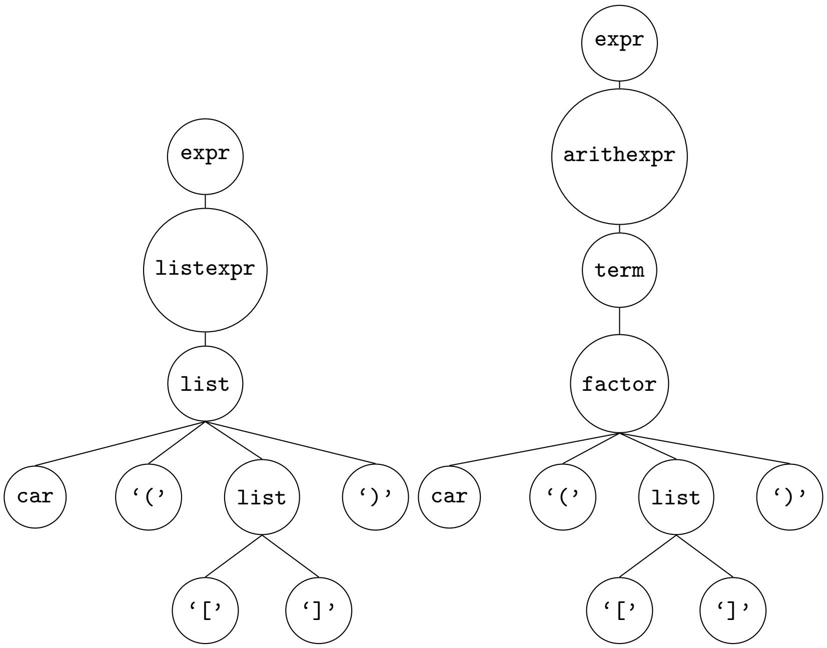
CS 550: Programming Languages
Final

Dustin Ingram

June 15, 2012

1 Grammars, Attributes, and Ambiguity

1. **Solution:** There are two possible derivations for `car([])`:



2. **Solution:**

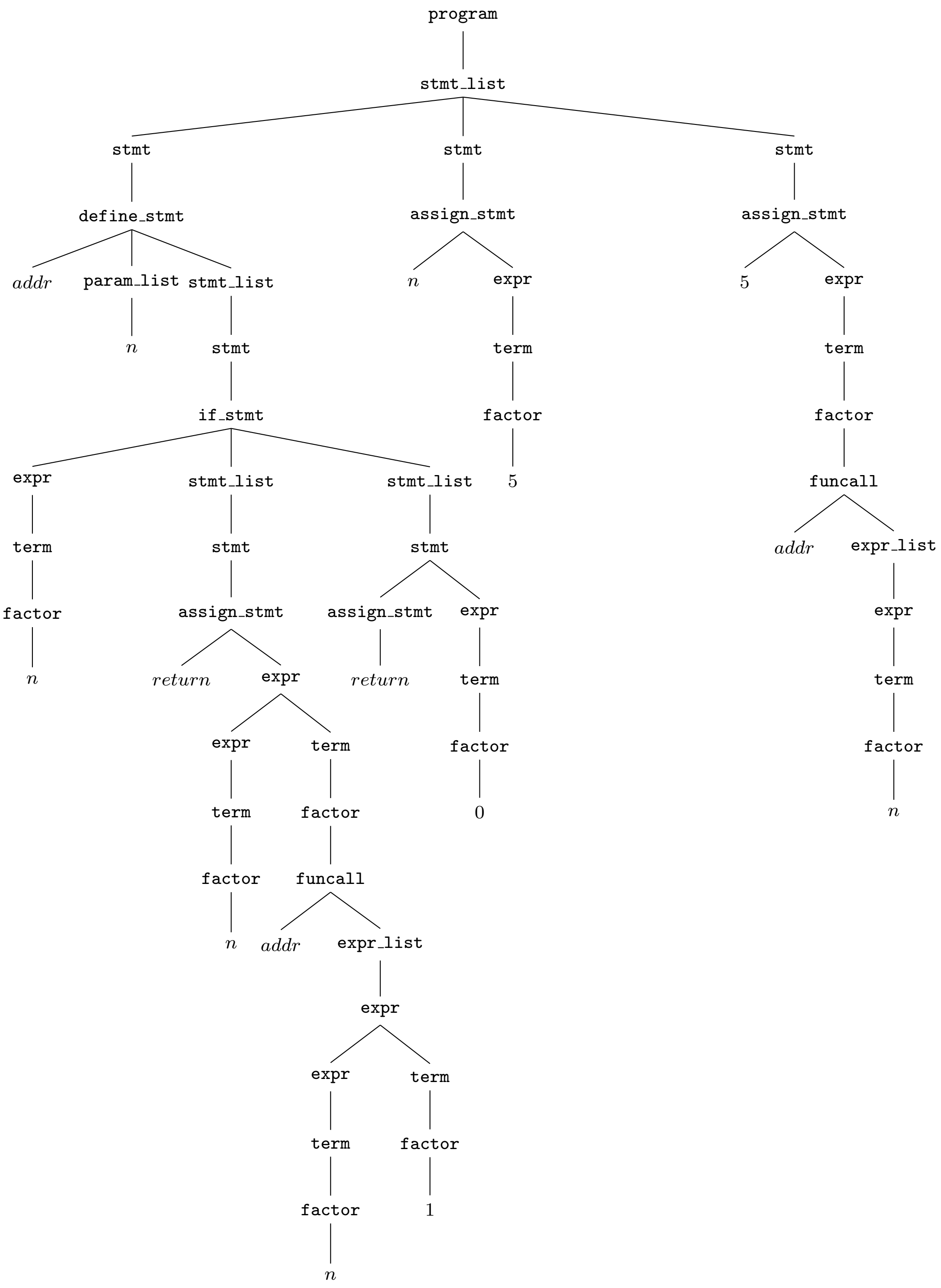
FIRST:	
P	atom, ‘, (
E	atom, ‘, (
Es	atom, ‘, (

FOLLOW:	
P	*none*
E	atom, ‘, (,), $$
Es	)
$$	*none*
atom	atom, ‘, (, $$
‘	atom, ‘, (
(	atom, ‘, (
)	atom, ‘, (,), $$

PREDICT:	
P → E $$	atom, ‘, (
E → atom	atom
E → ‘ E	‘
E → (E Es)	(
Es → E Es	atom, ‘, (
Es → ε	)

2 Mini Language Interpreter

The program data structure is as follows:



As the `eval` method for the program object is called, this program structure is built and the environment is modified accordingly, as the `eval` method is then called on each child object, until the leaves are reached. The resulting environment is:

ENV:

`addr`	*procedure*
`n`	5
`s`	15
`return`	15

3 Shift Reduce Parsing and Parser Generation

1. **Solution:**

CSFM for the Grammar:

0.	list → •(	on (shift and goto 2
1.	list → (•(	on (shift and goto 1
	list → (•)	on) shift and reduce (pop 1 state, push list on input)
	list → (•sequence	on sequence shift and goto 6
	list → (•number	on number shift and reduce (pop 1 state, push list on input)
	list → (•list	on list shift and reduce (pop 2 states, push list on input)
	list → (•listelement	on listelement shift and goto 4
2.	list → (•)	on) shift and reduce (pop 1 state, push list on input)
	list → (•(	on (shift and goto 1
	list → (•sequence	on sequence shift and goto 3
	list → (•number	on number shift and reduce (pop 1 state, push list on input)
	list → (•listelement	on listelement shift and goto 4
	list → (•list	on list shift and reduce (pop 2 states, push list on input)
3.	list → (sequence•)	on) shift and reduce (pop 2 states, push list on input)
4.	sequence → (listelement•,	on , shift and goto 5
	sequence → (listelement•)	on) shift and reduce (pop 2 state, push sequence on input)
5 .	sequence → (listelement,•sequence	on sequence shift and reduce (pop 1 state, push sequence on input)
	sequence → (listelement,•number	on number shift and reduce (pop 1 state, push sequence on input)
	sequence → (listelement,•listelement	on listelement shift and goto 4
	sequence → (listelement,•list	on list shift and reduce (pop 2 states, push list on input)
	sequence → (listelement,•(	on (shift and goto 1
6.	list → (sequence•)	on) shift and reduce (pop 2 states, push list on input)

2. **Solution:**

Parse stack	Input stream	Comment
0	(1, (2))	
0 (	1, (2))	on (shift and goto 2
0 (*number*	, (2))	on number shift and reduce
0 (*number*,	(2))	on , shift and goto 5
0 (*number*, (	2))	on (shift and goto 1
0 (*number*, (*number*	))	on number shift and reduce
0 (*number*, (*number*)	)	on) shift and reduce
0 (*number*, (*number*))		on) shift and reduce
done		

4 Mini Language Compiler

1. **Solution:**

```
A: Code(stmt-list)
 : Code(expr)
 : LD t
 : JMZ B
 : JMP A
B: ...
```

2. **Solution:**

Activation Record:	
Parameters:	$n = 3$
Local Variables:	
Temporary Variables:	T1, . . . , T6
Return Value:	6
Previous Frame Pointer:	10
Return Address:	50

5 Dynamic Memory Allocation and Garbage Collection

1. **Solution:** $L = ((2, 3), ())$

2. **Solution:** $\textit{Available} = 0 = (0, 1, 5, 6)$

6 Dynamic Arrays

1. **Solution:** Changes to the grammar: `expr` → `IDENT ‘[’ expr ‘]’`
   New classes: `CreateArray` and `DeleteArray` for the `array` and `delete` functions, and an `Array` class for the array.

2. **Solution:** The `eval` method of `CreateArray` allocates a new `Array` object with n elements, using available memory cells.

   The `eval` method of `DeleteArray`, marking it's respective memory cells as free.

   The `eval` method of `Array` would look up the corresponding Identifier, then return the value for the index provided.

3. **Solution:** The `CreateArray` function must validate that the array size is a positive integer and that there is sufficient memory to create it.

   The `DeleteArray` function must validate that it has been passed an existing `Array`

   The `Array` function must validate that the identifier is valid, and that the index is a positive integer and that it is within the bounds of the array.