Universidade Federal da Fronteira Sul – UFFS
Campus Chapecó
Ciência da Computação

Lista de exercícios

1. Para as GLCs a seguir, dados os conjuntos de itens válidos, first e follow, construa a tabela SLR e realize o reconhecimento da fita de entrada *fornecida.*

(a)

0- D' ::= D
1- D ::= T L ; D
2- D ::= T L ;
3- T ::= int
4- T ::= real
5- L ::= id
6- L ::= id , L

	FIRST	FOLLOW
D	int,real	$
T	int,real	id
L	id	;

I_0={D' -> .D; D -> .T L ; D; D -> .T L ;; T -> .int; T -> .real}
I_1={D' -> D.}
I_2={D -> T.L ; D; D -> T.L ;; L -> .id; L -> .id , L}
I_3={T -> int.}
I_4={T -> real.}
I_5={D -> T L.; D; D -> T L.;}
I_6={L -> id.; L -> id., L}
I_7={D -> T L ;.D; D -> T L ;.; D -> .T L ; D; D -> .T L ;; T -> .int; T -> .real}
I_8={L -> id ,.L; L -> .id; L -> .id , L}
I_9={D -> T L ; D.}
I_{10}={L -> id , L.}

	;	int	real	id	,	$	D	T	L
0									
1									
2									
3									
4									
5									
6									
7									
8									
9									
10									

Execute a análise sintática da fita de entrada a seguir:

0	int a ; real b ; $

(b)

0- S' ::= S
1- S ::= while true do { A }
2- A ::= S
3- A ::= exp

	FIRST	FOLLOW
S	while	$,}
A	while,exp	}

I_0={S' -> .S; S -> .while true do { A }}
I_1={S' -> S.}
I_2={S -> while.true do { A }}
I_3={S -> while true.do { A }}
I_4={S -> while true do.{ A }}
I_5={S -> while true do {.A }; A -> .S; A -> .exp; S -> .while true do { A }}
I_6={S -> while true do { A.}}
I_7={A -> S.}
I_8={A -> exp.}
I_9={S -> while true do { A }.}

	while	true	do	{	}	exp	$	S	A
0									
1									
2									
3									
4									
5									
6									
7									
8									
9									

Execute a análise sintática da fita de entrada a seguir:

0	while true do { exp } $

2. Para as GLCs a seguir, construa um esquema de tradução (ações semânticas) para fornecer, durante a análise do código escrito pelo programador, as informações como descrito para cada glc.

Esquema de tradução para fornecer, como resultado, o número de variáveis declaradas como int e o número de variáveis declaradas como real.

D ::= T L ; D	
D ::= T L ;	
T ::= int	
T ::= real	
L ::= id	
L ::= id , L	

Esquema de tradução para informar, como resultado, o número de ocorrências aninhadas de while. Por exemplo, em while true do { exp } são zero ocorrências. Em while true do { **while** true do { exp } } uma ocorrência.

S ::= while true do { A }	
A ::= S	
A ::= exp	

3. **Utilizando a GLC e respectivo esquema de tradução dado, construa as árvores de derivação anotadas para as expressões dadas. Ao final da redução, S.cod deve conter o código intermediário (operações básicas de três endereços) da expressão analisada.**

S → id := E	{ S.cod = E.cod ‖ geracod (id.nome ":=" E.nome)}
E → E1 + E2	{ E.nome = geratemp; E.cod = E1.cod ‖ E2.cod ‖ geracod (E.nome ":=" E1.nome "+" E2.nome) }
E → E1 * E2	{ E.nome = geratemp; E.cod = E1.cod ‖ E2.cod ‖ geracod (E.nome ":=" E1.nome "*" E2.nome)}
E → (E1)	{ E.nome = E1.nome; E.cod = E1.cod }
E → id	{ E.nome = id.nome; E.cod = " " }

(a) x = a + (b * (c + d)) * c

(b) x = ((a + b * (b + c * d)) + c) * d

4. **Decomponha as expressões abaixo em código de operações básicas (intermediário), construa o GAD para o código intermediário e gere a sequência otimizada de execução (conjunto L).**

(a) x = a + (b * (c + d)) * c

(a) x = ((a + b * (b + c * d)) + c) * d