

CS 410 Midterm Exam Solution  
Spring 2005 [Bono]

**Problem 1 [10 pts.]**

Short answer problems.

- A. During semantic analysis a symbol table is created and used. It is a lookup table that associates

identifier with type  
(the lookup key) (the info we want to know about the key)  
(Note: the answer "symbol" is worth no credit.)

- B. Describe a feature of Espresso that will require the compiler to make multiple passes over the AST during semantic analysis.

- Entities can be used before they are defined  
(eg. class names, fields, methods)

- C. Consider the problem of handling a multi-line Espresso comment in flex, such that we also keep an accurate count of the current line number for error reporting. Briefly describe a strategy for solving this problem (you do not have to describe the whole solution). Note: there is more than one correct answer.

- Pattern to recognize `"/*`
- Code in action has a loop to process rest of comment

alt. answer

On `"/*` pattern set a start condition and use special case patterns to recognize comment and count lines

- D. Name the category of parser built by bison. \_\_\_\_\_

(Note: the answer "table-driven" is worth no credit)

- bottom-up
- LR
- shift-reduce
- LALR

- E. Write a regular expression for strings of positive integers that start and end with a 3.

Example strings in the language: 3, 33, 303, 31237293

$$3 \left( \epsilon \mid (0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9)^* \right) 3$$

## Problem 2 [10 pts.]

Consider the following regular expression:

(Note:  $\epsilon$  denotes the empty string; + is used a symbol in the alphabet, not as a regular expression operator.)

$$\epsilon \mid a(+a)^*$$

**Part A.** Which strings are in the language denoted by the regular expression (circle the letters for all that apply):

a.  $\epsilon$

b. a

c. +a

d. a+a+

e. a+a+a

**Part B.** Write a grammar for the language denoted by the regular expression.

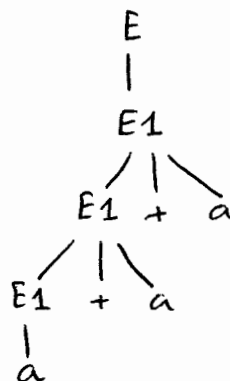
$$E \rightarrow \epsilon \mid E1$$

$$E1 \rightarrow a \\ \mid E1+a$$

**Part C.** Using your grammar from part B, show parse trees for each of the strings you circled in part A. (Note: as a check on your grammar's correctness you should not be able to derive any of the strings you did *not* circle.)

E  
|  
 $\epsilon$

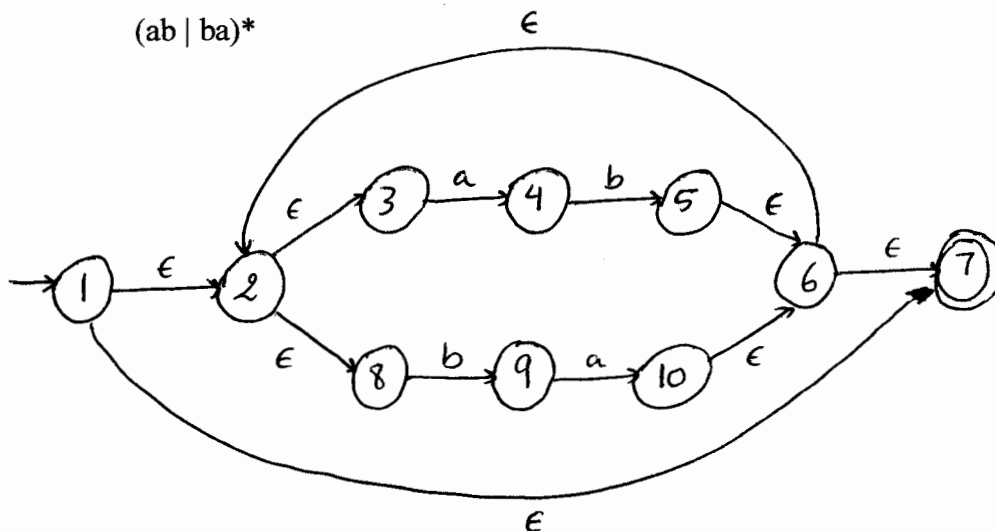
E  
|  
E1  
|  
a



**Problem 3 [15 pts.]**

**Part A.** Using Thompson's Construction, build an NFA equivalent to the regular expression given below.

$(ab \mid ba)^*$



**Part B.** Use subset construction to build a DFA equivalent to the NFA you gave in part A. Show your work.

	a	b
<u>[12378]</u>	[4]	[9]
[4]	-	[235678]
[9]	[23678,10]	-
<u>[235678]</u>	[4]	[9]
<u>[23678 10]</u>	[4]	[9]

### Problem 4 [8 pts.]

Give the FIRST and FOLLOW sets for each of the non-terminals in the following grammar:

A  $\rightarrow$  B C D

B  $\rightarrow$  w

| B x

C  $\rightarrow$  y C z

| m

D  $\rightarrow$  D B

| a

$$\text{FIRST}(D) = \{a\}$$

$$\text{FIRST}(C) = \{y, m\}$$

$$\text{FIRST}(B) = \{w\}$$

$$\text{FIRST}(A) = \text{FIRST}(B) = \{w\}$$

$$\text{FOLLOW}(A) = \{\$ \}$$

$$\text{FOLLOW}(B) = x \cup \text{FIRST}(C) \cup \text{FOLLOW}(D) = \{x, y, m, \$, w\}$$

$$\text{FOLLOW}(C) = z \cup \text{FIRST}(D) = \{a, z\}$$

$$\text{FOLLOW}(D) = \text{FOLLOW}(A) \cup \text{FIRST}(B) = \{\$, w\}$$

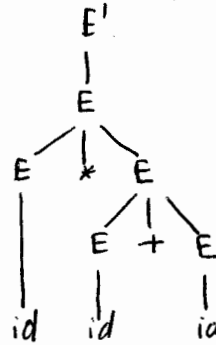
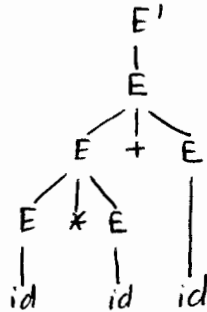
**Problem 5 [27 pts. total]**

Consider the following augmented grammar.

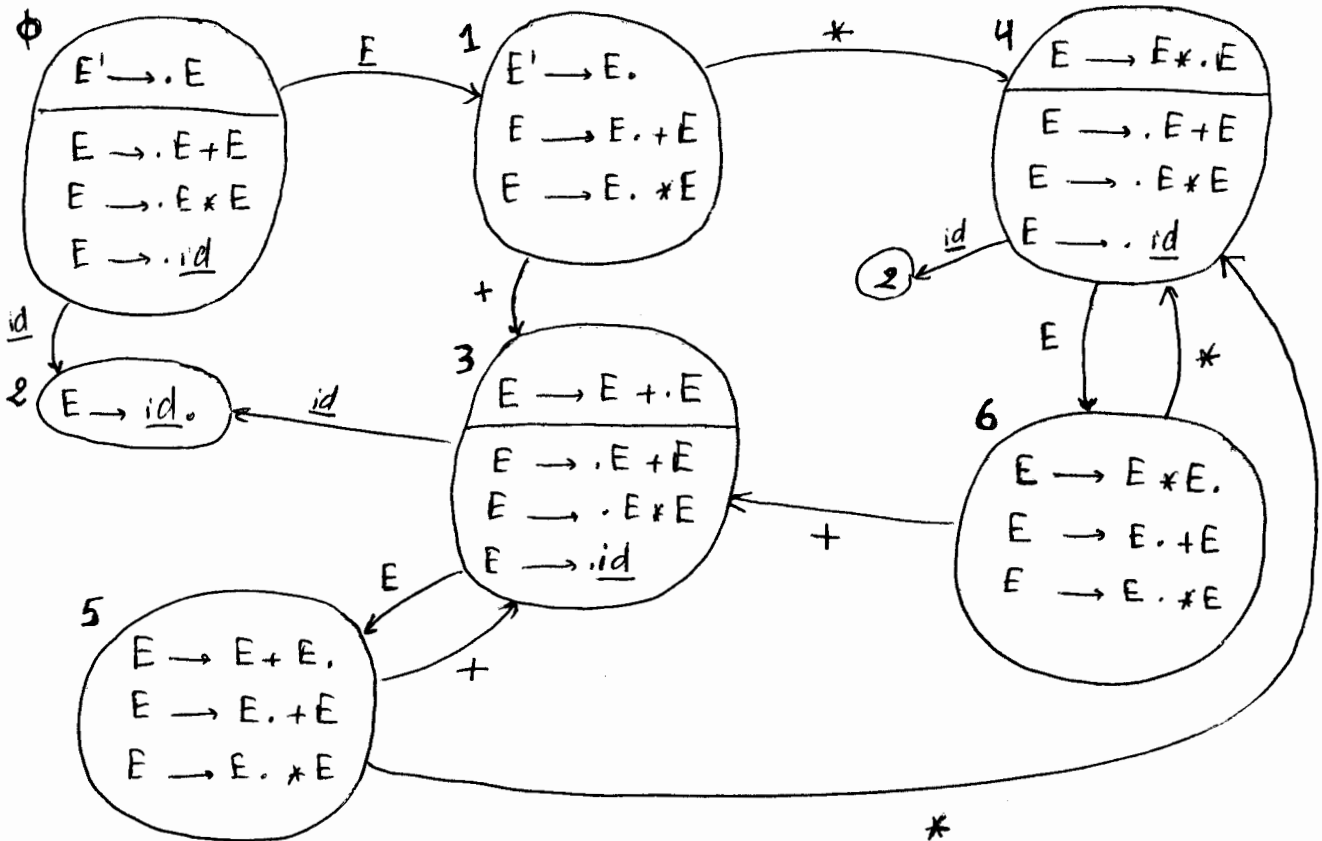
- 0)  $E' \rightarrow E$
- 1)  $E \rightarrow E + E$
- 2)  $E \rightarrow E * E$
- 3)  $E \rightarrow id$

**Part A [5].** Show that the grammar is ambiguous.

*(many correct answers)*



**Part B [10].** Build the DFA for recognizing viable prefixes for an SLR parser for the grammar. Show the set of items for each state. Note: the grammar has already been augmented for you. You may use the space below, or the back of the previous page (just direct us to look there).



### Problem 5 (cont.)

**Part C [8].** Show all the shift-reduce conflicts that would result from building SLR parser tables from this grammar. There are four: use the following table for your answer. Show your work.

Note: this problem is not asking you to build complete action/goto tables.

	<u>State</u>	<u>Input</u>	<u>Shift and go to state</u>	<u>Reduce by production</u>
R	1. <u>5</u>	<u>+</u>	<u>3</u>	<u><math>E \rightarrow E + E</math></u>
S	2. <u>5</u>	<u>*</u>	<u>4</u>	<u><math>E \rightarrow E + E</math></u>
R	3. <u>6</u>	<u>+</u>	<u>3</u>	<u><math>E \rightarrow E * E</math></u>
R	4. <u>6</u>	<u>*</u>	<u>4</u>	<u><math>E \rightarrow E * E</math></u>

**Part D [4].** To the left of each of the numbered shift-reduce conflicts you showed above, write a **S** or a **R** to show the correct interpretation for the normal precedence and associativity for + and \*. (Where **S** stands for shift, and **R** stands for reduce.) Hint: the items in the state indicate what is on the stack when we're in that state. E.g., one of them will correspond to  $E * E$  on the stack and a possible next input symbol is another \*; that would happen in the process of parsing input  $id * id * id$ .

(See above)

## Problem 6 [10 pts.]

Below is a grammar for the regular expression notation assuming an alphabet of

{a, b}. Thus, strings in the language are regular expressions.

Write semantic rules for the grammar to compute the `bool` attribute `E.eps`, which is true if the *language defined* by the regular expression given contains  $\epsilon$ , the empty string. To clarify, we aren't asking if the sentence we're deriving contains  $\epsilon$ , but rather the whether  $\epsilon$  is in the language denoted by the sentence when it is interpreted as a regular expression.

Here are a few sample sentences and values:

<u>sentence</u>	<u>E.eps</u>
ab*	false (the empty string <i>is not</i> in the language denoted by (a b)*)
(a b)*	true (the empty string <i>is</i> in the language denoted by (a b)*)

A note about the grammar below: Two symbols, | and  $\epsilon$ , that can normally appear in context-free grammar notation are *terminals* in the grammar below. To help clarify we put single quotes around every terminal in the grammar below, and we didn't use | to show alternate right-hand sides in the grammar. Thus, the first line below is just one production with three symbols on its RHS. And the last production below has a single terminal on its RHS (the empty string is not in the language defined by the grammar).

$$E \rightarrow E_1 \mid T \quad E.\text{eps} = E_1.\text{eps} \mid T.\text{eps};$$

$$E \rightarrow T \quad E.\text{eps} = T.\text{eps};$$

$$T \rightarrow T_1 F \quad T.\text{eps} = T_1.\text{eps} \ \&\& \ F.\text{eps};$$

$$T \rightarrow F \quad T.\text{eps} = F.\text{eps};$$

$$F \rightarrow E_1 \mid * \quad F.\text{eps} = \text{true};$$

$$F \rightarrow 'a' \quad F.\text{eps} = \text{false};$$

$$F \rightarrow 'b' \quad F.\text{eps} = \text{false};$$

$$F \rightarrow '\epsilon' \quad F.\text{eps} = \text{true};$$

$$F \rightarrow (E) \quad F.\text{eps} = E.\text{eps};$$