$Id: cmps104a-2016q4-exam1.mm,v 1.54 2016-10-20 13:54:07-07 - - $

page 1 page 2 page 3 page 4 Total / 42

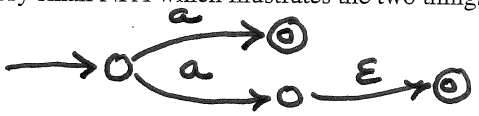
*Please print clearly :*

**Name :** SOLUTION

**Login :** @ucsc.edu

*No books ; No calculator ; No computer ; No email ; No internet ; No notes ; No phone. Neatness counts ! Do your scratch work elsewhere and enter only your final answer into the spaces provided. Points will be deducted for messy or unreadable answers.*

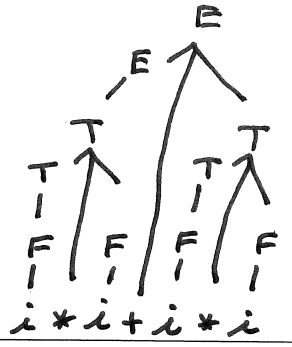
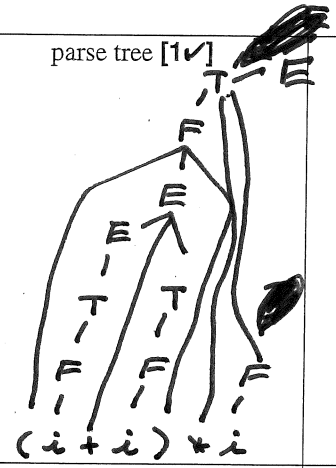
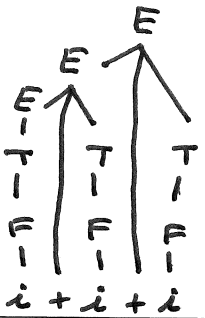
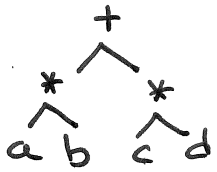
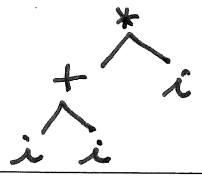
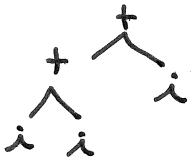
1. Draw a very small NFA which illustrates the two things that are prohibited in a DFA. **[1✓]**




ε-trans
2 out trans a

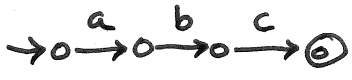
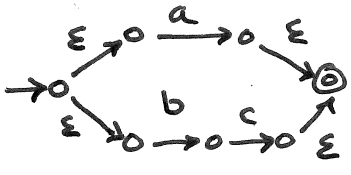
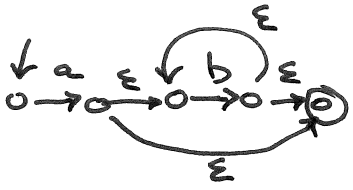
2. Given the ETF grammar at the left : (a) Draw parse trees in the upper box matching each lexical symbol in the middle box with an appropriate terminal symbol. (b) Draw abstract syntax trees in the lower box for each of the expressions.

$E \rightarrow E + T$
$E \rightarrow T$
$T \rightarrow T * F$
$T \rightarrow F$
$F \rightarrow ( E )$
$F \rightarrow i$

| parse tree **[1✓]** | parse tree **[1✓]** | parse tree **[1✓]** |
|---|---|---|
|  |  |  |
| a * b + c * d | ( a + b ) * c | a + b + c |
| abstract syntax tree **[1✓]** | abstract syntax tree **[1✓]** | abstract syntax tree **[1✓]** |
|  |  |  |

3.

4. Using Thompson's construction ***exactly***, draw nondeterministic finite αὐτόματα for each of the following regular expressions. **[3✓]**

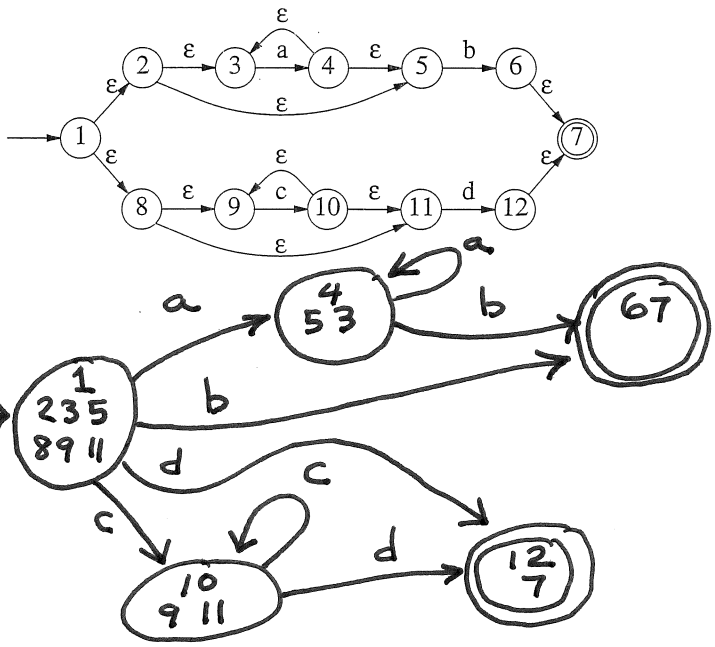
| abc | a\|bc | ab* |
|---|---|---|
|  |  |  |

5. Given the nondeterministic finite αὐτόματον shown here:
   (a) Write the regular expression that was used by Thompson's construction to create this NFA. **[1✓]**

   a*b|c*d

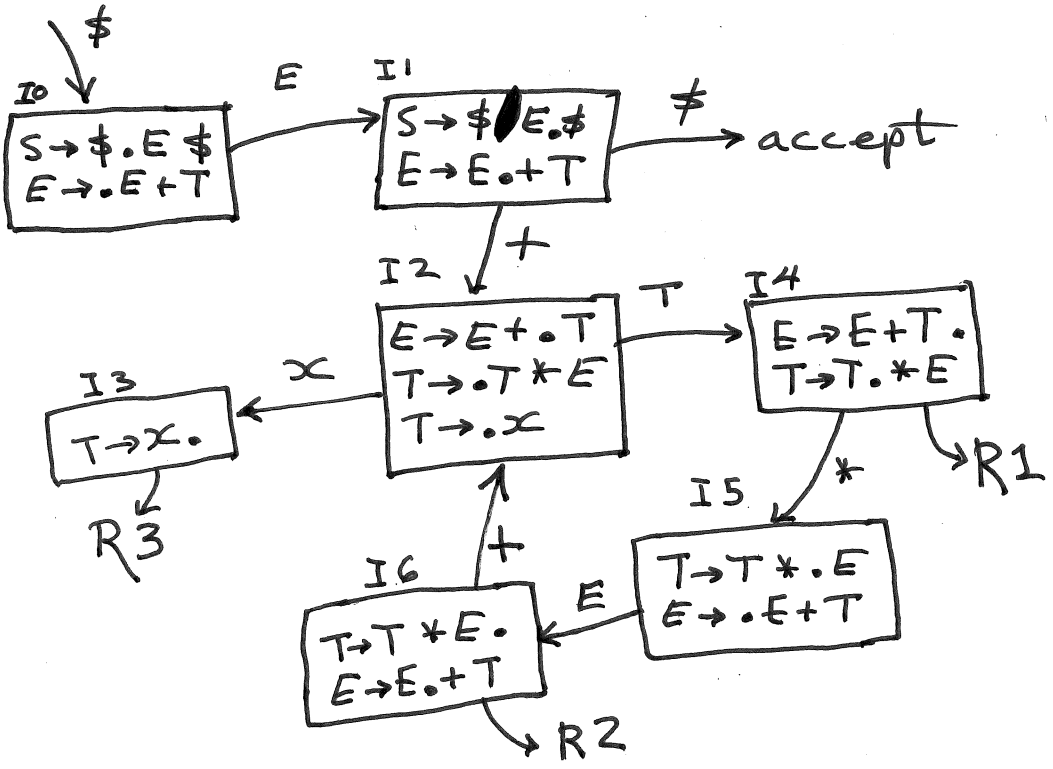
   (b) Fill in the table of ε-closures for each state. **[2✓]**
   (c) Use the subset algorithm to construct the equivalent ***deterministic*** finite αὐτόματον. Inside each state of the DFA, write the numbers of the NFA states to which it corresponds. Draw the DFA in the space under the NFA. **[2✓]**

| state *s* | ε-closure (*s*) |
|---|---|
| 1 | 1 2 3 5 8 9 11 |
| 2 | 2 3 5 |
| 3 | 3 |
| 4 | 4 5 3 |
| 5 | 5 |
| 6 | 6 7 |
| 7 | 7 |
| 8 | 8 9 11 |
| 9 | 9 |
| 10 | 10 9 11 |
| 11 | 11 |
| 12 | 12 7 |



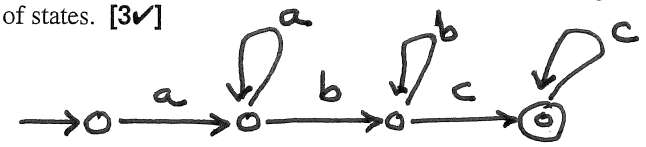

6. Given the grammar presented here, and using the style from the LALR(1) handout: Construct the characteristic finite state machine (CFSM), sets of items and transition diagram, showing shifts, reductions, and acceptance. **[5✓]**

| | |
|---|---|
| 0. | S → $ E $ |
| 1. | E → E + T |
| 2. | T → T * E |
| 3. | T → x |

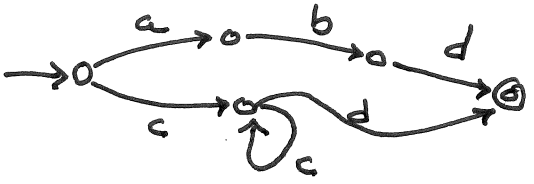


7. Draw deterministic finite αὐτόματα for each of the following `flex` regular expressions. Use the minimum possible number of states. **[3✓]**

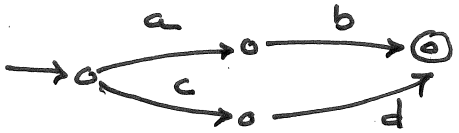
(i) `a+b+c+`




(ii) `(ab|c+)d`




(iii) `ab|cd`




8. Write `flex` regular expressions for each of the following, as they might appear in the second section of a lexical specification. Do not assume any macros. **[3✓]**

a. An identifier which consists of one or more (upper- or lower-case) letters, digits, and underscores. It must begin with a letter. Underscores may not appear as the first or last character, nor immediately next to each other.

`[a-zA-Z](_?[a-zA-Z0-9])*`

b. A hexadecimal integer constant consistent with C, C++, and Java syntax.

`0[xX][0-9A-Fa-f]+`

c. A string which consists of zero or more characters and which begins and ends with either a single quote (') or a double quote ("), provided that it begins and ends with the same kind of quote. Any character may appear within the string, except for the newline character and the quote which bounds the string.

`\"[^"\n]*\"|'[^'\n]*'`

9. Write an unambiguous `bison` grammar for the following language. Show necessary declarations in the first and second parts of the grammar. Do *not* show any semantic actions. **[4✓]**

(a) A program is a sequence of zero or more items.
(b) An item is a number or an identifier or a string or a list.
(c) A list consists of a left parenthesis followed by zero or more items followed by a right parenthesis.

```
%token NUM ID STR
%start prog
%%
prog : zitems
     ;
zitems : zitems item
       |
       ;
item : NUM | ID | STR | list ;
list : '(' zitems ')'
     ;
```

Multiple choice. To the ***left*** of each question, write the letter that indicates your answer. Write *Z* if you don't want to risk a wrong answer. Wrong answers are worth negative points. **[12✔]**

| | | | |
|---|---|---|---|
| number of correct answers | | × 1 = | $= a$ |
| number of wrong answers | | × ½ = | $= b$ |
| number of missing answers | | × 0 = | 0 |
| column total $c = \max(a - b, 0)$ | 12 | | $= c$ |

1. What does the regular expression . (dot) match ?
   (A) `[/n]`
   (B) `[\n]`
   (C) `[^/n]`
   (D) `[^\n]`

D

2. What kind of grammar does `bison` recognize ?
   (A) LALR(1)
   (B) LL(1)
   (C) context sensitive
   (D) regular

A

3. What is the external variable that points at the lexical information just matched by the scanner ?
   (A) `yyin`
   (B) `yylex`
   (C) `yylval`
   (D) `yytext`

D

4. The subset construction :
   (A) converts a DFA into an CFSM.
   (B) converts a regular expression into an NFA.
   (C) converts an NFA into a DFA.
   (D) minimizes a finite automaton.

C

5. For a grammar $G = \langle V_N, V_T, P, S \rangle$, $P$ consists of elements of the form $(A \rightarrow \beta)$, where :
   (A) $A \in V_N$ and $\beta \in (V_N \cup V_T)^*$
   (B) $A \in V_N$ and $\beta \in (V_N \cap V_T)^*$
   (C) $A \in V_T$ and $\beta \in (V_N \cup V_T)^+$
   (D) $A \in V_T$ and $\beta \in V_T^*$

A

6. What kind of grammar (in the Chomsky hierarchy) does `bison` recognize ?
   (A) unrestricted
   (B) context-sensitive
   (C) context-free
   (D) regular

C

7. Which semantic action in a `flex` grammar is clearly wrong ?
   (A) `return "*";`
   (B) `return '*';`
   (C) `return 0x2A;`
   (D) `return STAR;`

A

8. The `flex` expression `a|bc*` means :
   (A) `(a|(bc))*`
   (B) `(a|b)(c*)`
   (C) `a|((bc)*)`
   (D) `a|(b(c*))`

D

9. If an NFA with $n$ states is converted into a DFA, then the DFA is used to scan a string consisting of $k$ characters, how long will it take to scan the string ?
   (A) $O(2^n \times k)$
   (B) $O(k)$
   (C) $O(n^2)$
   (D) $O(n \times k)$

B

10. Given the grammar
    A → x A
    A → y
    what is an example of input that it matches ?
    (A) xxxxxxxxy
    (B) xyyyyyyyy
    (C) yxxxxxxxx
    (D) yyyyyyyyx

A


11. Given the grammar
    E → E + E
    E → E * E
    E → ( E )
    E → i
    we classify it as :
    (A) LL(1)
    (B) LR(1)
    (C) ambiguous
    (D) regular

C

12. The variable `yylval` points at :
    (A) a single node for the AST constructed by the scanner.
    (B) a string in the string set.
    (C) the input file scanned by the lexer.
    (D) the root of the AST constructed by the parser.

A