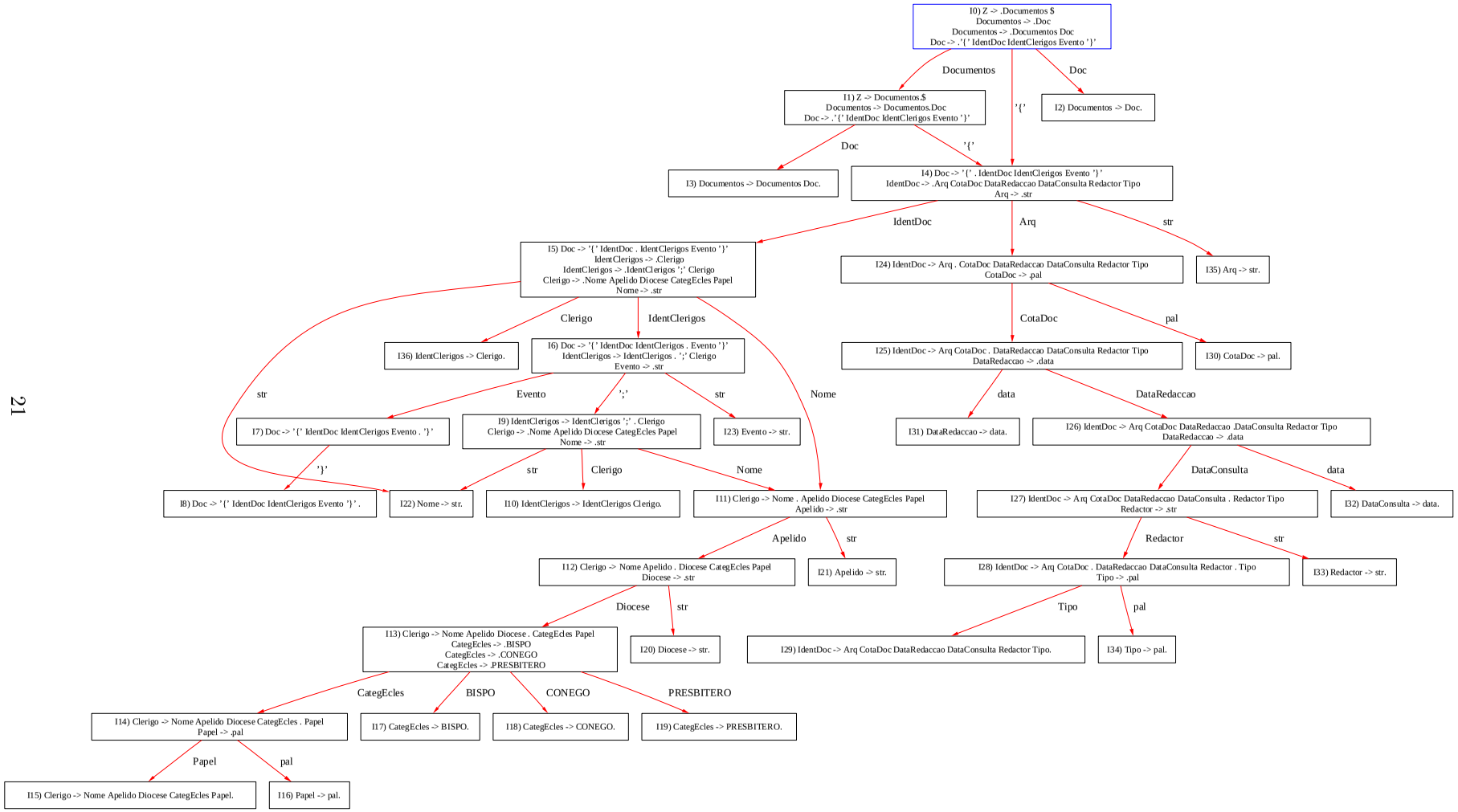
Resolução – 2.parte

Para obter a tabelas de decisão BU, ACTION e GOTO, tem de se começar por construir o autómato de reconhecimento (DFA) LR(0), seguindo o método sistemático recordado no documento anexo. Começando no estado 0 com o item

$$[Z \rightarrow \bullet \text{Documentos '\$'}]$$

obtemos o a DFA com 34 estados, que se mostra na figura 5.
A partir da função de transição, δ, do autómato LR(0) calculado, é possível derivar as tabelas de decisão BU cuja construção aqui não se inclui, ficando ao cuidado do leitor.

Olhando para o DFA da figura 5, é imediato concluir que a gramática em causa **é LR(0)**, pois não existem conflitos, nem de *redução/transição* nem de *redução/redução*, em nenhum dos estados do autómato.


I0) Z -> .Documentos $
Documentos -> .Doc
Documentos -> .Documentos Doc
Doc -> .'{' IdentDoc IdentClerigos Evento '}'
Documentos
Doc
'{'
I1) Z -> Documentos.$
Documentos -> Documentos.Doc
Doc -> .'{' IdentDoc IdentClerigos Evento '}'
I2) Documentos -> Doc.
Doc
'{'
I3) Documentos -> Documentos Doc.
I4) Doc -> '{' . IdentDoc IdentClerigos Evento '}'
IdentDoc -> .Arq CotaDoc DataRedaccao DataConsulta Redactor Tipo
Arq -> .str
IdentDoc
Arq
str
I5) Doc -> '{' IdentDoc . IdentClerigos Evento '}'
IdentClerigos -> .Clerigo
IdentClerigos -> .IdentClerigos ';' Clerigo
Clerigo -> .Nome Apelido Diocese CategEcles Papel
Nome -> .str
I24) IdentDoc -> Arq . CotaDoc DataRedaccao DataConsulta Redactor Tipo
CotaDoc -> .pal
I35) Arq -> str.
Clerigo
IdentClerigos
CotaDoc
pal
I36) IdentClerigos -> Clerigo.
I6) Doc -> '{' IdentDoc IdentClerigos . Evento '}'
IdentClerigos -> IdentClerigos . ';' Clerigo
Evento -> .str
I25) IdentDoc -> Arq CotaDoc . DataRedaccao DataConsulta Redactor Tipo
DataRedaccao -> .data
I30) CotaDoc -> pal.
str
Evento
';'
str
Nome
data
DataRedaccao
I7) Doc -> '{' IdentDoc IdentClerigos Evento . '}'
I9) IdentClerigos -> IdentClerigos ';' . Clerigo
Clerigo -> .Nome Apelido Diocese CategEcles Papel
Nome -> .str
I23) Evento -> str.
I31) DataRedaccao -> data.
I26) IdentDoc -> Arq CotaDoc DataRedaccao .DataConsulta Redactor Tipo
DataRedaccao -> .data
'}'
str
Clerigo
Nome
DataConsulta
data
I8) Doc -> '{' IdentDoc IdentClerigos Evento '}' .
I22) Nome -> str.
I10) IdentClerigos -> IdentClerigos Clerigo.
I11) Clerigo -> Nome . Apelido Diocese CategEcles Papel
Apelido -> .str
I27) IdentDoc -> Arq CotaDoc DataRedaccao DataConsulta . Redactor Tipo
Redactor -> .str
I32) DataConsulta -> data.
Apelido
str
Redactor
str
I12) Clerigo -> Nome Apelido . Diocese CategEcles Papel
Diocese -> .str
I21) Apelido -> str.
I28) IdentDoc -> Arq CotaDoc . DataRedaccao DataConsulta Redactor . Tipo
Tipo -> .pal
I33) Redactor -> str.
Diocese
str
Tipo
pal
I13) Clerigo -> Nome Apelido Diocese . CategEcles Papel
CategEcles -> .BISPO
CategEcles -> .CONEGO
CategEcles -> .PRESBITERO
I20) Diocese -> str.
I29) IdentDoc -> Arq CotaDoc DataRedaccao DataConsulta Redactor Tipo.
I34) Tipo -> pal.
CategEcles
BISPO
CONEGO
PRESBITERO
I14) Clerigo -> Nome Apelido Diocese CategEcles . Papel
Papel -> .pal
I17) CategEcles -> BISPO.
I18) CategEcles -> CONEGO.
I19) CategEcles -> PRESBITERO.
Papel
pal
I15) Clerigo -> Nome Apelido Diocese CategEcles Papel.
I16) Papel -> pal.