UNIVERSIDADE FEDERAL DE OURO PRETO – UFOP

CONSTRUÇÃO DE COMPILADORES 2018.2

TRABALHO PRÁTICO 2 - ANÁLISE SINTÁTICA

MARINA DE SOUZA MENDES – 15.1.5978

Gramática

#Regra	Production rule	Internal representation
1	*Program* → *Funs*	Program
2	*Funs* → *Fun*	Sequence of Function definitions
3	*Funs* → *Fun Funs*	
4	*Fun* → *TypeId* `(` *TypeIds* `) =` *Exp*	Function definition
5	*TypeId* → `bool id`	Type and identifier
6	*TypeId* → `int id`	
7	*TypeIds* → *TypeId*	Sequence of types and identifiers
8	*TypeIds* → *TypeId* , *TypeIds*	
9	*Exp* → `num`	Expressions
10	*Exp* → `id`	
11	*Exp* → *Exp* `+` *Exp*	
12	*Exp* → *Exp* `<` *Exp*	
13	*Exp* → `id(`*Exps*`)`	
14	*Exp* → `if` *Exp* `then` *Exp* `else` *Exp*	

#Regra	Production rule	Internal representation
15	*Exp* → `let id =` *Exp* `in` *Exp*	
16	*Exps* → *Exp*	Sequence of expressions
17	*Exps* → *Exp* `,` *Exps*	

1) Reescrever a gramática dada (se necessário) para satisfazer as condições de uma gramática LL(1)

Para que a gramática satisfaça as condições de uma gramática LL(1) ela não deve conter ambiguidades, logo, a recursividade à esquerda deve ser removida. As regras 11 e 12 possuem recursão à esquerda e ao removê-las novas regras são inseridas. A nova gramática, após a remoção de recursão à esquerda, é a seguinte:

#Regra	Production rule	Internal representation
1	*Program* → *Funs*	Program
2	*Funs* → *Fun*	Sequence of Function definitions
3	*Funs* → *Fun Funs*	
4	*Fun* → *TypeId* `(` *TypeIds* `) =` *Exp*	Function definition
5	*TypeId* → `bool id`	Type and identifier
6	*TypeId* → `int id`	
7	*TypeIds* → *TypeId*	Sequence of types and identifiers
8	*TypeIds* → *TypeId* `,` *TypeIds*	
9	*Exp* → *A Exp'*	
10	Exp' → + A	
11	Exp' → ε	
12	*A* → *B A'*	

#Regra	Production rule	Internal representation
13	A’ → < B A’	
14	A’ → ε	
15	*B* → `num`	Expressions
16	*B* → `id`	
17	*B* → `id (` *Exps* `)`	
18	*B* → `if` *B* `then` *B* `else` *B*	
19	*B* → `let id =` *B* `in` *B*	
20	*Exps* → *Exp*	Sequence of expressions
21	*Exps* → *Exp* `,` *Exps*	

Nesta gramática há ainda muitas regras que podem ser fatoradas para que a gramática seja LL(1) (regras 2, 3, 7, 8, 16, 17, 20 e 21). Após a fatoração, a gramática é a seguinte:

#Regra	Production rule	Internal representation
1	*Program* → *Funs*	Program
2	*Funs* → *Fun Funs’*	Sequence of Function definitions
3	*Funs’* → Funs	
4	*Funs’* → ε	
5	*Fun* → *TypeId* `(` *TypeIds* `) =` *Exp*	Function definition
6	*TypeId* → `bool id`	Type and identifier
7	*TypeId* → `int id`	
8	*TypeIds* → *TypeId TypeIds’*	Sequence of types and identifiers
9	*TypeIds’* → , *TypeIds*	
10	*TypeIds’* → ε	
11	*Exp* → *A Exp’*	

#Regra	Production rule	Internal representation
12	Exp’ → + A	
13	Exp’ → ε	
14	*A* → *B A’*	
15	A’ → < B A’	
16	A’ → ε	
17	*B* → `num`	Expressions
18	*B* → `id B’`	
19	B’ → `(` *Exps* `)`	
20	*B’* → ε	
21	*B* → `if` *B* `then` *B* `else` *B*	
22	*B* → `let id =` *B* `in` *B*	
23	*Exps* → *Exp Exps’*	Sequence of expressions
24	*Exps’* → `,` *Exps*	
25	*Exps’* → ε	

2) Calcular os conjuntos First e Follow da gramática para construir a tabela LL(1).

Não terminais	Nullable	First	Follow
Program	F	bool, int	
Funs	F	bool, int	$
Funs’	V	bool, int	$
Fun	F	bool, int	bool, int, $
TypeId	F	bool, int	,, (,)
TypeIds	F	bool, int	)
TypeIds’	V	,	)
Exp	F	num, id, if, let	,, bool, int,), $
Exp’	V	+	,, bool, int,), $
Exps	F	num, id, if, let	)
Exps’	V	,	)
A	F	num, id, if, let	+, ,, bool, int,), $
A’	V	<	+, ,, bool, int,), $

B	F	num, id, if, let	then, else, in, <, +,), bool, int, ,, $
B’	V	(	then, else, in, <, +,), bool, int, ,, $

3) Construir a tabela LL(1).

	(	)	=	,	bool	int	id	+	<	num	if	then	else	let	in	$
Program					1	1										
Funs					2	2										
Funs’					3	3										
Fun					5	5										
TypeId					6	7										
TypeIds					8	8										
TypeIds’		10		9												
Exp							11			11	11			11		
Exp’		13		13	13	13		12								13
Exps							23			23	23			23		
Exps’		25		24												
A							14			14	14			14		
A’		16		16	16	16		16	15							16
B							18			17	21			22		
B’	19	20		20	20	20		20	20			20	20		20	20

A tabela LL(1) não possui conflitos, portanto a gramática é LL(1).