CS 3342 - ASN 1

Date

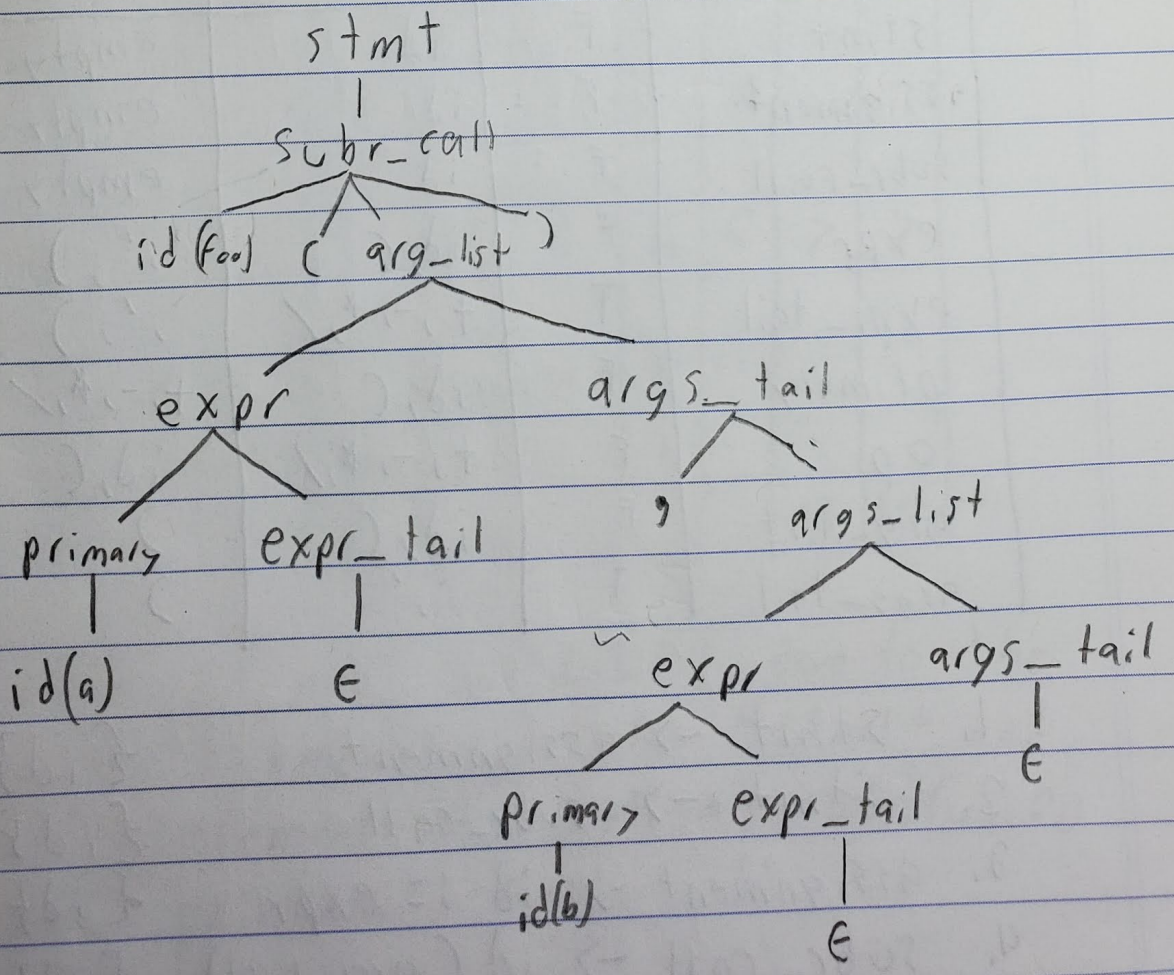
1a) "(¬(", ENTER)|(\ENTER))*"

- strings have to start and end with "
- can have any char except from the set {" and ENTER}
  or can be \" or \ENTER

b) ([A-Za-z] ¬(File, For, from))*

- Strings can have any combo of upper and lower case except for the words for, from file

2a)



Date

b) stmt -> subr_call
-> id (args_list)
-> id (expr args_tail)
-> id (expr, args_list)
-> id (expr, expr args_tail)
-> id (expr, expr)
-> id (expr, primary expr_tail)
-> id (expr, primary)
-> id (expr, id)
-> id (primary expr_tail, id)
-> id (primary, id)
-> id (id, id)

c)

	EPS	First-set	Follow-set
stmt	F	id	empty
assignment	F	id	empty
subr_call	F	id	empty
expr	F	id, (	;, ,)
expr_tail	T	+, -, *, /	;, ,)
primary	F	id, (	+, -, *, /
op	F	+, -, *, /	id, (
arg_list	F	id, (	)
args_tail	T	,	)

1. stmt -> assignment {id}
2. stmt -> subr_call {id}
3. assignment -> id := expr {id}
4. subr_call -> id (arg_list) {id}

Date

5.	expr -> primary expr_tail	{id, (}
6.	expr_tail -> op expr	{+, -, *, /}
7.	expr_tail -> ∈	{;,)}
8.	primary -> id	{id}
9.	primary -> subr_call	{id}
10.	primary -> (expr)	{(}
11.	op -> + | - | * | /	{+, -, *, /}
12.	arg_list -> expr args_tail	{id, (}
13.	args_tail -> , arg_list	{,}
14.	args_tail -> ∈	{)}

Parser table

	id	:=	(	+	-	*	/	,	)
stmt	1,2								
assignment	3								
subr_call	4								
expr	5		5						
expr_tail	-			6	6	6	6	7	7
primary	8,9		10						
op	-			11	11	11	11		
args_list	12		12						
args_tail	-							13	14

Grammer is not in LL(1) due to the confusion regarding the "stmt" and "id" statements. As well as confusion of "primary" and "id"

Date

d) Factor out "id" from productions with "stmt" and "expr"

stmt → id stmt_tail
stmt_tail → := expr
stmt_tail → (arg_list)
expr → primary expr_tail
expr_tail → op expr
expr_tail → ϵ
primary → id primary_tail
primary_tail → ϵ
primary_tail → (args_list)
primary → (expr)
op → + | - | * | /
args_list → expr args_tail
args_tail → , arg_list
args_tail → ϵ

3a i) since $\bar{E}_s$ expands to ϵ
∴ EPS($\bar{E}_s$) = true

ii) First($\bar{E}_s$)
for $\bar{E}_s \rightarrow \bar{E}\ \bar{E}_s$
FIRST($\bar{E}_s$) ≐ FIRST($\bar{E}$)
for $\bar{E}$ → atom
FIRST($\bar{E}$) = atom
for $\bar{E} \rightarrow$ '$\bar{E}$
FIRST($\bar{E}$) = '

Date

for $\bar{E} \rightarrow (\bar{E}\ \bar{E}s)$

$FIRST(\bar{E}) = ($

for $\bar{E}s \rightarrow \epsilon$

$FIRST(\bar{E}s) = \epsilon$

$\therefore FIRST(\bar{E}s) = \{atom, ', (, \epsilon\}$

iii) $Follow(\bar{E})$

for $\bar{E} \rightarrow \bar{E}\ \$\$$

$Follow(\bar{E}) = \$\$$

for $\bar{E} \rightarrow 'E$

$follow(\bar{E}) = Follow(E)$

for $\bar{E} \rightarrow (E\ Es)$

$Follow(\bar{E}) = First(Es) = atom, (, '$

$follow(E) = follow(\bar{E})$

for $\bar{E}s \rightarrow \bar{E}\ \bar{E}s$

$follow(\bar{E}) = follow(\bar{E}s)$

$follow(\bar{E}) = First(\bar{E}s) = atom, (, '$

for $E \rightarrow (\bar{E}\ \bar{E}s)$

$follow(\bar{E}s) = First()) =)$

For $\bar{E}s \rightarrow \bar{E}\ \bar{E}s$

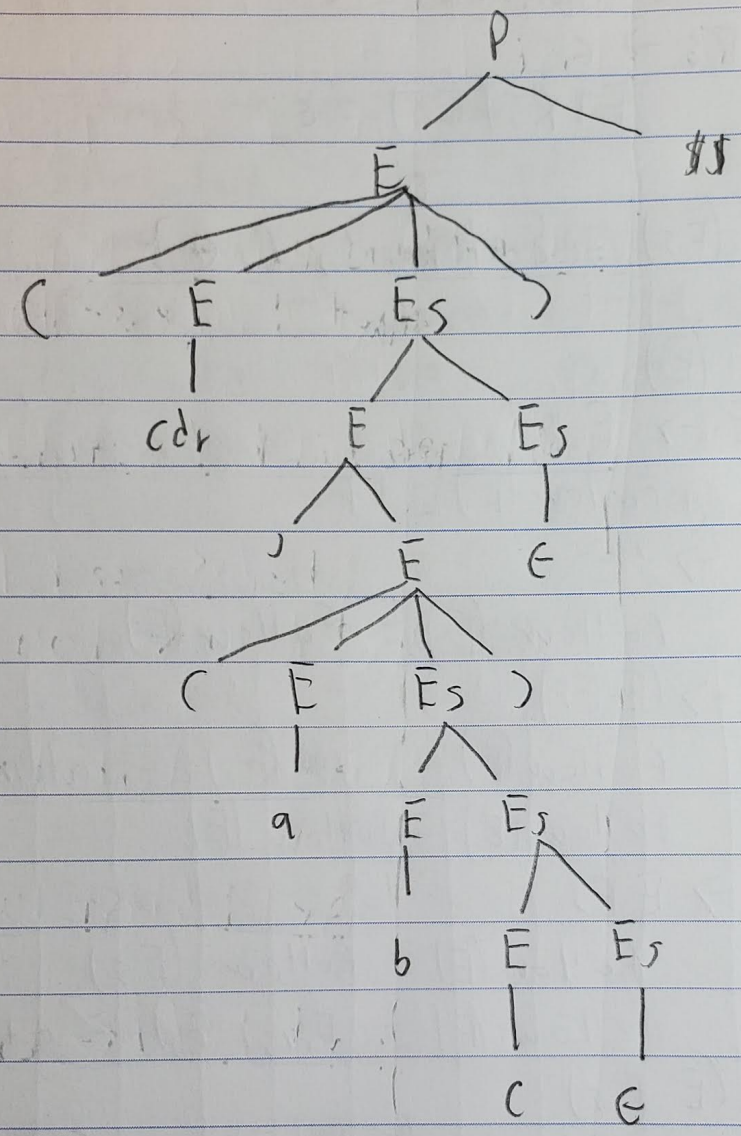
$follow(\bar{E}s) = Follow(\bar{E}s) =)$

$\therefore Follow(\bar{E}) = \{\$\$, atom, ', (,)\}$

iv) $Predict(\bar{E}s \rightarrow \epsilon) = \{\ )\}$

Date

3 b) Parse tree for (cdr '(a,b,C)) $$



C) P → E $$
→ (E Es) $$
→ (cdr Es) $$
→ (cdr E Es) $$
→ (cdr '(E Es) Es) $$
→ (cdr '(a Es) Es) $$
→ (cdr '(a E Es) Es) $$
→ (cdr '(a b Es) Es) $$
→ (cdr '(a b E Es) Es) $$

Page

Date

→ (cdr '(a b c Es) Es) $$
→ (cdr '(a b c) Es) $$
→ (cdr '(a b c)) $$

d)

Parse stack	input stream	comment
P	(cdr '(a b c)) $$	
E $ $	(cdr '(a b c)) $$	predict P → E $ $
(E Es) $ $	(cdr '(a b c)) $$	predict E → (E Es)
E Es) $ $	cdr '(a b c)) $$	match (
atom Es) $ $	cdr '(a b c)) $ $	predict E → atom
Es) $ $	'(a b c)) $ $	match atom
E Es) $ $	'(a b c)) $ $	predict Es → E Es
' E Es) $ $	'(a b c)) $ $	predict E → ' E
E Es) $ $	(a b c)) $ $	match '
(E Es) Es) $ $	(a b c)) $ $	predict E → (E, Es)
E Es) Es) $ $	a b c)) $ $	match (
atom Es) Es) $ $	a b c)) $ $	predict E → atom
Es) Es) $ $	b c)) $ $	match atom
E Es) Es) $ $	b c)) $ $	predict Es → E Es
atom Es) Es) $ $	b c)) $ $	predict E → atom
Es) Es) $ $	c)) $ $	match atom
E Es) Es) $ $	c)) $ $	predict Es → E Es
atom Es) Es) $ $	c)) $ $	predict E → atom
Es) Es) $ $	)) $ $	match atom
) Es) $ $	)) $ $	predict Es → ϵ
Es) $ $	) $ $	match)
) $ $	) $ $	predict Es → ϵ
$ $	$ $	match)

Date

4.

	State	Transition
0	decl_list -> • decl_list decl ;	on decl_list shift goto 2
	decl_list -> • decl ;	on decl shift goto 1
	decl -> • id : type	on id shift goto 3
1.	decl_list -> decl • ;	on ; shift, reduce (Pop 2 states push decl_list
2.	decl_list -> decl_list • decl ;	on decl shift goto 13
	decl -> • id : type	on id shift goto 3
3.	decl -> id • : type	on : shift goto 4
4.	decl -> id : • type	on type shift (reduce Pop 2 states, push decl
	type -> • int	on int shift, reduce (pop 1, state push type)
	type -> • real	on real shift reduce (pop 1, stat push type)
	type -> • char	on char shift, reduce (pop 1 state push type)
	type -> • array const .. const of type	on array shift goto 5
	type -> • record decl_list end	on record shift goto 11
5.	type -> array • const .. const of type	on const shift goto 6
6.	type -> array const • .. const of type	on shift and goto 7

Page

7.	type → array const . . const of type	on . shift goto 8
8.	type → array const .. . const of type	on const shift goto 9
9.	type → array const .... const . of type	on of shift goto 10
10.	type → array const .... const of . type	on type shift and reduce (pop 5 states push type)
	type → . int	on int shift and reduce (pop 1 state push type)
	type → . real	on real shift and reduce (pop 1 state push type)
	type → . char	on char shift and reduce (pop 1 state push type)
	type → . array const .... const of type	on array shift goto 7
	type → record decl_list end	on record shift goto 11
11.	type → record . decl_list end	on decl_list shift goto 12
	decl_list → . decl ; decl ;	on decl_list shift goto 2
	decl_list → . decl ;	on decl shift goto 1
	decl → id - type	on id shift goto 3
12.	type → record decl_list . end	on end shift reduce (pop 3 states, push type)
13.	decl-list → decl_list decl . ;	on ; shift and reduce (pop 3 states, push decl_li

	Parse stack	input stream	Date Comments
0		foo ; record	
0	foo 3	; record a	shift foo
0	foo 3 ; 4	record a ;	shift ;
0	foo 3 ; 4 record 11	a : char	shift record
0	foo 3 ; 4 record 11 a	; char ;	shift a
	3		
0	foo 3 ; 4 record 11 a	char ; b	shift ;
	3 ; 4		
0	foo 3 ; 4 record 11	type ; b	shift char reduce by
	a 3 ; 4		type → char
0	foo 3 ; 4 record 11	decl ; b	shift type reduce by
			decl → id : type
0	foo 3 ; 4 record 11	; b ;	shift decl
	decl 1		
0	foo 3 ; 4 record	decl_list b ;	shift ; and reduce by
	11		decl_list → decl ;
0	foo 3 ; 4 record 11	b ; array	shift b decl_list
	decl_list 0		
0	foo 3 ; 4 record 11	; array1	shift b
	decl_list 0 b 3		
0	foo 3 ; 4 record 11	1 ..	shift array
	decl_list 0 b 3 ; 4		
	array 5		
0	foo 3 ; 4 record 11	.. 2	shift 1
	decl_list 0 b 3 ;		
	4 array 5 16		
0	foo 3 ; 4 record 11	. 2 of	shift .
	decl_list 0 b 3 ; 4		
	array 5 16 . 7		
	of type		

Date

foo 3;4 record 11 decl_list 0b 3;4 array 516,7,8	2 of real	shift 6
foo 3;4 record 11 decl_list 0b 3;4 array 516,7,829	of real;	shift 2
foo 3;4 record 11 decl_list 0b 3;4 array 516.7,829 of 10	real; end	shift of
foo 3;4 record 11 decl_list 0b 3;4 array 516,7,829 of 10	type; end	shift real reduce by type -> real
foo 3;4 record 11 decl_list 0b 3;4	type; end	shift type reduced by type -> array const ... const of type
foo 3;4 record 11 decl_list 0b 3;4	decl; end	shift type and reduce by decl -> id ; type
foo 3;4 record 11 decl_list 0 decl1	; end;	shift decl
0 foo 3;4 record 11	decl_list end;	shift ; and reduce by decl_list -> decList decl;
foo 3;4 record 11 decl_list 12	end;	shift decl_list
foo 3;4	type;	Shift end and reduce by type -> record decl_list end
	decl;	

Date

0	decl ;	shift type and reduce by decl → id : type
0 decl 1	;	shift decl
0	decl_list	shift ; and reduce by decl_list → decl
[done]		