Universidade Federal de Santa Catarina
INE5622 – Introdução à Compiladores
Alunos: Lucas Pian Rodrigues, Marcelo de Oliveira, Sandro Santana Ribeiro, Vinícius Dias de Paula Ramos.

Trabalho Parte 2 – First and Follow

1. Transformações Realizadas

a) De: FLIST → FDEF FLIST | FDEF

Para: FLIST → FDEF FLIST'
FLIST' → FDEF FLIST' | ε
Adição de um novo não-terminal afim de eliminar a recursividade à esquerda.

b) De: PARLIST → int id, PARLIST | int id | ε

Para: PARLIST → int id PARLIST'
PARLIST' → , int id PARLIST' | ε

Adição de um novo não-terminal afim de resolver conflito de prefixo comum, neste caso int id.

c) De: VARLIST → id, VARLIST | id
Para: VARLIST → id VARLIST'
VARLIST' → , id VARLIST' | ε

Adição de um novo não-terminal afim de resolver conflito de prefixo comum, neste caso id e resolver a recursividade à esquerda.

d) De: STMT → int VARLIST;
| ATRIBST;
| PRINTST;
| RETURNST;
| IFSTMT
| {STMTLIST}
| ;

Para:STMT → M_STMT
M_STMT → if (EXPR) M_STMT else M_STMT
| BASIC_STMT
BASIC_STMT → int VARLIST;
| id := EXPR;
| print EXPR;
| return RETURNST';
| {STMTLIST}
| ;
RETURNST' → EXPR | ε

Adição de um novo não-terminal para separar comandos compostos (if) de stmts básicos.

e) De: STMTLIST → STMT STMTLIST | STMT

Para: STMTLIST → STMT STMTLIST'
STMTLIST' → STMT STMTLIST' | ε

Adição de um novo não-terminal afim de eliminar a recursividade à esquerda.

f) De: EXPR → NUMEXPR < NUMEXPR
| NUMEXPR <= NUMEXPR
| NUMEXPR > NUMEXPR
| NUMEXPR >= NUMEXPR
| NUMEXPR == NUMEXPR
| NUMEXPR <> NUMEXPR
| NUMEXPR

Para: EXPR → NUMEXPR EXPR'
EXPR' → RELOP NUMEXPR | ε
RELOP → < | <= | > | >= | == | <>

Adição de um novo não-terminal afim de resolver conflito de prefixo comum, neste caso NUMEXPR.

g) De: NUMEXPR → NUMEXPR + TERM
| NUMEXPR - TERM
| TERM

Para: NUMEXPR → TERM NUMEXPR'
NUMEXPR' → + TERM NUMEXPR'
| - TERM NUMEXPR'
| ε

Adição de um novo não-terminal afim de eliminar a recursividade à esquerda.

h) De: TERM → TERM * FACTOR
| TERM / FACTOR
| FACTOR

Para: TERM → FACTOR TERM'
TERM' → * FACTOR TERM' | / FACTOR TERM' | ε

Adição de um novo não-terminal afim de eliminar a recursividade à esquerda.

i) De: FACTOR → num
| (NUMEXPR)
| id

Para: FACTOR → num
| (NUMEXPR)
| id FACTOR'
FACTOR' → (ARGLIST) | ε

Adição de um novo não-terminal afim de resolver problema de ambiguidade entre chamada de função e chamada de variável simples.

j) De: ARGLIST → EXPR, ARGLIST | EXPR | ε

Para: ARGLIST → EXPR ARGLIST'
ARGLIST' → , EXPR ARGLIST' | ε

Adição de um novo não-terminal afim de resolver conflito de prefixo comum, neste caso EXPR.

2. Gramática LSI-2024-2 Final, transformada em LL(1):

MAIN → STMT | FLIST | ε
FLIST → FDEF FLIST'
FLIST' → FDEF FLIST' | ε

FDEF → def id (PARLIST) { STMTLIST }

PARLIST → int id PARLIST'
PARLIST' → , PARLIST’ | ε

VARLIST → id VARLIST'
VARLIST' → , VARLIST’ | ε

STMT → M_STMT
M_STMT → if (EXPR) M_STMT else M_STMT
| BASIC_STMT

BASIC_STMT → int VARLIST;
| id := EXPR;
| print EXPR;
| return RETURNST';
| {STMTLIST}
| ;
RETURNST' → EXPR | ε

STMTLIST → STMT STMTLIST'
STMTLIST' → STMT STMTLIST' | ε

EXPR → NUMEXPR EXPR'
EXPR' → RELOP NUMEXPR | ε
RELOP → < | <= | > | >= | == | <>

NUMEXPR → TERM NUMEXPR'
NUMEXPR' → + TERM NUMEXPR'
| - TERM NUMEXPR'
| ε

TERM → FACTOR TERM'
TERM' → * FACTOR TERM'
| / FACTOR TERM'
| ε

FACTOR → num
| (NUMEXPR)
| id FACTOR'
FACTOR' → (ARGLIST) | ε

ARGLIST → EXPR ARGLIST' | ε
ARGLIST' → , EXPR ARGLIST' | ε

NOTA: Separação e numeração das produções para auxiliar na identificação das tabelas, conforme ferramenta de parser disponibilizada no moodle.

0) S :== MAIN $

1) MAIN ::= FLIST
2) MAIN ::= STMTLIST
3) MAIN ::= ''

4) FLIST ::= FDEF FLIST'
5) FLIST' ::= FDEF FLIST'
6) FLIST' ::= ''

7) FDEF ::= def id (PARLIST) { STMTLIST }

8) PARLIST ::= int id PARLIST'

9) PARLIST ::= ''

10) PARLIST' ::= , int id PARLIST'

11) PARLIST' ::= ''

12) VARLIST ::= id VARLIST'

13) VARLIST' ::= , id VARLIST'

14) VARLIST' ::= ''

15) STMT ::= M_STMT

16) M_STMT ::= if (EXPR) M_STMT else M_STMT

17) M_STMT ::= BASIC_STMT

18) BASIC_STMT ::= int VARLIST ;

19) BASIC_STMT ::= id := EXPR ;

20) BASIC_STMT ::= print EXPR ;

21) BASIC_STMT ::= return RETURNST' ;

22) BASIC_STMT ::= { STMTLIST }

23) BASIC_STMT ::= ;

24) RETURNST' ::= EXPR

25) RETURNST' ::= ''

26) STMTLIST ::= STMT STMTLIST'

27) STMTLIST' ::= STMT STMTLIST'

28) STMTLIST' ::= ''

29) EXPR ::= NUMEXPR EXPR'

30) EXPR' ::= RELOP NUMEXPR

31) EXPR' ::= ''

32) RELOP ::= <

33) RELOP ::= <=

34) RELOP ::= >

35) RELOP ::= >=

36) RELOP ::= ==

37) RELOP ::= <>

38) NUMEXPR ::= TERM NUMEXPR'

39) NUMEXPR' ::= + TERM NUMEXPR'

40) NUMEXPR' ::= - TERM NUMEXPR'

41) NUMEXPR' ::= ''

42) TERM ::= FACTOR TERM'

43) TERM' ::= * FACTOR TERM'

44) TERM' ::= / FACTOR TERM'

45) TERM' ::= ''

46) FACTOR ::= num

47) FACTOR ::= (NUMEXPR)

48) FACTOR ::= id FACTOR'

49) FACTOR' ::= (ARGLIST)

50) FACTOR' ::= ''

51) ARGLIST ::= EXPR ARGLIST'

52) ARGLIST ::= ''

53) ARGLIST' ::= , EXPR ARGLIST'

54) ARGLIST' ::= ''

3. Conjuntos First e Follow:

Não-Terminais	First	Follow
S	$, def, if, int, id, print, return, {, ;	
MAIN	def, if, int, id, print, return, {, ;	$
FLIST	def	$
FLIST'	def	$
FDEF	def	def, $
PARLIST	int	)
PARLIST'	,	)
VARLIST	id	;
VARLIST'	,	;
STMT	if, int, id, print, return, {, ;	}, if, int, id, print, return, {, ;, $
M_STMT	if, int, id, print, return, {, ;	else, }, if, int, id, print, return, {, ;, $
BASIC_STMT	int, id, print, return, {, ;	else, }, if, int, id, print, return, {, ;, $
RETURNST'	num, (, id	;
STMTLIST	if, int, id, print, return, {, ;	}, $
STMTLIST'	if, int, id, print, return, {, ;	}, $
EXPR	num, (, id	), ;, ,
EXPR'	<, <=, >, >=, ==, <>	), ;, ,
RELOP	<, <=, >, >=, ==, <>	num, (, id
NUMEXPR	num, (, id	), ;, <, <=, >, >=, ==, <>, ,
NUMEXPR'	+, -	), ;, <, <=, >, >=, ==, <>, ,
TERM	num, (, id	), ;, <, <=, >, >=, ==, <>, +, -, ,
TERM'	*, /	), ;, <, <=, >, >=, ==, <>, +, -, ,
FACTOR	num, (, id	), ;, <, <=, >, >=, ==, <>, +, -, *, /, ,
FACTOR'	(	), ;, <, <=, >, >=, ==, <>, +, -, *, /, ,
ARGLIST	num, (, id	)
ARGLIST'	,	)

4. Tabela de Reconhecimento Sintático (baseada em referências para as produções expandidas):

Não-Terminais	$	def	id	(	)	{	}	int	,	if	else	;	:=	print	return	<	<=	>	>=	==	<>	+	-	*	/	num
S	0	0	0			0		0		0		0		0	0											
MAIN	3	1	2			2		2		2		2		2	2											
FLIST		4																								
FLIST'	6	5																								
FDEF		7																								
PARLIST					9			8																		
PARLIST'					11				10																	
VARLIST			12																							
VARLIST'									13			14														
STMT			15			15		15		15		15		15	15											
M_STMT			17			17		17		16		17		17	17											
BASIC_STMT			19			22		18				23		20	21											
RETURNST'			24	24								25														24
STMTLIST			26			26		26		26		26		26	26											
STMTLIST'	28		27			27	28	27		27		27		27	27											
EXPR			29	29																						29
EXPR'					31				31			31				30	30	30	30	30	30					
RELOP																32	33	34	35	36	37					
NUMEXPR			38	38																						38
NUMEXPR'					41				41			41				41	41	41	41	41	41	39	40			
TERM			42	42																						42
TERM'					45				45			45				45	45	45	45	45	45	45	45	43	44	
FACTOR			48	47																						46
FACTOR'				49	50				50			50				50	50	50	50	50	50	50	50	50	50	
ARGLIST			51	51	52																					51
ARGLIST'					54				53																	

4.1 Tabelas de reconhecimento no formato original:

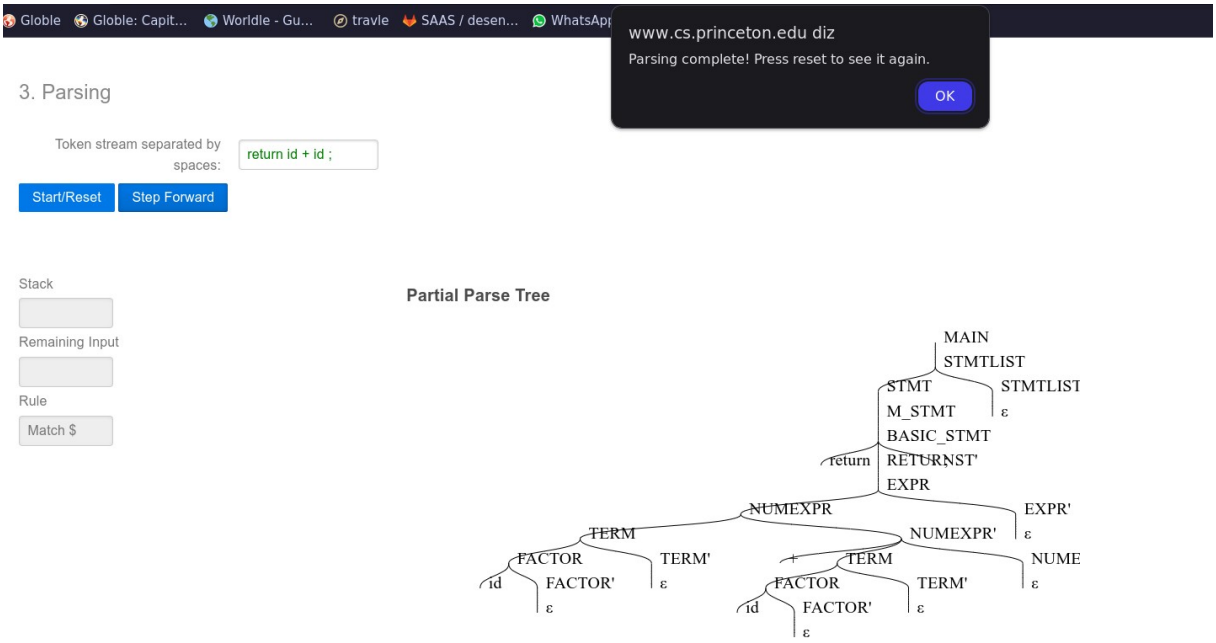
}	int	,	if	else	;	:=	print	return	<
	S ::= MAIN $		S ::= MAIN $		S ::= MAIN $		S ::= MAIN $	S ::= MAIN $	
	MAIN ::= STMTLIST		MAIN ::= STMTLIST		MAIN ::= STMTLIST		MAIN ::= STMTLIST	MAIN ::= STMTLIST	
	PARLIST ::= int id PARLIST'								
		PARLIST' ::= , int id PARLIST'							
		VARLIST' ::= , id VARLIST'			VARLIST' ::= ε				
	STMT ::= M_STMT		STMT ::= M_STMT		STMT ::= M_STMT		STMT ::= M_STMT	STMT ::= M_STMT	
	M_STMT ::= BASIC_STMT		M_STMT ::= if (EXPR) M_STMT else M_STMT		M_STMT ::= BASIC_STMT		M_STMT ::= BASIC_STMT	M_STMT ::= BASIC_STMT	
	BASIC_STMT ::= int VARLIST ;				BASIC_STMT ::= ;		BASIC_STMT ::= print EXPR ;	BASIC_STMT ::= return RETURNST' ;	
					RETURNST' ::= ε				
	STMTLIST ::= STMT STMTLIST'		STMTLIST ::= STMT STMTLIST'		STMTLIST ::= STMT STMTLIST'		STMTLIST ::= STMT STMTLIST'	STMTLIST ::= STMT STMTLIST'	
STMTLIST' ::= ε	STMTLIST' ::= STMT STMTLIST'		STMTLIST' ::= STMT STMTLIST'		STMTLIST' ::= STMT STMTLIST'		STMTLIST' ::= STMT STMTLIST'	STMTLIST' ::= STMT STMTLIST'	
		EXPR' ::= ε			EXPR' ::= ε				EXPR' ::= RELOP NUMEXPR
									RELOP ::= <
		NUMEXPR' ::= ε			NUMEXPR' ::= ε				NUMEXPR' ::= ε
		TERM' ::= ε			TERM' ::= ε				TERM' ::= ε
		FACTOR' ::= ε			FACTOR' ::= ε				FACTOR' ::= ε
		ARGLIST' ::= , EXPR ARGLIST'							

	$	def	id	(	)	{
S	S ::= MAIN $	S ::= MAIN $	S ::= MAIN $			S ::= MAIN $
MAIN	MAIN ::= ε	MAIN ::= FLIST	MAIN ::= STMTLIST			MAIN ::= STMTLIST
FLIST		FLIST ::= FDEF FLIST'				
FLIST'	FLIST' ::= ε	FLIST' ::= FDEF FLIST'				
FDEF		FDEF ::= def id (PARLIST) { STMTLIST }				
PARLIST					PARLIST ::= ε	
PARLIST'					PARLIST' ::= ε	
VARLIST			VARLIST ::= id VARLIST'			
VARLIST'						
STMT			STMT ::= M_STMT			STMT ::= M_STMT
M_STMT			M_STMT ::= BASIC_STMT			M_STMT ::= BASIC_STMT
BASIC_STMT			BASIC_STMT ::= id := EXPR ;			BASIC_STMT ::= { STMTLIST }
RETURNST'			RETURNST' ::= EXPR	RETURNST' ::= EXPR		
STMTLIST			STMTLIST ::= STMT STMTLIST'			STMTLIST ::= STMT STMTLIST'
STMTLIST'	STMTLIST' ::= ε		STMTLIST' ::= STMT STMTLIST'			STMTLIST' ::= STMT STMTLIST'
EXPR			EXPR ::= NUMEXPR EXPR'	EXPR ::= NUMEXPR EXPR'		
EXPR'					EXPR' ::= ε	
RELOP						
NUMEXPR			NUMEXPR ::= TERM NUMEXPR'	NUMEXPR ::= TERM NUMEXPR'		
NUMEXPR'					NUMEXPR' ::= ε	
TERM			TERM ::= FACTOR TERM'	TERM ::= FACTOR TERM'		
TERM'					TERM' ::= ε	
FACTOR			FACTOR ::= id FACTOR'	FACTOR ::= (NUMEXPR)		
FACTOR'				FACTOR' ::= (ARGLIST)	FACTOR' ::= ε	
ARGLIST			ARGLIST ::= EXPR ARGLIST'	ARGLIST ::= EXPR ARGLIST'	ARGLIST ::= ε	
ARGLIST'					ARGLIST' ::= ε	

<=	>	>=	==	<>	+	-	*	/
EXPR' ::= RELOP NUMEXPR	EXPR' ::= RELOP NUMEXPR	EXPR' ::= RELOP NUMEXPR	EXPR' ::= RELOP NUMEXPR	EXPR' ::= RELOP NUMEXPR				
RELOP ::= <=	RELOP ::= >	RELOP ::= >=	RELOP ::= ==	RELOP ::= <>				
NUMEXPR' ::= ε	NUMEXPR' ::= ε	NUMEXPR' ::= ε	NUMEXPR' ::= ε	NUMEXPR' ::= ε	NUMEXPR' ::= + TERM NUMEXPR'	NUMEXPR' ::= - TERM NUMEXPR'		
TERM' ::= ε	TERM' ::= ε	TERM' ::= ε	TERM' ::= ε	TERM' ::= ε	TERM' ::= ε	TERM' ::= ε	TERM' ::= * FACTOR TERM'	TERM' ::= / FACTOR TERM'
FACTOR' ::= ε	FACTOR' ::= ε	FACTOR' ::= ε	FACTOR' ::= ε	FACTOR' ::= ε	FACTOR' ::= ε	FACTOR' ::= ε	FACTOR' ::= ε	FACTOR' ::= ε

num
RETURNST' ::= EXPR
EXPR ::= NUMEXPR EXPR'
NUMEXPR ::= TERM NUMEXPR'
TERM ::= FACTOR TERM'
FACTOR ::= num
ARGLIST ::= EXPR ARGLIST'

5 Exemplo


Globle
Globle: Capit...
Worldle - Gu...
travle
SAAS / desen...
WhatsApp
www.cs.princeton.edu diz
Parsing complete! Press reset to see it again.
OK
3. Parsing
Token stream separated by spaces:
return id + id ;
Start/Reset
Step Forward
Stack
Remaining Input
Rule
Match $
Partial Parse Tree
MAIN
STMTLIST
STMT
STMTLIST
M_STMT
ε
BASIC_STMT
return
RETURNST'
EXPR
NUMEXPR
EXPR'
TERM
NUMEXPR'
ε
FACTOR
TERM'
+
TERM
NUME
id
FACTOR'
ε
FACTOR
TERM'
ε
ε
id
FACTOR'
ε
ε