The Parser of Pascal

C++ Implementation

Outline

- 1.Introduction
- 2.Journal of testing
- 3.Conclusion

Outline

- 1.Introduction
- 2.Journal of testing
- 3.Conclusion

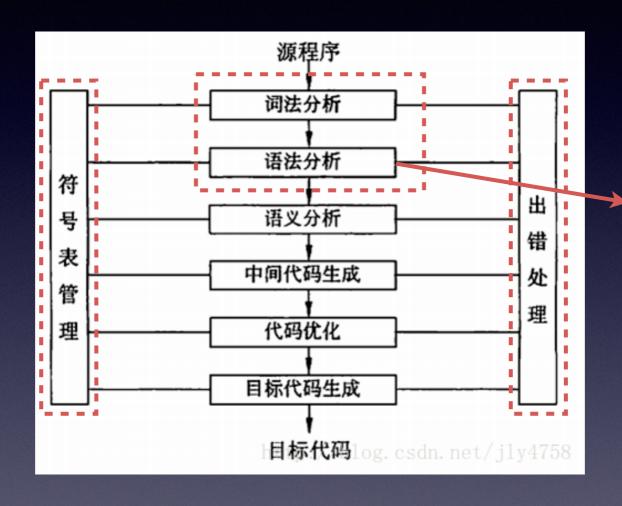


Overview



Figure: Processing Pipeline

Overview



We have reached here!

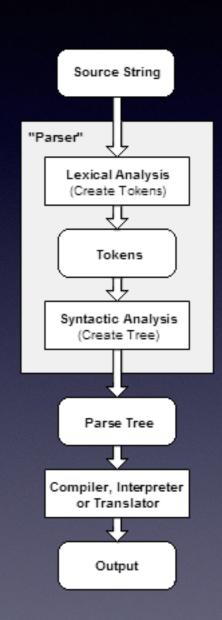
Figure: Processing Pipeline

Introduction

- Writing a C++ parser with respect to SLR(1) mechanism.
- Two approaches to construct SLR(1) parser:
 - 1.Flex+Bison: Mature and Simple
 - 2.Hand-Writing: Learning SLR(1) the "hard" way

Parsing

- · Given a grammar and a statement;
- Judging if the statement matches the grammar.



Bottom-Up LR Parsing

- Scanning and parsing the input text.
- Building up the parse tree bottom up, and Left to right.
- Producing a Rightmost derivation in reverse

Bottom-Up LR Parsing

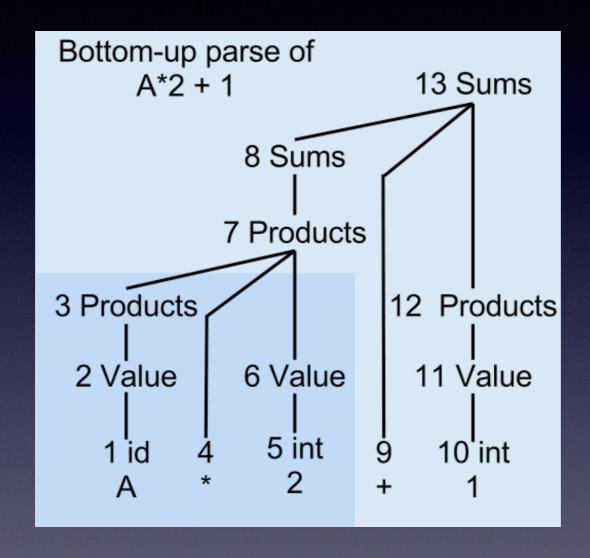


Figure: An Example of Bottom-Up Parsing

Shift and Reduce Actions

- Shift: advances in the input stream by one symbol
- Reduce: applies a completed grammar rule

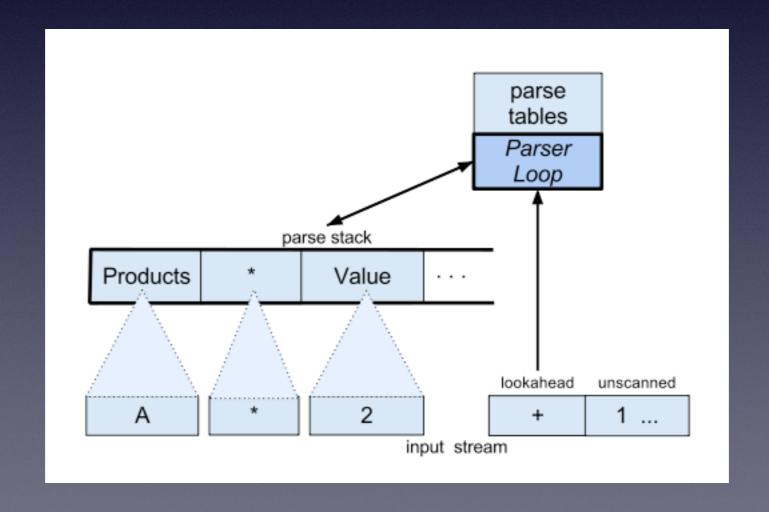
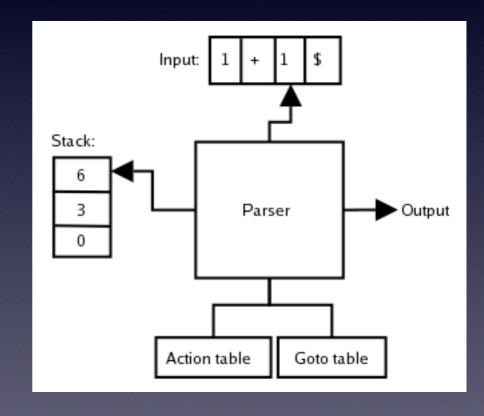


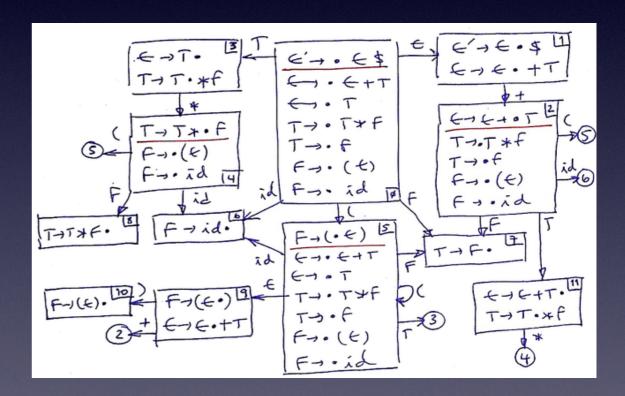
Table-based LR

- Stack: Runtime State, Symbol
- Goto Table: State Transition
- Action Table: Applied Action



Parsing Table Construction

- Items, eg. $E \rightarrow \bullet E + B$
- Extended closure of item sets
- Building the canonical LR(1) collection
- DFA: Guide of Parsing



Conflict Resolution

- Shift-Reduce Conflicts
- Reduce-Reduce Conflicts
- Operator priority: Self-defined
- SLR(1):
 - Lookahead
 - Follow set



Problems with Grammars

- Grammars can cause problems when constructing a LR parser
 - Shift-reduce conflicts
 - Reduce-reduce conflicts

1/12/2010

@ 2002-10 Hall Perkins & UW CSE

D-32

SLR(1)

$$A \rightarrow a \cdot Xb$$

- X is both a terminal and the next token in the input string
- Action: Shift;
- New State: the state containing the item $A \rightarrow aX \cdot b$

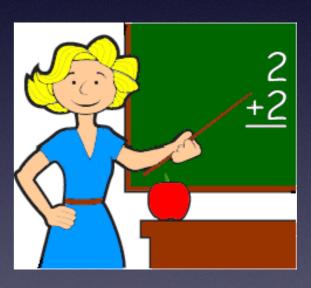
SLR(1)

$$A \rightarrow y \bullet \uparrow$$

- Check the next token "X" in the input string(Lookahead)
- If "X" is in Follow(A):
- Action: **Reduce** by the rule $A \rightarrow y$;
- Else: Raise an error!

Outline

- 1.Introduction
- · 2.Journal of testing
- 3.Conclusion



Input Grammar

- S program id; | compound_stmt.
- compound_stmt begin stmts end
- stmts stmt | stmts; stmt
- stmt id := expr | compound_stmt | if_stmt | for_stmt | while bool do stmt | ε
- if_stmt if bool then stmt | if bool then stmt else stmt
- for_stmt for id := expr to expr do stmt | for id := expr downto expr do stmt
- bool expr > expr | expr < expr
- expr expr + expr | expr expr | expr * expr | expr / expr | expr ^ factor | factor
- factor id | num | (expr)

Test Program

```
program test;
  begin
     x = 19;
     for i:=100 downto 15 do
       if x < y + (15-9) then y := x
             else begin
          while x+y*z>x do begin y:=y+y^5-1 end;
           z := z * 7 + x
  end
end.
```

• Generating Token file by using our lexer program.

```
parser_tests — -bash — 58×24
ChenMac:parser_tests wasdns$ cat given_example_tocken.txt 🗏
Token(PROGRAM, 'PROGRAM')
Token(ID, 'test')
Token(BEGIN, 'BEGIN')
Token(ID, 'x')
Token(ASSIGN, ':=')
Token(INTEGER_CONST, 19)
Token(SEMI, ';')
Token(ID, 'for')
Token(ID, 'i')
Token(ASSIGN, ':=')
Token(INTEGER_CONST, 100)
Token(ID, 'downto')
Token(INTEGER_CONST, 15)
Token(ID, 'do')
Token(ID, 'if')
Token(ID, 'x')
Token(LANGBRA, '<')
Token(ID, 'y')
Token(PLUS, '+')
Token(LPAREN, '(')
Token(INTEGER_CONST, 15)
Token(MINUS, '-')
Token(INTEGER_CONST, 9)
```

Figure 2: Our Lexer Output(Partly)

• Converting the token.txt with the parser-familiar format

```
build — -bash — 58×24
ChenMac:build wasdns$ cat input.txt
program
id
begin
id
:=
num
for
id
:=
num
downto
num
do
if
id
<
id
+
num
```

Figure 3: Converted Lexer Output(Partly)

• Generating output files in build/folder, using Shell scripts.

```
ChenMac:pascal-compiler wasdns$ ls
LICENSE
                                                 run_lexer.py
                        docs
                                                 run_lexer_ply.py
                         lexer_tests
README
build
                        parser_tests
                                                 run_parser_demo.sh
                        ply_frontend
cleanup.sh
                                                 src
ChenMac:pascal-compiler wasdns$ ./run_parser_demo.sh
ChenMac:pascal-compiler wasdns$ cd build/
ChenMac:build wasdns$ ls
action_and_goto.txt
                        grammar.txt
                                                 parser
                                                 slr.txt
                         input.txt
error.txt
first_and_follow.txt
                        output.txt
ChenMac:build wasdns$
```

Generating output files in build/ folder, using Shell scripts.

```
ChenMac:pascal-compiler wasdns$ ls
LICENSE
                                                 run_lexer.py
                        docs
                                                 run_lexer_ply.py
                         lexer_tests
README
                                                run_parser_demo.sh
build
                        parser_tests
                        ply_frontend
cleanup.sh
                                                 src
[ChenMac:pascal-compiler wasdns$ ./run_parser_demo.sh
ChenMac:pascal-compiler wasdns$ cd build/
ChenMac:build wasdns$ ls
action_and_goto.txt
                        grammar.txt
                                                 parser
error.txt
                                                 slr.txt
                        input.txt
first_and_follow.txt
                        output.txt
ChenMac:build wasdns$
```

What's does this script do?

- 1.Creating build/ folder;
- 2. Running Lexer and generating **output files** in build/;
- 3. Compiling parser program;
- 4. Reading and parsing the grammar and the output files;
- 5. Generating results in build/folder.

• Generating output files in build/folder, using Shell scripts.

```
ChenMac:pascal-compiler wasdns$ ls
LICENSE
                                                 run_lexer.py
                        docs
                                                 run_lexer_ply.py
                        lexer_tests
README
build
                        parser_tests
                                                 run_parser_demo.sh
                        ply_frontend
cleanup.sh
                                                 src
[ChenMac:pascal-compiler wasdns$ ./run_parser_demo.sh
ChenMac:pascal-compiler wasdns$ cd build/
ChenMac:build wasdns$ ls
action_and_goto.txt
                        grammar.txt
                                                 parser
                                                 slr.txt
error.txt
                        input.txt
first_and_follow.txt
                        output.txt
ChenMac:build wasdns$
```

Results of running parser

1.first_and_follow.txt: Print the first set and follow set

```
00
```

```
====FIRST====
FIRST(S') = { program }
FIRST(S) = { program }
FIRST(compound_stmt) = { begin }
FIRST(stmts) = \{; , begin , for , id , if , while , \epsilon \}
FIRST(stmt) = \{ begin , for , id , if , while , \epsilon \}
FIRST(if_stmt) = { if }
FIRST(for_stmt) = { for }
FIRST(bool) = \{ ( , id , num \}
FIRST(expr) = { ( , id , num }
FIRST(factor) = { ( , id , num }
FIRST(id) = { id }
FIRST(;) = { ; }
FIRST(.) = {.}
FIRST(begin) = { begin }
FIRST(end) = { end }
FIRST(:=) = { := }
FIRST(while) = { while }
FIRST(do) = { do }
FIRST(if) = { if }
FIRST(then) = { then }
FIRST(else) = { else }
FIRST(to) = { to }
FIRST(downto) = { downto }
FIRST(>) = { > }
FIRST(<) = { < }
FIRST(+) = \{ + \}
FIRST(-) = \{ - \}
FIRST(*) = \{ * \}
FIRST(/) = { / }
FIRST(^) = { ^ }
FIRST(num) = { num }
FIRST(() = { ( } )
FIRST()) = { } 
FIRST(program) = { program }
FIRST(for) = { for }
FIRST(\$) = \{\$\}
_____
====F0LL0W====
FOLLOW(S) = \{ \$ \}
FOLLOW(compound_stmt) = { . , ; , else , end }
FOLLOW(stmts) = { ; , end }
FOLLOW(stmt) = { ; , else , end }
FOLLOW(if_stmt) = { ; , else , end }
FOLLOW(for_stmt) = { ; , else , end }
FOLLOW(bool) = { do , then }
```

Figure: First and Follow

Figure: First and Follow

```
8 0
                                                     first_and_follow.txt
                                                                                                 使用"文本编辑"打开
====FIRST====
FIRST(S') = { program }
FIRST(S) = { program }
FIRST(compound_stmt) = { begin }
FIRST(stmts) = { ; , begin , for , id , if , while , ε }
FIRST(stmt) = { begin , for , id , if , while , ε }
FIRST(if_stmt) = { if }
FIRST(for_stmt) = { for }
FIRST(bool) = { ( , id , num }
FIRST(expr) = { ( , id , num }
FIRST(factor) = { ( , id , num }
FIRST(id) = { id }
FIRST(;) = { ; }
FIRST(.) = { . }
FIRST(begin) = { begin }
FIRST(end) = { end }
FIRST(:=) = { := }
FIRST(while) = { while }
FIRST(do) = { do }
FIRST(if) = { if }
FIRST(then) = { then }
FIRST(else) = { else }
FIRST(to) = { to }
FIRST(downto) = { downto }
FIRST(>) = { > }
FIRST(<) = { < }
FIRST(+) = \{ + \}
FIRST(-) = \{ - \}
FIRST(*) = { * }
FIRST(/) = { / }
FIRST(^) = { ^ }
FIRST(num) = { num }
FIRST(() = { ( } )
FIRST()) = { ) }
FIRST(program) = { program }
FIRST(for) = { for }
FIRST(\$) = \{\$\}
====F0LL0W====
FOLLOW(S) = \{ s \}
FOLLOW(compound_stmt) = { . , ; , else , end }
FOLLOW(stmts) = { ; , end }
FOLLOW(stmt) = { ; , else , end }
FOLLOW(if_stmt) = { ; , else , end }
                                              Follow
FOLLOW(for_stmt) = { ; , else , end }
FOLLOW(bool) = { do , then }
-----
```

Figure: First and Follow

• Generating output files in build/folder, using Shell scripts.

```
ChenMac:pascal-compiler wasdns$ ls
LICENSE
                                                 run_lexer.py
                        docs
                                                 run_lexer_ply.py
                         lexer_tests
README
build
                        parser_tests
                                                 run_parser_demo.sh
                        ply_frontend
cleanup.sh
                                                 src
[ChenMac:pascal-compiler wasdns$ ./run_parser_demo.sh
ChenMac:pascal-compiler wasdns$ cd build/
ChenMac:build wasdns$ ls
action_and_goto.txt
                        grammar.txt
                                                 parser
                                                 slr.txt
                        input.txt
error.txt
first_and_follow.txt
                        output.txt
ChenMac:build wasdns$
```

Results of running parser

```
1.first_and_follow.txt: Print the first set and follow set

2.action_and_goto.txt: Print the action and goto table
```

																			1	actio	n_ar	nd_g	oto.t	xt ~	,												
stat	:e	id	;	١.	beg	in e	nd ::	= whil	.e do	if	then	ı else	to down	<u>to</u> >	<	+	l- 	*	/	<u> </u>		num	()	progra	m fo 	r \$	S	compound_stm	nt stmt	s stm1	t if_stm	t for_stm 	t boo	llexpr	factor	1 1
0	١		l 		<u> </u>		<u> </u>	<u> </u>		1	1	l		<u> </u>	l 	1	<u> </u>						l	l 	S2	<u> </u>	1	1		ı	_I	1		1	1	1	1
1	١		l	1	1			<u> </u>			<u> </u>	I		I	l	<u> </u>	<u> </u>		١		I		l	l	l 	<u> </u>	ac	c		ı	<u> </u>			1	I	1	1
2	١	S3	l		ı					1	<u> </u>	l		l	l	I	1	١	١		ا		l	l 		<u> </u>	ı	ı		1	<u> </u>			1	1	1	1
3	l		S4		1	<u> </u>	1			1	<u> </u>	l		l	l	1	1	ı	١		ا		l	l 		<u> </u>	1	1			<u> </u>			1	l		1
4	١				S6		ı	ı		1	<u> </u>	l		l	l	1	1	١	١		ا		l	l 		<u> </u>	ı	ı	5		ı		l	ı	1		1
5	١			S7	ı		ı			1	<u> </u>	l		l	l	١		١	١		ا		l	l		<u> </u>	I	ı		1	1		1	1			1
6	١	S10	r10		S6	r	10	S14		S15	5	r10		l	l	I	1	١	١		ا		l	l 		S1	6	ı	11	8	9	12	13	ı	1		1
7	١				ı	l	I	ı		1		I		ı		١		١	١	I	ا		l			<u> </u>	r1			ı	I		ı	ı			1
8	١		S18	1	١	S	17	I	١	1	I	I		I	I	١	١	١	١	I	١		l	l	I	ı	I	ı	l	I	I	l	I	I	I	1	1
9	I		r3	I	ı	r:	3	ı	ı	I	I	I	I I	I	I	I	ı	ı	ı	ı	I		ı		I	ı	ı	ı	ı	ı	ı	ı	ı	I	I	I	I
10	Ī			I	ı	I	S	19	ı	I	I	I	I I	I	I	I		ı	ı	I			ı		ı	ı	ı	ı	ı	ı	ı	ı	ı	ı	ı	I	Ī
11	ī		r6	I	ı	r	6	ı	1	ı	1	r6	I I	ı	I	ı	1	ı	ı	I			ı			1	ı	ı	ı	ı	ı	ı	I	ı	1	I	Ī
12	Ī		r7		ı	r	7	1	1	ı	1	r7	I I	ı		I	1	ı	Ī	I			ı			1	ı	ı		ı	ı		I	ı	I		Ī
13	Ī		r8	ı	ı	r	В	ı	1	ı	ı	r8	I I	ı	I	1	1	ı	ı	1			Ι			1	ı	ı	1	ı	1		Ι	1	1	I	ī
14	ī	S23		ı	ı	ı	ı	ı	1	ı	ı	ı	I I	ı	ı	ı	ı	ı	ı	1		S24	S25	ı		ı	ī	ı		ı	1		ı	20	21	22	ī
15	ī	S23		ı	ı	ı	ı	ı	1	ı	ı	ı		ı	ı	ı	ı	ı	ı	1		S24	S25	l	ı	ı	ı	ı		ı	ı	ı	ı	26	21	22	ï
16	ī	S27		ı	ı	ı	ı	ı	1	1	ı	ı	I I	ı	ı	ı	1	ı	ı	1			ı	ı	1	ı	ı	ı		ı	ı	1	ı	1	ı	ı	ī
17	Ī		r2	r2	ı	r	2	ı	1	1	ı	r2		ı	ı	ı	ı	ı	ı	1			ı	ı	ı	1	ı	ı		ı	ı	ı	1	ı	ı	ı	ī
18	Ī	S10	r10		S6	r:	10	514	1	S15	5	r10	I I	ı	l	1	1	ı	ı	1			1		Ι	51	6	ı	11	ı	28	12	13	1	1	ı	ī
19	Ī	S23		1	1	1	1	ı	1	ı	ı	ı	I I	ı	l	1	1	ı	ı	1		524	S25		Ι	1	ı	ı		ı	Ι	1	I	1	29	22	ī
20	Ī				ı		1	1	53	0	ı	ı	I I	1		1	1	ı		1			I		Ι	1	ı	ı		1	Ι	1	I	1	1	ı	ī
21	Ī				1	1	1	1	1	1	ı	ı	I I	531	S3	2 53	3 53	4 S	35 9	36	S37		I		1	1	1	ı		1	Ι	1	I	1	ī	ı	ī
22	Ī		r22		ı	r	22	1	r2	2	r22	r22	r22 r22	r22	r2	2 r2	2 r2	2 r	22 1	22	r22		I	r22	2	1	ı	ı		ı	Ι	1	I	1	ı	ı	ī
23	ı		r23		1	r	23	1	r2	3	r23	r23	r23 r23	r23	r2:	3 r2	3 r2	3 r	23 1	23	r23		l	r23	3	1	1	ı		1	Ι	1	I	1	ı	1	ī
24	1		r24		ı	r	24	Ι	r24	4	r24	r24	r24 r24	r24	r2	4 r2	4 r2	4 r	24 1	24	r24			r24	4		ı	ı		Ι	Ι		I	Ι	ı	Ι	ī
25		S23		1	ı	ı	ı	Ι	ı	ı	ı	ı	I I	Ι	 	1	1		I		l	S24	S25	 	Ι		ı	ı		Ι	Ι	Ι	I	Ι	38	22	ī
26			 	1			1		1	1	S39	ı	I I	Ι	 	1	1		I	I	l		 I	 	I			1		Ι		1	I	1	Ι	Ι	ī
27			 	1			S	40	1	1		ı	I I	Ι	 	1			I	I	 		 I	 	 		1	1		Ι		Ι	I	1	Ι	Ι	ī
28	I		r4	1		r	4	Ι	1	1		ı	I I	Ι	 	1	1		I	I	l		 I	 	I			1		Ι		1	I			 	ī
29			r5			r	5	I	1	1		r5	I I	I	 	S3	3 S3	4 S	35 9	36	 S37		 	 	I		1	1		Ι		1	I	1		 	ī
30	۱	S10	r10)	S6	r:	10	514	1	S15	 5	r10	I I	I	 	1			I	I	 		 I	 	I	51	.6	Ι	11	Ι	41	12	13		Ι	 	ī
131	1	5231		ı	1	1	1	1	1	1	1	1	1 1	1	1	ı	1	1	1			524	1525	ı	1		1	1	1	1	1	1	1	1	147	122	1

00	0																\overline{C}			ac	tion_	and	got	to.tx	~		9/9									319		
state	e id	;		begi	n end 	:=	whil	e do	if	the	n el	se te	o downt	:o >	<	+	5	te	lt	e	1,	A	1	CU	lC	n	am fo	or \$	S	compound_s	tmt stmt	s stm	t if_st	mt for_str	nt boo	l <u>ex</u> p	r facto	rl
10	<u>.</u>			<u> </u>		<u>. </u>	<u> </u>										!			<u>. </u>	-	٠				S2			1				-				<u> </u>	
1	1			<u> </u>		<u> </u>	1		1	1			1							<u> </u>	1	1						a	cc	1		1	1	1	1	1		
2	53			l		I	1	1	1	1	1	1	1	1	1					1	1	1	1				1			1			1	1	1		1	
3	1	54			1	1	1	1	1	1	1	1	1	1	1	_	_	_		1	1	1	. !	_				1	1	1		1	1	1	1	1	1	1
4	1	1 1		S6	1	1	1	1	1	1	1	1	1	1	1	1	1	1		1	1	1	1	1	-		1	1	1	5		1	1	1	1	1	1	1
5	1	1 1	57	1	1	1	1	1	1	1	1	1	1	1	1	1	1			1	1	1	1	1			1	1	1	1		1	1	1	1	1	1	1
16	510	r10		56	r10	1	S14	1	51	5	r:	10	1	-1	I	-	- 1			1	1	1	I	- 1			Si	16	- 1	11	8	19	12	13		1	1	
17	1	1 1		1	1	1	1	1	1	1	1	1	1	1	1	-1	-	-		1	1	1	1	- 1	1		1	Ir	1	1	1	1	1	1	1	1	1	1
8	1	518			S17	ī	1	1	1	1	1	1	1	1		1	I			1	1	1	T	1	ı			1	1	1	1	1	1	1		1	1	
19	1	r3		1	r3	1	1	1	1	1	1	1	1	1	1	1	Ī	I		1	1	ı	1	I	1		1	1	1	1		1	1	1		1	1	1
10	1	1		1	1	519	1	1	1	1	1	1	1	1	1	1	ī	ı		1	1	1	1	1	1		1	1	1	1		1	1	1	1	1	1	1
11	ī	r6		1	r6	ı	1	1	1	1	16	1	1	1	1	1	1	i		1	1	1	1	1	1		1	1	1	1		1	1	ī	1	1		ī
12	1	r7		1	17	ı	1	1	1	1	17	1	1	1	1	1	ī	1		ī	1	1	1	1	1		1	1	1	1	1	1	1	1		1	1	
13	1	r8			r8	1	1	1	1	ī	18		1	1	1	1	ī	I		1	1	1	1		1		1	1	1	1	1	1	1	1		1	1	ī
14	523	31 1		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		1	1	152	24 5	25	1		1	1	1	1		1	1	1	120	21	22	1
15	523	3		1	1	1	1	1	1	1	I	1		1	1	1	1	1		1	1	152	24 5	25	1		ī	1	1	1	1	1	1	1	26	21	22	ī
16	527	71 1		1	1	1	1	1	1	1	1	1	1	1	1	1	1			1	1	ī	1	ī	1		1	1	1		1	1	1	1	1	1	1	1
17	1	r2	r2	1	r2	1	1	1	1	1	r2		1	1	1	1	T			1	ī	1	T	1	1		1	1	1	1		1	1	1		1	1	1
18	510) r10		S6	r10	1	514	1	51	5	r:	10	1	1	1	1	1			ī	ī	1	ī	1	ī		51	16	1	11		28	12	13		1	1	
19	523	3		1	1	1	1	1	1	1	1	1	1	1	1	1	1			1	1	152	24 5	25	1		1	1	1			ī	1	1	1	29	22	
120	1			1	1	1	1	530	01	1	1		1	1	1	1	1	ī		1	1	1	ī	1	1		ī	1	1	1		1	1	1	1	1	1	
21	1	1		1	1	1	1	1	1	1	1	1	1	IS	31 5	32 5	33	534	535	53	5 53	7		1	1		T	1	1	1		1	1	1	1	1		
22	1	r22		1	r22	1	1	r22	2	r22	Ira	2 r	22 r22	Ir	22 r	22 r	22	r22	r22	r2	2 r2	2	1	1	22		1	1	1	1		1	1	1	1	1	1	
23	1	r23		1	r23	1	1	r23	3	r23	r2	3 r	23 r23	In	23 r	23 r	23	r23	r23	r2	3 r2	3	1	1	23		1	1	1	1	1	1	1	1	1	1	1	
24	1	r24			r24	1	1	r2	4	r24	r2	4 r	24 r24	r2	24 r	24 r	24	r24	r24	r2	4 r2	4	1	1	-24		1	1	1	1	1	1	1	1		1	1	
25					1		1	1	1	1		1		1	1	1		1						25					1		1	1	1	1			22	
126					1	1	1		1	539				1	1	1									-		1	1	1	1		1	1	1		1		
27					<u> </u>			<u> </u>				<u>-</u> -		1	I	1				1	1	1	1						<u>-</u> -			1						-
28					14				1	1	1			1	1	1				1	1	1	1									1			1	1		
29								-			Ir	5				15	331	5341	S35	153	5 53	71										1		1		1		
130																				1										11		141	12	13		-		
131															-					1	-	IS	2419	251						1		1		1		142	122	
			200											100			1																					

Figure: Action and Goto

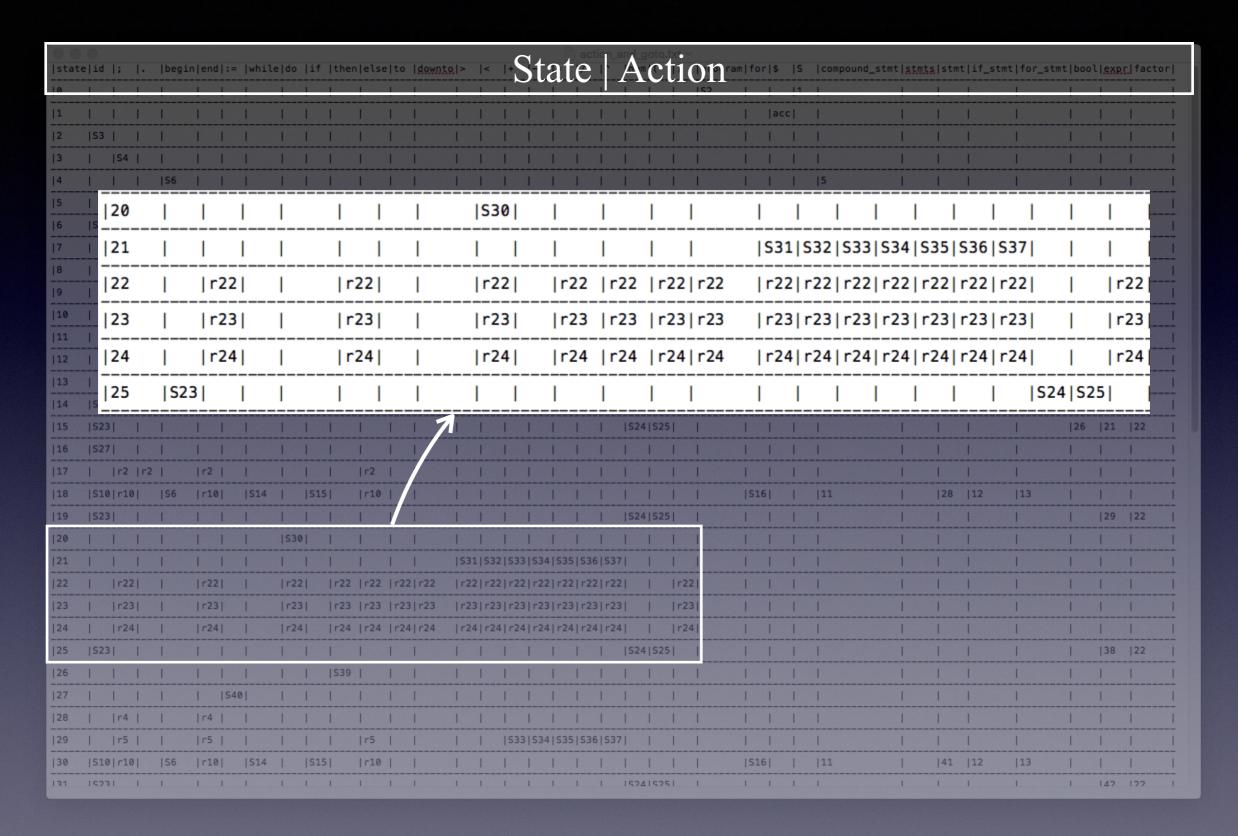


Figure: Action and Goto

• Generating output files in build/folder, using Shell scripts.

```
ChenMac:pascal-compiler wasdns$ ls
LICENSE
                                                 run_lexer.py
                        docs
                                                 run_lexer_ply.py
                        lexer_tests
README
build
                        parser_tests
                                                 run_parser_demo.sh
                        ply_frontend
cleanup.sh
                                                 src
[ChenMac:pascal-compiler wasdns$ ./run_parser_demo.sh
ChenMac:pascal-compiler wasdns$ cd build/
ChenMac:build wasdns$ ls
action_and_goto.txt
                        grammar.txt
                                                 parser
error.txt
                                                 slr.txt
                        input.txt
first_and_follow.txt
                        output.txt
ChenMac:build wasdns$
```

Results of running parser

```
1.first_and_follow.txt: Print the first set and follow set2.action_and_goto.txt: Print the action and goto table3.output.txt: Print the runtime stack and applied actions
```

```
output.txt ~
(125)
stateStack: 0 2 3 4 6 8 18 16 27 40 51 54 57 59 12
symbolStack: program id ; begin stmts ; for id := expr downto expr do if_stmt
INPUT: end . $
根据stmt-> if_stmt规约
stateStack: 0 2 3 4 6 8 18 16 27 40 51 54 57 59 61
symbolStack: program id ; begin stmts ; for id := expr downto expr do stmt
INPUT: end . $
根据for_stmt-> for id := expr downto expr do stmt规约
(127)
stateStack: 0 2 3 4 6 8 18 13
symbolStack: program id ; begin stmts ; for_stmt
INPUT: end . $
根据stmt-> for_stmt规约
stateStack: 0 2 3 4 6 8 18 28
symbolStack: program id ; begin <u>stmts</u> ; stmt
INPUT: end . $
根据stmts-> stmts ; stmt规约
stateStack: 0 2 3 4 6 8
symbolStack: program id ; begin stmts
INPUT: end . $
移入
(130)
stateStack: 0 2 3 4 6 8 17
symbolStack: program id ; begin stmts end
根据compound_stmt-> begin stmts end规约
stateStack: 0 2 3 4 5
symbolStack: program id ; compound_stmt
INPUT: . $
移入
stateStack: 0 2 3 4 5 7
symbolStack: program id ; compound_stmt .
INPUT: $
根据S-> program id ; compound_stmt .规约
stateStack: 0 1
symbolStack: S
INPUT: $
accept
```

Figure: Runtime Output

```
No. | State Stack | Symbol Stack | Input | Action
INPUT: end . $
根据stmt-> if_stmt规约
stateStack: 0 2 3 4 6 8 18 16 27 40 51 54 57 59 61
symbolStack: program id ; begin stmts ; for id := expr downto expr do stmt
根据for_stmt-> for id := expr downto expr do stmt规约
stateStack: 0 2 3 4 6 8 18 13
symbolStack: program id ; begin stmts ; for_stmt
INPUT: end . $
根据stmt-> for_stmt规约
stateStack: 0 2 3 4 6 8 18 28
symbolStack: program id ; begin stmts ; stmt
INPUT: end . $
根据stmts-> stmts ; stmt规约
stateStack: 0 2 3 4 6 8
\begin{tabular}{ll} symbol Stack: program id ; begin $\pm mts \\ INPUT: end . $ \\ \end{tabular}
移入
stateStack: 0 2 3 4 6 8 17
symbolStack: program id ; begin stmts end
根据compound_stmt-> begin stmts end规约
stateStack: 0 2 3 4 5
symbolStack: program id ; compound_stmt
INPUT: . $
移入
stateStack: 0 2 3 4 5 7
symbolStack: program id ; compound_stmt .
INPUT: $
根据S-> program id ; compound_stmt .规约
stateStack: 0 1
symbolStack: S
INPUT: $
accept
```

Figure: Runtime Output

```
No. | State Stack | Symbol Stack | Input | Action
INPUT: end . $
根据stmt-> if_stmt规约
stateStack: 0 2 3 4 6 8 18 16 27 40 51 54 57 59 61
symbolStack: program id ; begin stmts ; for id := expr downto exp
                                                 (128)
根据for_stmt-> for id := expr downto expr do stmt规约
                                                 stateStack: 0 2 3 4 6 8 18 28
stateStack: 0 2 3 4 6 8 18 13
                                                 symbolStack: program id ; begin stmts ; stmt
symbolStack: program id ; begin stmts ; for_stmt
INPUT: end . $
                                                 INPUT: end . $
根据stmt-> for stmt规约
                                                 根据stmts-> stmts ; stmt规约
stateStack: 0 2 3 4 6 8 18 28
symbolStack: program id ; begin stmts ; stmt
INPUT: end . $
                                                 (129)
根据stmts-> stmts ; stmt规约
                                                 stateStack: 0 2 3 4 6 8
stateStack: 0 2 3 4 6 8
                                                 symbolStack: program id ; begin stmts
symbolStack: program id ; begin stmts
INPUT: end . $
                                                 INPUT: end . $
                                                 移入
stateStack: 0 2 3 4 6 8 17
symbolStack: program id ; begin stmts end
                                                 (130)
根据compound_stmt-> begin stmts end规约
                                                 stateStack: 0 2 3 4 6 8 17
stateStack: 0 2 3 4 5
                                                 symbolStack: program id ; begin stmts end
symbolStack: program id ; compound_stmt
                                                 INPUT: . $
移入
                                                 根据compound_stmt-> begin stmts end规约
stateStack: 0 2 3 4 5 7
symbolStack: program id ; compound_stmt .
INPUT: $
```

Figure: Runtime Output

根据S-> program id ; compound_stmt .规约

stateStack: 0 1 symbolStack: S INPUT: \$ accept Generating output files in build/ folder, using Shell scripts.

```
ChenMac:pascal-compiler wasdns$ ls
LICENSE
                                                 run_lexer.py
                        docs
                                                 run_lexer_ply.py
                        lexer_tests
README
build
                        parser_tests
                                                 run_parser_demo.sh
cleanup.sh
                        ply_frontend
                                                 src
ChenMac:pascal-compiler wasdns$ ./run_parser_demo.sh
ChenMac:pascal-compiler wasdns$ cd build/
ChenMac:build wasdns$ ls
action_and_goto.txt
                        grammar.txt
                                                 parser
                        input.txt
error.txt
                                                 slr.txt
first_and_follow.txt
                        output.txt
ChenMac:build wasdns$
```

Results of running parser

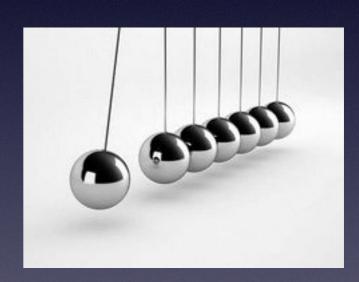
```
1.first_and_follow.txt: Print the first set and follow set
2.action_and_goto.txt: Print the action and goto table
3.output.txt: Print the runtime stack and applied actions
4.error.txt: Print runtime error log, while continuing analyzing.
```

```
int main() {
582
         char gramarFile[50] = "./grammar.txt";
583
         char outputFile[50] = "./output.txt";
584
         char actionAndGotoFile[50] = "./action_and_goto.txt";
585
         char DFAFile[50] = "./slr.txt";
586
         char errorFile[50] = "./error.txt";
587
         char inputFile[50] = "./input.txt";
588
589
         init();
590
         getGrammar(gramarFile);
591
         getCanonical();
592
593
594
         calFirst();
         printFirst();
595
         calFollow();
596
         printFollow();
597
598
         printDFA(DFAFile);
599
         setActionAndGoto();
600
601
         printActionAndGoto(actionAndGotoFile);
         readInput(inputFile);
602
         solve(outputFile, errorFile);
603
604
         return 0;
605 }
```

Figure: Overview of Main Procedure

Outline

- 1.Introduction
- 2.Journal of testing
- 3.Conclusion



Conclusion

- Introduction of SLR(1) and Our Program;
- Giving an Example of Testing Our Program.



Experiences

- · Requiring enough time to complete this task.
- Fully understanding of parser mechanism.
- The other things are coming soon. Stay Tuned!



That's all. Thank you!

Group Members: 吴媛媛, 林诗尧, 陈翔