

项目二测试报告

一、测试项目名称

项目二：语法分析和语法树生成测试

二、测试内容

对 TINY 和 mini-c 语言的 BNF 文法进行语法分析，并根据单词编码生成分析表和语法树。

三、测试用例

- 1) mini-c 语法.txt
- 2) mini-c.lex
- 3) TINY 语法.txt
- 4) TINY.lex

四、测试结果

由于状态过多，分析表过长，部分板块无法展示完全，因此只展示部分，若要查看所有信息，可通过可执行程序进行测试。

4.1 First 集 Follow 集

TINY

SLR Analyzer

文法规则输入

PROGRAM -> STMT-SEQUENCE
STMT-SEQUENCE -> STMT-SEQUENCE STATEMENT | STATEMENT
STATEMENT -> IF-STMT | REPEAT-STMT | ASSIGN-STMT | READ-STMT | WRITE-STMT
IF-STMT -> if EXP then STMT-SEQUENCE end | if EXP then STMT-SEQUENCE else STMT-SEQUENCE end
REPEAT-STMT -> repeat STMT-SEQUENCE until EXP
ASSIGN-STMT -> identifier := EXP
READ-STMT -> read identifier
WRITE-STMT -> write EXP
EXP -> SIMPLE-EXP COMPARISON-OP SIMPLE-EXP | SIMPLE-EXP
COMPARISON-OP -> < | = | <= | <> | >= | >
SIMPLE-EXP -> SIMPLE-EXP ADDOP TERM | TERM
ADDOP -> + | -
TERM -> TERM MULOP FACTOR | FACTOR
MULOP -> * | / | %
FACTOR -> (EXP) | number | identifier

错误报告

是否为SLR(1): ☒ 是

编码分析输入

First Follow

非终结符First集

1	2
ADDOP	+ -
ASSIGN-STMT	identifier
COMPARISON-OP	< <= <> > >=
EXP	(identifier number
FACTOR	(identifier number
IF-STMT	if
MULOP	% * /
PROGRAM	identifier if read repeat write
READ-STMT	read
REPEAT-STMT	repeat

非终结符Follow集

1	2
ADDOP	(identifier number
ASSIGN-STMT	\$; else end until
COMPARISON-OP	(identifier number
EXP	\$) ; else end then until
FACTOR	\$ %) * + - / ; < <= <> > >= else end then until
IF-STMT	\$; else end until
MULOP	(identifier number
PROGRAM	\$
READ-STMT	\$; else end until
REPEAT-STMT	\$; else end until

SLR Analyzer

文法规则输入

PROGRAM -> STMT-SEQUENCE
STMT-SEQUENCE -> STMT-SEQUENCE STATEMENT | STATEMENT
STATEMENT -> IF-STMT | REPEAT-STMT | ASSIGN-STMT | READ-STMT | WRITE-STMT
IF-STMT -> if EXP then STMT-SEQUENCE end | if EXP then STMT-SEQUENCE else STMT-SEQUENCE end
REPEAT-STMT -> repeat STMT-SEQUENCE until EXP
ASSIGN-STMT -> identifier := EXP
READ-STMT -> read identifier
WRITE-STMT -> write EXP
EXP -> SIMPLE-EXP COMPARISON-OP SIMPLE-EXP | SIMPLE-EXP
COMPARISON-OP -> < | = | <= | <> | >= | >
SIMPLE-EXP -> SIMPLE-EXP ADDOP TERM | TERM
ADDOP -> + | -
TERM -> TERM MULOP FACTOR | FACTOR
MULOP -> * | / | %
FACTOR -> (EXP) | number | identifier

错误报告

是否为SLR(1): ☒ 是

编码分析输入

First Follow

非终结符First集

1	2
IF-STMT	if
MULOP	% * /
PROGRAM	identifier if read repeat write
READ-STMT	read
REPEAT-STMT	repeat
SIMPLE-EXP	(identifier number
STATEMENT	identifier if read repeat write
STMT-SEQUENCE	identifier if read repeat write
TERM	(identifier number
WRITE-STMT	write

非终结符Follow集

1	2
IF-STMT	\$; else end until
MULOP	(identifier number
PROGRAM	\$
READ-STMT	\$; else end until
REPEAT-STMT	\$; else end until
SIMPLE-EXP	\$) + - ; < <= <> > >= else end then until
STATEMENT	\$; else end until
STMT-SEQUENCE	\$; else end until
TERM	\$ %) * + - / ; < <= <> > >= else end then until
WRITE-STMT	\$; else end until

mini-c

SLR Analyzer

文法规则输入

导入文件 保存文法 SLR(1) 分析

```
CONTEXTS: SINT -> { LOCAL-DEFINITIONS STATEMENTS } LIST ;
LOCAL-DEFINITIONS -> LOCAL-DEFINITIONS VARIABLE-DEFINITION | empty
STATEMENT-LIST -> STATEMENT-LIST STATEMENT | empty
STATEMENT -> EXPRESSION-STM | COMPOUND-STM | CONDITION-STM | DOWHILE-STM | RETURN-STM
EXPRESSION-STM -> EXPRESSION ;
CONDITION-STM -> if ( EXPRESSION ) STATEMENT | if ( EXPRESSION ) STATEMENT else STATEMENT
DOWHILE-STM -> do STATEMENT while ( EXPRESSION ) ;
RETURN-STM -> return ; | return EXPRESSION ;
EXPRESSION -> VARIABLE = EXPRESSION | SIMPLE-EXPRESSION
VARIABLE -> id | id [ EXPRESSION ]
SIMPLE-EXPRESSION -> ADDITIVE-EXPRESSION RELOP ADDITIVE-EXPRESSION | ADDITIVE-EXPRESSION
RELOP -> <= | < | > | >= | == | !=
ADDITIVE-EXPRESSION -> ADDITIVE-EXPRESSION ADOP TERM | TERM
ADOP -> + | -
TERM -> TERM MULOP FACTOR | FACTOR
MULOP -> * | / | %
FACTOR -> ( EXPRESSION ) | VARIABLE | CALL | num
CALL -> id ( ARGUMENTS )
ARGUMENTS -> ARGUMENT-LIST | empty
ARGUMENT-LIST -> ARGUMENT-LIST , EXPRESSION | EXPRESSION
```

错误报告

是否为SLR(1): ☐ 否

移进-规约冲突: DFA的状态q8中移进项的first集和归约项的follow集交集不为空

编码分析输入

导入编码 保存编码 文法分析

First Follow LR(0) DFA SLR(1) 分析表 语动分析表 语法树

非终结符first集

1	2
ADDITIVE-EXPRESSION	(id num
ADOP	+ -
ARGUMENT-LIST	(id num
ARGUMENTS	(empty id num
CALL	id
COMPOUND-STM	{
CONDITION-STM	if
DEFINITION	double float int void
DEFINITION-LIST	double float int void
DOWHILE-STM	do

非终结符follow集

1	2
ADDITIVE-EXPRESSION	[=) +, -, ; < == > >=]
ADOP	(id num
ARGUMENT-LIST),
ARGUMENTS	}
CALL	[= %) * +, -, / ; < == > >=]
COMPOUND-STM	{ (; do double else float id if int num return void while { }
CONDITION-STM	{ (; do else id if num return while { }
DEFINITION	\$ double float int void
DEFINITION-LIST	\$ double float int void
DOWHILE-STM	{ (; do else id if num return while { }

SLR Analyzer

文法规则输入

导入文件 保存文法 SLR(1) 分析

```
CONTEXTS: SINT -> { LOCAL-DEFINITIONS STATEMENTS } LIST ;
LOCAL-DEFINITIONS -> LOCAL-DEFINITIONS VARIABLE-DEFINITION | empty
STATEMENT-LIST -> STATEMENT-LIST STATEMENT | empty
STATEMENT -> EXPRESSION-STM | COMPOUND-STM | CONDITION-STM | DOWHILE-STM | RETURN-STM
EXPRESSION-STM -> EXPRESSION ;
CONDITION-STM -> if ( EXPRESSION ) STATEMENT | if ( EXPRESSION ) STATEMENT else STATEMENT
DOWHILE-STM -> do STATEMENT while ( EXPRESSION ) ;
RETURN-STM -> return ; | return EXPRESSION ;
EXPRESSION -> VARIABLE = EXPRESSION | SIMPLE-EXPRESSION
VARIABLE -> id | id [ EXPRESSION ]
SIMPLE-EXPRESSION -> ADDITIVE-EXPRESSION RELOP ADDITIVE-EXPRESSION | ADDITIVE-EXPRESSION
RELOP -> <= | < | > | >= | == | !=
ADDITIVE-EXPRESSION -> ADDITIVE-EXPRESSION ADOP TERM | TERM
ADOP -> + | -
TERM -> TERM MULOP FACTOR | FACTOR
MULOP -> * | / | %
FACTOR -> ( EXPRESSION ) | VARIABLE | CALL | num
CALL -> id ( ARGUMENTS )
ARGUMENTS -> ARGUMENT-LIST | empty
ARGUMENT-LIST -> ARGUMENT-LIST , EXPRESSION | EXPRESSION
```

错误报告

是否为SLR(1): ☐ 否

移进-规约冲突: DFA的状态q8中移进项的first集和归约项的follow集交集不为空

编码分析输入

导入编码 保存编码 文法分析

First Follow LR(0) DFA SLR(1) 分析表 语动分析表 语法树

非终结符first集

1	2
PROGRAM	double float int void
RELOP	[= < <== > >=]
RETURN-STM	return
SIMPLE-EXPRESSION	(id num
STATEMENT	{ (; do id if num return { }
STATEMENT-LIST	{ (; do empty id if num return { }
TERM	(id num
TYPE-INDICATOR	double float int void
VARIABLE	id
VARIABLE-DEFINITION	double float int void

非终结符follow集

1	2
PROGRAM	\$
RELOP	(id num
RETURN-STM	{ (; do else id if num return while { }
SIMPLE-EXPRESSION), ;]
STATEMENT	{ (; do else id if num return while { }
STATEMENT-LIST	{ (; do id if num return { }
TERM	[= %) * +, -, / ; < == > >=]
TYPE-INDICATOR	id
VARIABLE	[= %) * +, -, / ; < == > >=]
VARIABLE-DEFINITION	{ (; do double float id if int num return void { }

TINY

SLR Analyzer
⌵ ⌶

文法规则输入

```

PROGRAM -> STMT-SEQUENCE
STMT-SEQUENCE -> STMT-SEQUENCE ; STATEMENT | STATEMENT
STATEMENT -> IF-STMT | REPEAT-STMT | ASSIGN-STMT | READ-STMT | WRITE-STMT
IF-STMT -> if EXP then STMT-SEQUENCE end | if EXP then STMT-SEQUENCE elso STMT-SEQUENCE end
REPEAT-STMT -> repeat STMT-SEQUENCE until EXP
ASSIGN-STMT -> identifier = EXP
READ-STMT -> read identifier
WRITE-STMT -> write EXP
EXP -> SIMPLE-EXP COMPARISON-OP SIMPLE-EXP | SIMPLE-EXP
COMPARISON-OP -> { = | < | <= | > | >= | }
SIMPLE-EXP -> SIMPLE-EXP ADOPF TERM | TERM
ADOPF -> + | -
TERM -> TERM MULOP FACTOR | FACTOR
MULOP -> * | / | %
FACTOR -> { EXP } | number | identifier
        
```

错误报告

是否为SLR(1): ☒ 是 ☐ 否

First Follow
LR(0) DFA
SLR(1) 分析表
项目分析表
语法制

DFA状态及各项目录

状态	项目集
0	PROGRAM->STMT-SEQUENCE STMT-SEQUENCE->STMT-SEQUENCE; STATEMENT STMT-SEQUENCE->STATEMENT STATEMENT->IF-STMT STATEMENT->REPEAT-STMT STATEMENT->ASSIGN-STMT STATEMENT->READ-STMT STATEMENT->WRITE-STMT
1	PROGRAM->STMT-SEQUENCE . STMT-SEQUENCE->STMT-SEQUENCE; STATEMENT

DFA状态转换表

	ADOPF	ASSIGN-STMT	COMPARISON-OP	EXP	FACTOR	IF-STMT	MULOP	PROGRAM	READ-STMT	REPEAT-STMT	SIMPLE-ID
0	5					3		6		4	
1											
2											
3											
4											
5											
6											
7											
8											
9	14	17									15

编码分析输入

[illegible]

mini-c

[illegible]

SLR Analyzer

文法规则输入

```
<E>-><S> | <A-B*>|<B/A/B+C/D+E*>|<D> ;  
LOCAL-DEFINITIONS-> LOCAL-DEFINITIONS VARIABLE-DEFINITION | empty  
STATEMENT-LIST-> STATEMENT-LIST STATEMENT | empty  
STATEMENT-> EXPRESSION-STMT | COMPOUND-STMT | CONDITION-STMT | DOWHILE-STMT | RETURN-STMT  
EXPRESSION-STMT-> EXPRESSION - |  
CONDITION-STMT-> if { EXPRESSION } STATEMENT | if { EXPRESSION } STATEMENT else STATEMENT  
DOWHILE-STMT-> do STATEMENT while { EXPRESSION } :  
RETURN-STMT-> return ; | return EXPRESSION  
EXPRESSON-> VARIABLE = EXPRESSION | SIMPLE-EXPRESSION  
VARIABLE-> id | id [ EXPRESSION ]  
SIMPLE-EXPRESSION-> ADDITIVE-EXPRESSION RELOP ADDITIVE-EXPRESSION | ADDITIVE-EXPRESSION  
RELOP-> <= | < | > | >= | == | !=  
ADDITIVE-EXPRESSION-> ADDITIVE-EXPRESSION ADOPOTERM | TERMM  
ADOPOTERM-> + | - | * | /  
TERM-> TERM MULOP FACTOR | FACTOR  
MULOP-> * | / | %  
FACTOR-> ( EXPRESSION ) | VARIABLE | CALL | num  
CALL-> id ( ARGUMENTS )  
ARGUMENTS-> ARGUMENT-LIST | empty  
ARGUMENT-LIST-> ARGUMENT-LIST , EXPRESSION | EXPRESSION
```

☐ 是否为SLR(1)： ☒ 否

错误报告

移进—规约冲突：I_kA的状态q₈中移进项的first集和归约项的follow集的交集不为空。

First Follow LR(0) DFA SLR(1) 分析表 语句分析表 语法树

I₀A状态及各项归属

状态	项目集
99	SIMPLE-EXPRESSION->+ADDITIVE-EXPRESSION RELOP ADDITIVE-EXPRESSION SIMPLE-EXPRESSION->+ADDITIVE-EXPRESSION ADDITIVE-EXPRESSION->+ADDITIVE-EXPRESSION ADOPOTERM ADDITIVE-EXPRESSION->TERM TERM->TERM MULOP FACTOR TERM->FACTOR FACTOR->(EXPRESSION) FACTOR->VARIABLE FACTOR->CALL FACTOR->num CALL->id (ARGUMENTS)
99	DOWHILE-STMT->do STATEMENT while { EXPRESSION } :
100	CONDITION-STMT->if { EXPRESSION } STATEMENT else STATEMENT .
101	DOWHILE-STMT->do STATEMENT while { EXPRESSION } .

I₀A状态转换表

%c	void id (; [num]) , { do if return } = + - < <= == > >= % * / while else
92	
93	
94	48 52 54
95	
96	
97	99
98	48 52 42 54 27 44 43 45
99	101
100	
101	

编码分析输入

TINY

SLR Analyzer

✖
🔍
☰

文法规则输入

导入文件
保存文法
SLR(1)分析

```

PROGRAM -> STMT-SEQUENCE
STMT-SEQUENCE -> STMT-SEQUENCE STATEMENT | STATEMENT
STATEMENT -> IF-STMT | REPEAT-STMT | ASSIGN-STMT | READ-STMT | WRITE-STMT
IF-STMT -> if EXP then STMT-SEQUENCE end | if EXP then STMT-SEQUENCE else STMT-SEQUENCE end
REPEAT-STMT -> repeat STMT-SEQUENCE until EXP
ASSIGN-STMT -> Identifier := EXP
READ-STMT -> read Identifier
WRITE-STMT -> write EXP
EXP -> SIMPLE-EXP COMPARISON-OP SIMPLE-EXP | SIMPLE-EXP
COMPARISON-OP -> < | = | <= | < > | >= | >
SIMPLE-EXP -> SIMPLE-EXP ADOOP TERM | TERM
ADOOP -> + | -
TERM -> TERM NULOP FACTOR | FACTOR
NULOP -> * | / | %
FACTOR -> ( EXP ) | number | Identifier

```

错误报告

是否为SLR(1):
☒ 是

First Follow
LR(0) DFA
SLR(1) 分析表
选句分析表
语法制

	\$	%	(
0			
1	接受		
2	r(STMT-SEQUENCE -> STATEMENT)		
3	r(STATEMENT -> IF-STMT)		
4	r(STATEMENT -> REPEAT-STMT)		
5	r(STATEMENT -> ASSIGN-STMT)		
6	r(STATEMENT -> READ-STMT)		
7	r(STATEMENT -> WRITE-STMT)		
8			s18
9			
10			
11			
12			s18
13			
14			
15	r(EXP -> SIMPLE-EXP)		
16	r(SIMPLE-EXP -> TERM)	s40	
17	r(TERM -> FACTOR)	r(TERM -> FACTOR)	
18			s18
19	r(FACTOR -> number)	r(FACTOR -> number)	
20	r(FACTOR -> identifier)	r(FACTOR -> identifier)	
21			
22			s18

[illegible]

mini-c

SLR Analyzer

文法规则输入

导入文件 保存文法 SLR(1) 分析

```
COMPOUND-STMT -> ( LOCAL-DEFINITIONS STATEMENT-LIST )
LOCAL-DEFINITIONS -> LOCAL-DEFINITIONS VARIABLE-DEFINITION | empty
STATEMENT-LIST -> STATEMENT-LIST STATEMENT | empty
STATEMENT -> EXPRESSION-STATEMENT | COMPOUND-STMT | CONDITION-STMT | DOWHILE-STMT | RETURN-STMT
EXPRESSION-STATEMENT -> EXPRESSION ;
CONDITION-STMT -> if ( EXPRESSION ) STATEMENT | if ( EXPRESSION ) STATEMENT else STATEMENT
DOWHILE-STMT -> do STATEMENT while ( EXPRESSION ) ;
RETURN-STMT -> return ; | return EXPRESSION ;
EXPRESSION -> VARIABLE = EXPRESSION | SIMPLE-EXPRESSION
VARIABLE -> id | if [ EXPRESSION ]
SIMPLE-EXPRESSION -> ADDITIVE-EXPRESSION RELOP ADDITIVE-EXPRESSION | ADDITIVE-EXPRESSION
RELOP -> <= | < | > | >= | == | !=
ADDITIVE-EXPRESSION -> ADDITIVE-EXPRESSION ADDOP TERM | TERM
ADDOP -> + | -
TERM -> TERM MULOOP FACTOR | FACTOR
MULOOP -> * | / | %
FACTOR -> ( EXPRESSION ) | VARIABLE | CALL | num
CALL -> id ( ARGUMENTS )
ARGUMENTS -> ARGUMENT-LIST | empty
ARGUMENT-LIST -> ARGUMENT-LIST , EXPRESSION | EXPRESSION
```

错误报告

是否为SLR(1): ☐ 否

移进-规约冲突: DFA的状态96中移进项的first集和归约项的follow集交集不为空

编码分析输入

导入编码 保存编码 文法分析

First Follow LR(0) DFA SLR(1) 分析表 语初分析表 语法树

0		!=		\$
1			接受	
2			r(DEFINITION-LIST -> DEFINITION)	
3			r(DEFINITION -> VARIABLE-DEFINITION)	
4			r(DEFINITION -> FUNCTION-DEFINITION)	
5				
6				
7				
8				
9				
10			r(DEFINITION-LIST -> DEFINITION-LIST DEFINITION)	
11				
12			r(VARIABLE-DEFINITION -> TYPE-INDICATOR id ;)	
13				
14				
15				
16				
17				
18				
19				
20				
21				
22				

SLR Analyzer

文法规则输入

导入文件 保存文法 SLR(1) 分析

```
COMPOUND-STMT -> ( LOCAL-DEFINITIONS STATEMENT-LIST )
LOCAL-DEFINITIONS -> LOCAL-DEFINITIONS VARIABLE-DEFINITION | empty
STATEMENT-LIST -> STATEMENT-LIST STATEMENT | empty
STATEMENT -> EXPRESSION-STATEMENT | COMPOUND-STMT | CONDITION-STMT | DOWHILE-STMT | RETURN-STMT
EXPRESSION-STATEMENT -> EXPRESSION ;
CONDITION-STMT -> if ( EXPRESSION ) STATEMENT | if ( EXPRESSION ) STATEMENT else STATEMENT
DOWHILE-STMT -> do STATEMENT while ( EXPRESSION ) ;
RETURN-STMT -> return ; | return EXPRESSION ;
EXPRESSION -> VARIABLE = EXPRESSION | SIMPLE-EXPRESSION
VARIABLE -> id | if [ EXPRESSION ]
SIMPLE-EXPRESSION -> ADDITIVE-EXPRESSION RELOP ADDITIVE-EXPRESSION | ADDITIVE-EXPRESSION
RELOP -> <= | < | > | >= | == | !=
ADDITIVE-EXPRESSION -> ADDITIVE-EXPRESSION ADDOP TERM | TERM
ADDOP -> + | -
TERM -> TERM MULOOP FACTOR | FACTOR
MULOOP -> * | / | %
FACTOR -> ( EXPRESSION ) | VARIABLE | CALL | num
CALL -> id ( ARGUMENTS )
ARGUMENTS -> ARGUMENT-LIST | empty
ARGUMENT-LIST -> ARGUMENT-LIST , EXPRESSION | EXPRESSION
```

错误报告

是否为SLR(1): ☐ 否

移进-规约冲突: DFA的状态96中移进项的first集和归约项的follow集交集不为空

编码分析输入

导入编码 保存编码 文法分析

First Follow LR(0) DFA SLR(1) 分析表 语初分析表 语法树

	{		}
IN-STMT -> return EXPRESSION ;			r(RETURN-STMT -> return EXPRESSION ;)
ITION-STMT -> if (EXPRESSION) STATEMENT			r(CONDITION-STMT -> if (EXPRESSION) STATEMENT)
ITION-STMT -> if (EXPRESSION) STATEMENT else STATEMENT			r(CONDITION-STMT -> if (EXPRESSION) STATEMENT else STATEMENT)
HILE-STMT -> do STATEMENT while (EXPRESSION) ;			r(DOWHILE-STMT -> do STATEMENT while (EXPRESSION) ;)

4.4 语句分析表

TINY

SLR Analyzer

文法规则输入

导入文件 保存文法 SLR(1)分析

```
PROGRAM -> STMT-SEQUENCE
STMT-SEQUENCE -> STMT-SEQUENCE ; STATEMENT | STATEMENT
STATEMENT -> IF-STM | REPEAT-STM | ASSIGN-STM | READ-STM | WRITE-STM
IF-STM -> if EXP then STMT-SEQUENCE end | if EXP then STMT-SEQUENCE else STMT-SEQUENCE end
REPEAT-STM -> repeat STMT-SEQUENCE until EXP
ASSIGN-STM -> identifier := EXP
READ-STM -> read identifier
WRITE-STM -> write EXP
EXP -> SIMPLE-EXP COMPARISON-OP SIMPLE-EXP | SIMPLE-EXP
COMPARISON-OP -> < | = | <= | <= | > | >= | >= | >=
SIMPLE-EXP -> SIMPLE-EXP ADOP TERM | TERM
ADOP -> + | -
TERM -> TERM MULOP FACTOR | FACTOR
MULOP -> * | / | %
FACTOR -> ( EXP ) | number | identifier
```

错误报告

是否为SLR(1): 是

编码分析输入

导入编码 保存编码 文法分析

```
LETTER 110*
DIGIT 111*
= 112
< 113
<= 114
> 115
>= 116
> 117
+ 118
- 119
* 120
/ 121
% 122
:= 123
:= 124
1
100 108 x 124 100 109 0 113 108 x 101 108 fact 123 109 1 124 104 108 fact 123 108 fact 120 108 x 124
108 x 123 108 x 119 109 1 105 108 x 112 109 0 124 107 108 fact 103 124 108 abc 124 100 108 abc
115 108 x 101 108 y 123 109 123 124 108 abc 123 108 x 122 108 y 102 108 abc 123 108 y 121 108 x 103
124 100 108 abc 113 108 y 101 107 108 abc 103 124 100 108 abc 117 108 y 101 107 108 y 103
```

First Follow LR(0) DFA SLR(1) 分析表 语句分析表 语法规则

步骤

状态栈

1	\$ 0
2	\$ 0 read 11
3	\$ 0 read 11 identifier 23
4	\$ 0 READ-STM 6
5	\$ 0 STATEMENT 2
6	\$ 0 STMT-SEQUENCE 1
7	\$ 0 STMT-SEQUENCE 1; 13
8	\$ 0 STMT-SEQUENCE 1; 13 if 8
9	\$ 0 STMT-SEQUENCE 1; 13 if 8 number 19
10	\$ 0 STMT-SEQUENCE 1; 13 if 8 FACTOR 17
11	\$ 0 STMT-SEQUENCE 1; 13 if 8 TERM 16
12	\$ 0 STMT-SEQUENCE 1; 13 if 8 SIMPLE-EXP 15
13	\$ 0 STMT-SEQUENCE 1; 13 if 8 SIMPLE-EXP 15 < 29
14	\$ 0 STMT-SEQUENCE 1; 13 if 8 SIMPLE-EXP 15 COMPARISON-OP 27
15	\$ 0 STMT-SEQUENCE 1; 13 if 8 SIMPLE-EXP 15 COMPARISON-OP 27 identifier 20
16	\$ 0 STMT-SEQUENCE 1; 13 if 8 SIMPLE-EXP 15 COMPARISON-OP 27 FACTOR 17
17	\$ 0 STMT-SEQUENCE 1; 13 if 8 SIMPLE-EXP 15 COMPARISON-OP 27 TERM 16
18	\$ 0 STMT-SEQUENCE 1; 13 if 8 SIMPLE-EXP 15 COMPARISON-OP 27 SIMPLE-EXP 45
19	\$ 0 STMT-SEQUENCE 1; 13 if 8 EXP 14
20	\$ 0 STMT-SEQUENCE 1; 13 if 8 EXP 14 then 26
21	\$ 0 STMT-SEQUENCE 1; 13 if 8 EXP 14 then 26 identifier 10
22	\$ 0 STMT-SEQUENCE 1; 13 if 8 EXP 14 then 26 identifier 10 := 22
23	\$ 0 STMT-SEQUENCE 1; 13 if 8 EXP 14 then 26 identifier 10 := 22 number 19

SLR Analyzer

文法规则输入

导入文件 保存文法 SLR(1)分析

```
PROGRAM -> STMT-SEQUENCE
STMT-SEQUENCE -> STMT-SEQUENCE ; STATEMENT | STATEMENT
STATEMENT -> IF-STM | REPEAT-STM | ASSIGN-STM | READ-STM | WRITE-STM
IF-STM -> if EXP then STMT-SEQUENCE end | if EXP then STMT-SEQUENCE else STMT-SEQUENCE end
REPEAT-STM -> repeat STMT-SEQUENCE until EXP
ASSIGN-STM -> identifier := EXP
READ-STM -> read identifier
WRITE-STM -> write EXP
EXP -> SIMPLE-EXP COMPARISON-OP SIMPLE-EXP | SIMPLE-EXP
COMPARISON-OP -> < | = | <= | <= | > | >= | >= | >=
SIMPLE-EXP -> SIMPLE-EXP ADOP TERM | TERM
ADOP -> + | -
TERM -> TERM MULOP FACTOR | FACTOR
MULOP -> * | / | %
FACTOR -> ( EXP ) | number | identifier
```

错误报告

是否为SLR(1): 是

编码分析输入

导入编码 保存编码 文法分析

```
LETTER 110*
DIGIT 111*
= 112
< 113
<= 114
> 115
>= 116
> 117
+ 118
- 119
* 120
/ 121
% 122
:= 123
:= 124
1
100 108 x 124 100 109 0 113 108 x 101 108 fact 123 109 1 124 104 108 fact 123 108 fact 120 108 x 124
108 x 123 108 x 119 109 1 105 108 x 112 109 0 124 107 108 fact 103 124 108 abc 124 100 108 abc
115 108 x 101 108 y 123 109 123 124 108 abc 123 108 x 122 108 y 102 108 abc 123 108 y 121 108 x 103
124 100 108 abc 113 108 y 101 107 108 abc 103 124 100 108 abc 117 108 y 101 107 108 y 103
```

First Follow LR(0) DFA SLR(1) 分析表 语句分析表 语法规则

行为

用SIMPLE-EXP -> TERM 归约
移进34
用COMPARISON-OP -> > 归约
移进20
用FACTOR -> identifier 归约
用TERM -> FACTOR 归约
用SIMPLE-EXP -> TERM 归约
用EXP -> SIMPLE-EXP COMPARISON-OP SIMPLE-EXP 归约
移进26
移进12
移进20
用FACTOR -> identifier 归约
用TERM -> FACTOR 归约
用SIMPLE-EXP -> TERM 归约
用EXP -> SIMPLE-EXP 归约
用WRITE-STM -> write EXP 归约
用STATEMENT -> WRITE-STM 归约
用STMT-SEQUENCE -> STATEMENT 归约
移进50
用IF-STM -> if EXP then STMT-SEQUENCE end 归约
用STATEMENT -> IF-STM 归约
用STMT-SEQUENCE -> STMT-SEQUENCE; STATEMENT 归约
接受0

mini-c

SLR Analyzer

文法规则输入

导入文件 保存文法 SLR(1)分析

```
COMPOUND-STMT -> { LOCAL-DEFINITIONS STATEMENT* LIST }
LOCAL-DEFINITIONS -> LOCAL-DEFINITIONS VARIABLE-DEFINITION | empty
STATEMENT-LIST -> STATEMENT-LIST STATEMENT | empty
STATEMENT -> EXPRESSION-STM | COMPOUND-STMT | CONDITION-STMT | DOWHILE-STMT | RETURN-STMT
EXPRESSION-STM -> EXPRESSION | {
CONDITION-STMT -> if ( EXPRESSION ) STATEMENT | if ( EXPRESSION ) STATEMENT else STATEMENT
DOWHILE-STMT -> do STATEMENT while ( EXPRESSION ) ;
RETURN-STMT -> return ; | return EXPRESSION ;
EXPRESSION -> VARIABLE = EXPRESSION | SIMPLE-EXPRESSION
VARIABLE -> id | if { EXPRESSION }
SIMPLE-EXPRESSION -> ADDITIVE-EXPRESSION RELOP ADDITIVE-EXPRESSION | ADDITIVE-EXPRESSION
RELOP -> <= | < | > | >= | =
ADDITIVE-EXPRESSION -> ADDITIVE-EXPRESSION ADDOP TERM | TERM
ADDOP -> + | -
TERM -> TERM NILOP FACTOR | FACTOR
NILOP -> * | / | %
FACTOR -> ( EXPRESSION ) | VARIABLE | CALL | num
CALL -> id ( ARGUMENTS )
ARGUMENTS -> ARGUMENT-LIST | empty
ARGUMENT-LIST -> ARGUMENT-LIST , EXPRESSION | EXPRESSION
```

错误报告

是否为SLR(1): ☐ 否

移进-规约冲突: DFA的状态95中移进项的first集和归约项的follow集交集不为空

编码分析输入

导入编码 保存编码 文法分析

```
+ 122
+ 123
+ 124
+ 125
+ 126
+ 127
+ 128
+ 129
+ 130
+ 131
+ 132
}
102 109 number 125 104 109 result 125 103 109 arr 129 110 10 130 125 102 109 factorial 127 102 109 n
126 102 109 a 129 130 128 131 102 109 *tola 125 109 *tola 124 110 3.14 125 109 a 124 109 arr 125
101 127 109 a 129 110 0 120 116 110 3.2 125 131 109 a 129 110 0 130 124 110 5 121 110 1 125 132 107
131 109 *tola 124 109 result 110 109 n 125 109 n 124 109 n 123 110 1 125 132 108 127 109 n 118 110
1 125 125 109 109 *tola 125 132 102 109 main 127 106 125 131 109 number 124 110 5 125 109 result
124 109 factorial 127 109 number 128 125 109 arr 129 110 1 130 124 109 arr 129 110 3 120 127 110 1
122 110 2 128 130 125 101 127 109 result 113 110 100 128 131 105 110 1 125 132 100 131 105 110 1.234
125 132 132
```

First Follow LR(0) DFA SLR(1) 分析表 语词分析表 语法树

步骤	状态
1	\$ 0
2	\$ 0 int 6
3	\$ 0 TYPE-INDICATOR 5
4	\$ 0 TYPE-INDICATOR 5 id 11
5	\$ 0 TYPE-INDICATOR 5 id 11; 12
6	\$ 0 VARIABLE-DEFINITION 3
7	\$ 0 DEFINITION 2
8	\$ 0 DEFINITION-LIST 1
9	\$ 0 DEFINITION-LIST 1 double 8
10	\$ 0 DEFINITION-LIST 1 TYPE-INDICATOR 5
11	\$ 0 DEFINITION-LIST 1 TYPE-INDICATOR 5 id 11
12	\$ 0 DEFINITION-LIST 1 TYPE-INDICATOR 5 id 11; 12
13	\$ 0 DEFINITION-LIST 1 VARIABLE-DEFINITION 3
14	\$ 0 DEFINITION-LIST 1 DEFINITION 10
15	\$ 0 DEFINITION-LIST 1
16	\$ 0 DEFINITION-LIST 1 float 7
17	\$ 0 DEFINITION-LIST 1 TYPE-INDICATOR 5
18	\$ 0 DEFINITION-LIST 1 TYPE-INDICATOR 5 id 11
19	\$ 0 DEFINITION-LIST 1 TYPE-INDICATOR 5 id 11 [13
20	\$ 0 DEFINITION-LIST 1 TYPE-INDICATOR 5 id 11 [13 num 15
21	\$ 0 DEFINITION-LIST 1 TYPE-INDICATOR 5 id 11 [13 num 15] 21
22	\$ 0 DEFINITION-LIST 1 TYPE-INDICATOR 5 id 11 [13 num 15] 21; 25
23	\$ 0 DEFINITION-LIST 1 VARIABLE-DEFINITION 3

SLR Analyzer

文法规则输入

导入文件 保存文法 SLR(1)分析

```
COMPOUND-STMT -> { LOCAL-DEFINITIONS STATEMENT* LIST }
LOCAL-DEFINITIONS -> LOCAL-DEFINITIONS VARIABLE-DEFINITION | empty
STATEMENT-LIST -> STATEMENT-LIST STATEMENT | empty
STATEMENT -> EXPRESSION-STM | COMPOUND-STMT | CONDITION-STMT | DOWHILE-STMT | RETURN-STMT
EXPRESSION-STM -> EXPRESSION | {
CONDITION-STMT -> if ( EXPRESSION ) STATEMENT | if ( EXPRESSION ) STATEMENT else STATEMENT
DOWHILE-STMT -> do STATEMENT while ( EXPRESSION ) ;
RETURN-STMT -> return ; | return EXPRESSION ;
EXPRESSION -> VARIABLE = EXPRESSION | SIMPLE-EXPRESSION
VARIABLE -> id | if { EXPRESSION }
SIMPLE-EXPRESSION -> ADDITIVE-EXPRESSION RELOP ADDITIVE-EXPRESSION | ADDITIVE-EXPRESSION
RELOP -> <= | < | > | >= | =
ADDITIVE-EXPRESSION -> ADDITIVE-EXPRESSION ADDOP TERM | TERM
ADDOP -> + | -
TERM -> TERM NILOP FACTOR | FACTOR
NILOP -> * | / | %
FACTOR -> ( EXPRESSION ) | VARIABLE | CALL | num
CALL -> id ( ARGUMENTS )
ARGUMENTS -> ARGUMENT-LIST | empty
ARGUMENT-LIST -> ARGUMENT-LIST , EXPRESSION | EXPRESSION
```

错误报告

是否为SLR(1): ☐ 否

移进-规约冲突: DFA的状态95中移进项的first集和归约项的follow集交集不为空

编码分析输入

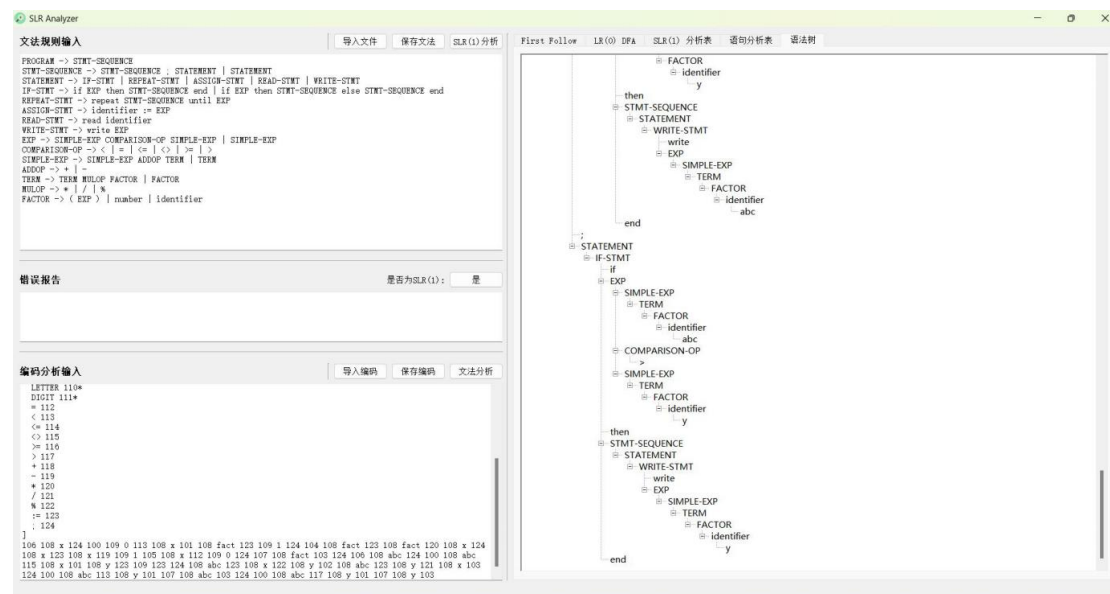
导入编码 保存编码 文法分析

```
+ 122
+ 123
+ 124
+ 125
+ 126
+ 127
+ 128
+ 129
+ 130
+ 131
+ 132
}
102 109 number 125 104 109 result 125 103 109 arr 129 110 10 130 125 102 109 factorial 127 102 109 n
126 102 109 a 129 130 128 131 102 109 *tola 125 109 *tola 124 110 3.14 125 109 a 124 109 arr 125
101 127 109 a 129 110 0 120 116 110 3.2 125 131 109 a 129 110 0 130 124 110 5 121 110 1 125 132 107
131 109 *tola 124 109 result 110 109 n 125 109 n 124 109 n 123 110 1 125 132 108 127 109 n 118 110
1 125 125 109 109 *tola 125 132 102 109 main 127 106 125 131 109 number 124 110 5 125 109 result
124 109 factorial 127 109 number 128 125 109 arr 129 110 1 130 124 109 arr 129 110 3 120 127 110 1
122 110 2 128 130 125 101 127 109 result 113 110 100 128 131 105 110 1 125 132 100 131 105 110 1.234
125 132 132
```

First Follow LR(0) DFA SLR(1) 分析表 语词分析表 语法树

行为
移进45
移进54
用FACTOR -> num 归约
用TERM -> FACTOR 归约
用ADDITIVE-EXPRESSION -> TERM 归约
用SIMPLE-EXPRESSION -> ADDITIVE-EXPRESSION 归约
用EXPRESSION -> SIMPLE-EXPRESSION 归约
移进80
用RETURN-STM -> return EXPRESSION; 归约
用STATEMENT -> RETURN-STM 归约
用STATEMENT-LIST -> STATEMENT-LIST STATEMENT 归约
移进34
用COMPOUND-STMT -> { LOCAL-DEFINITIONS STATEMENT-LIST } 归约
用STATEMENT -> COMPOUND-STMT 归约
用CONDITION-STMT -> if (EXPRESSION) STATEMENT else STATEMENT 归约
用STATEMENT -> CONDITION-STMT 归约
用STATEMENT-LIST -> STATEMENT-LIST STATEMENT 归约
移进34
用COMPOUND-STMT -> { LOCAL-DEFINITIONS STATEMENT-LIST } 归约
用FUNCTION-DEFINITION -> TYPE-INDICATOR id (PARAMETERS) COMPOUND-STMT 归约
用DEFINITION -> FUNCTION-DEFINITION 归约
用DEFINITION-LIST -> DEFINITION-LIST DEFINITION 归约
接受0

TINY



mini-c

The screenshot shows the SLR Analyzer interface with the following components:

- 文法规则输入 (Grammar Rule Input):** A text area containing the mini-c grammar rules, such as `LOCAL-DEFINITIONS -> (LOCAL-DEFINITIONS STATEMENT*) LOCAL-DEFINITIONS`.
- 错误报告 (Error Report):** A section indicating a shift-reduce conflict at state 95, where the `first` and `follow` sets of the `if` non-terminal are not empty.
- 编码分析输入 (Encoding Analysis Input):** A text area containing the tokenized input code for a factorial function.
- 语法树 (Syntax Tree):** A hierarchical tree structure representing the parsed code, showing the derivation from the root `PROGRAM` down to the terminal tokens.

This screenshot shows the same SLR Analyzer interface, but with a different view of the syntax tree and the error report. The error report still indicates a shift-reduce conflict at state 95. The syntax tree shows the derivation of the factorial function code, with the `if` statement being the root of the tree.

五、评价

5.1 软件能力

该软件能够对多种编程语言的文法进行语法分析，并生成相应的分析信息；能够根据单词编码，生成分析表和相应的语法树。

5.2 缺陷与限制

该软件的缺陷之一在于对输入的规范与否的检查并不严格，若输入的文法和编码不符合规范，软件可能不会提示错误信息，而是继续执行分析过程，这可能导致软件中断运行。

5.3 建议

加强在输入时对输入内容格式的检查，并在输入有误时产生提示并中断分析。

5.4 测试结论

软件能够正确分析出 First、Follow 集，生成正确的 LR(0) DFA 状态、转换和 SLR(1)分析表；将项目一生成的单词编码输入后，能够执行正确的文法分析并生成语法树。