Lista de exercícios para estudos
Não vale ponto – Não precisa entregar

1. Considere a seguinte especificação léxica (*"Flex-like"*):

```
(01|10)              { print "morango" }
1(01)*0              { print "banana" }
(1011*0|0100*1)      { print "laranja" }
```

   Forneça uma entrada para esse analisador léxico de forma que a string de saída seja $(\texttt{morango}^3\texttt{banana})^3(\texttt{morangolaranja})^2$, onde $\texttt{A}^\texttt{i}$ denota `A` repetido `i` vezes. Você pode usar uma notação simplificada na sua resposta, por exemplo, $(\texttt{01})^3 = \texttt{010101}$. Lembre-se que a ordem das regras e tamanho da sequência de caracteres influenciam em qual regra será usada.

   $(0110011010)^3(01101110)^2$

2. Considere a seguinte especificação léxica (*"Flex-like"*):

```
c*b     { print "X" }
ac      { print "Y" }
c*ac*   { print "Z" }
```

   Forneça a saída produzida pelo analisador léxico construído a partir dessa especificação para a seguinte entrada:

```
cbaccacacccbbcccbaccac
```

   $XZYZXXXZY$

3. Forneça as GLCs para cada uma das seguintes linguagens. Qualquer gramática é aceitável (incluindo ambíguas), desde que estejam corretas.

   (a) o conjunto de todas as strings sobre o alfabeto $\Sigma = \{1, 2, -, *\}$ representando multiplicações entre inteiros em que a expressão gera algum valor positivo. Exemplos de strings nesta linguagem: $1 * 2$ ou $21 * -1 * -121$ ou $- - 222$. Exemplos de strings que não pertencem a esta linguagem: $2 * -2$ ou $1 * \lambda$ ou -12 ou $12 - 12$.

$$
\begin{aligned}
P &\to P * P \mid N * N \mid I \mid -N \\
N &\to N * P \mid P * N \mid - P \\
I &\to DI \mid D \\
D &\to 1 \mid 2
\end{aligned}
$$

(b) o conjunto de todas as strings sobre o alfabeto $\Sigma = \{0, 1\}$, em que o número de 1's é pelo menos duas vezes o número de 0's.

$$
\begin{aligned}
S \to\ & P0P1P0P \mid \\
& P1P0P1P \mid \\
& P1P1P0P \mid \\
& 1P \mid \lambda
\end{aligned}
$$

4. Considere as gramáticas a seguir, e faça o que se pede

(a) Faça a fatoração à esquerda da gramática:

$$
\begin{aligned}
S &\to S + S \mid S + P \\
P &\to P * P \mid P * I \\
I &\to -I \mid (S) \mid D \\
D &\to 0 \mid 1N \\
N &\to 0 \mid 1 \mid NN \mid \lambda
\end{aligned}
$$

$$
\begin{aligned}
S &\to S + S' \\
P &\to P * P' \\
I &\to -I \mid (S) \mid D \\
D &\to 0 \mid 1N \\
N &\to 0 \mid 1 \mid NN \mid \lambda \\
S' &\to S \mid P \\
P' &\to P \mid I
\end{aligned}
$$

(b) Identifique e elimine a recursão à esquerda da gramática:

$$
\begin{aligned}
S &\to S\ a\ S \mid U \\
U &\to U\ u\ U \mid T \\
T &\to t \mid f \mid T\ n \mid (S)
\end{aligned}
$$

$$
\begin{aligned}
S &\to U\ S' \\
U &\to T\ U' \\
T &\to t\ T' \mid f\ T' \mid (S)\ T' \\
S' &\to a\ S\ S' \mid \lambda \\
U' &\to u\ U\ U' \mid \lambda \\
T' &\to n\ T' \mid \lambda
\end{aligned}
$$

5. Considere a GLC a seguir, em que o conjunto de terminais é 0 , 1 , (,) , ;:

$$S \rightarrow (T$$
$$T \rightarrow CA\ |)$$
$$A \rightarrow ;B\ |)$$
$$B \rightarrow CA\ |)$$
$$C \rightarrow 0 \mid 1 \mid S$$

(a) Construa os conjuntos FIRST para cada um dos não terminais.

- FIRST(S): { (}
- FIRST(T): {) , 0 , 1 , (}
- FIRST(A): { ; ,) }
- FIRST(B): {) , 0 , 1 , (}
- FIRST(C): { 0 , 1 , (}

(b) Construa os conjuntos FOLLOW para cada um dos não terminais.

- FOLLOW(S): { \$, ; ,) }
- FOLLOW(T): { \$, ; ,) }
- FOLLOW(A): { \$, ; ,) }
- FOLLOW(B): { \$, ; ,) }
- FOLLOW(C): { ; ,) }

(c) Construa a *parsing table* LL(1) para a gramática.

Ñ-term.	(	)	;	0	1	$
S	(T					
T	CA	)		CA	CA	
A		)	;B			
B	CA	)		CA	CA	
C	S			0	1	

(d) Mostre a sequência da pilha, entrada e ações que ocorrem durante um *parsing* LL(1) da string "(( ) ; 0)". No começo do *parse*, a pilha deve conter um único S.

pilha	entrada	ação
S \$	(( ) ; 0) \$	$S \rightarrow (T$
(T \$	(( ) ; 0) \$	match (
T \$	( ) ; 0) \$	$A \rightarrow CA$
C A \$	( ) ; 0) \$	$C \rightarrow S$
S A \$	( ) ; 0) \$	$S \rightarrow (T$
(T A \$	( ) ; 0) \$	match (
T A \$	) ; 0) \$	$T \rightarrow)$
) A \$	) ; 0) \$	match)
A \$	; 0) \$	$A \rightarrow; B$
; B \$	; 0) \$	match ;
B \$	0) \$	$B \rightarrow CA$
C A \$	0) \$	$C \rightarrow 0$
0 A \$	0) \$	match 0
A \$	) \$	$A \rightarrow)$
) \$	) \$	match)
\$	\$	aceita!

6. Considere a GLC a seguir, em que o conjunto de terminais é x , y , z:

$$A \rightarrow xCBy$$
$$B \rightarrow z \mid \lambda$$
$$C \rightarrow y \mid Bx$$

   (a) Construa os conjuntos FIRST para cada um dos não terminais.

   - FIRST(A): { `x` }
   - FIRST(B): { `z` }
   - FIRST(C): { `x , y , z` }

   (b) Construa os conjuntos FOLLOW para cada um dos não terminais. Desconsidere mostrar o símbolo \$, pois não há uma "pseudo produção" inicial.

   - FOLLOW(A): { }
   - FOLLOW(B): { `x , y`}
   - FOLLOW(C): { `y , z`}

7. Para cada uma das gramáticas a seguir, identifique e demonstre se elas são ou não LL(1).

   (a)

$$X \rightarrow aY \mid Z$$
$$Y \rightarrow a \mid c$$
$$Z \rightarrow bY$$

   R: A gramática é LL(1).

(b)

$$P \to dR$$
$$R \to o \mid S$$
$$S \to g \mid og$$

A gramática não é LL(1).
O FIRST(R) contém o FIRST(o)=o, e FIRST(R) contém FIRST(S)=o,g, logo, representa uma interseção não vazia entre duas possíveis regras a partir de um mesmo símbolo não terminal.

(c)

$$J \to aKL$$
$$K \to c \mid \lambda$$
$$L \to c$$

A gramática não é LL(1).
K gera lambda, então deve-se observar o FOLLOW(K). FIRST(K)=FIRST(c) e FOLLOW(K) (pela regra aKL, sendo K vazio) são c e c, uma interseção não vazia.

(d)

$$J \to aKL$$
$$K \to c \mid \lambda$$
$$L \to b$$

A gramática é LL(1).
FIRST(K)=FIRST(c)=ce e FOLLOW(K)=b, logo, não temos o problema anterior da interseção.

8. A gramática a seguir não é LL(1). Use o processo necessário para transformar esta gramática, de forma que ela produza uma linguagem equivalente, e satisfaça as condições para ser uma LL(1). Será necessário remover as recursões indiretas à esquerda, para depois remover as recursões diretas à esquerda.

$$A \to B! \mid x$$
$$B \to C$$
$$C \to A? \mid y$$

1) Eliminando a recursão indireta à esquerda. Substituir B por C na primeira regra:

$$A \to C! \mid x$$
$$B \to C$$
$$C \to A? \mid y$$

2) Eliminando regra de B (inalcançável) e substituindo C:

$$A \to A?! \mid y! \mid x$$
$$C \to A? \mid y$$

3) Eliminando regra de C (inalcançável), e eliminando a recursão direta à esquerda em A:

$$A \rightarrow y!A' \mid xA'$$
$$A' \rightarrow ?!A' \mid \lambda$$