

CS 410 F05 Midterm Solution [Bonus]

Problem 1 [18 pts.]

Short answer problems. Point values shown in brackets.

- A. [2] *bison* is a tool for automatically building a parser from a grammar.
- B. [2] A DFA is a machine to recognize strings in a regular set.
- C. [2] A parser is a machine to recognize strings in a context free language.
- D. [1] A regular expression denotes a regular set.
- E. [1] A context free grammar denotes a context free language.
- F. [2] Among other operations, symbol tables generally need an operation to check if a variable is defined in the innermost scope (i. e., one that would not check other outer scopes). Why?
check if a variable is multiply-defined in the scope

(N = nondeterministic; D = deterministic; FA = finite automaton; PDA = pushdown automaton)

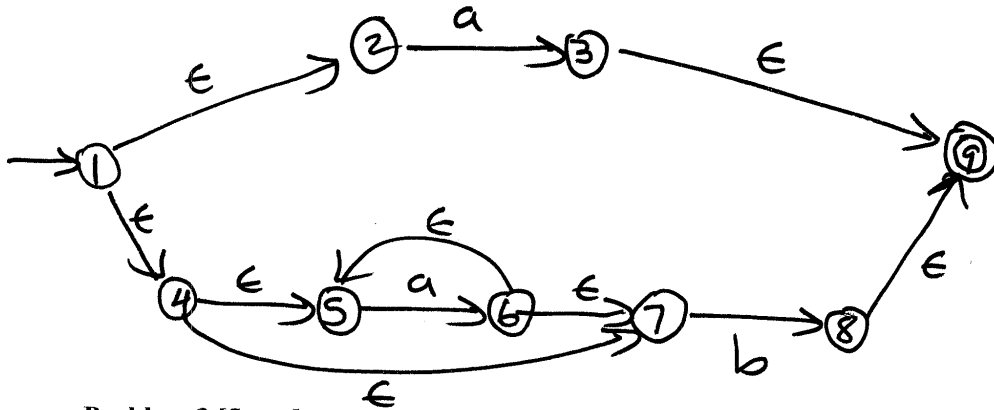
- G. [1] Is an NFA a more powerful machine than a DFA? no
- H. [1] Is an PDA (a.k.a., NPDA) a more powerful machine than a DPDA? yes
- I. [1] Are there things that can be recognized with a DFA that can't be recognized with a PDA? no

- J. [5] In a language such as C++, where identifiers have to be declared before they are used, how many passes are necessary to do semantic analysis and why?
one. At declaration sites add variables to symbol table. later at use sites, lookup vars in symbol table (to see if defined and to get type)

Problem 2 [5 pts.]

Using Thompson's Construction, build an NFA equivalent to the regular expression given below:

$a \mid a^*b$



Problem 3 [5 pts.]

Given the alphabet $\{a, b\}$ write a regular expression for the set of all strings such that every a in the string is immediately followed by at least two b 's.

$(abb(b)^*)^*$

OR

Alt answer: $b^*(abbb^*)^*$

many other correct answers.

Problem 4 [10 pts.]

Use subset construction to build a DFA equivalent to the NFA below. Show your work. Note: ϵ is the epsilon symbol.

	a	b	ϵ
1	{ 1 }	{ 3 }	{ }
<u>2</u>	{ }	{ }	{ 4 }
3	{ 2 }	{ 4 }	{ 2 }
4	{ 4 }	{ 2, 4 }	{ }

	a	b
[1]	[1]	[234]
[234]	[24]	[24]
[24]	[4]	[24]
[4]	[4]	[24]

Common mistake:
not getting that
 ϵ -closure(3) = {2, 3, 4}

Problem 5 [14 pts.]

Part A [10]. Give the FIRST and FOLLOW sets for each of the non-terminals in the following augmented grammar:

$S' \rightarrow S$
 $S \rightarrow S B x$
 $| y$
 $B \rightarrow m B r$
 $| r$

$$FIR(B) = \{m, r\}$$

$$FIR(S) = \{y\}$$

$$FIR(S') = FIR(S) = \{y\}$$

$$FOL(S') = \{\$ \}$$

$$FOL(S) = FOL(S') \cup FIR(B) = \{\$, m, r\}$$

$$FOL(B) = \{x, r\}$$

Part B [2]. Consider the DFA to recognize viable prefixes for an SLR parser for this grammar. Give an example *item* from the grammar above that will be part of a state that will have reduce entries. Note: Do not construct the whole DFA or even a whole set of items to answer this problem.

$$[S \rightarrow S B x \cdot]$$

Part C [2]. Using your answer from part B, name which set you computed in part A would be used to figure out the reduce entries for the state with the item from part B.

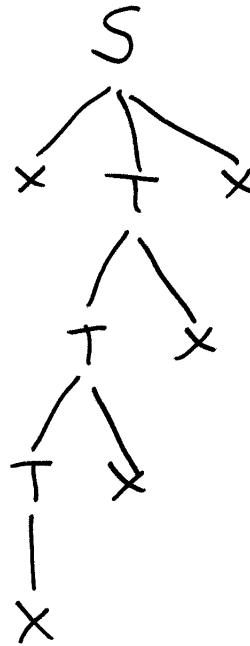
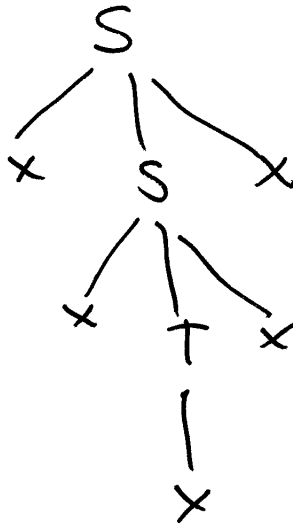
$$FOLLOW(S)$$

Problem 6 [5 pts.]

Show that the following grammar is ambiguous.

$$\begin{aligned} S &\rightarrow x S x \\ &\quad | x T x \end{aligned}$$

$$\begin{aligned} T &\rightarrow T x \\ &\quad | x \end{aligned}$$



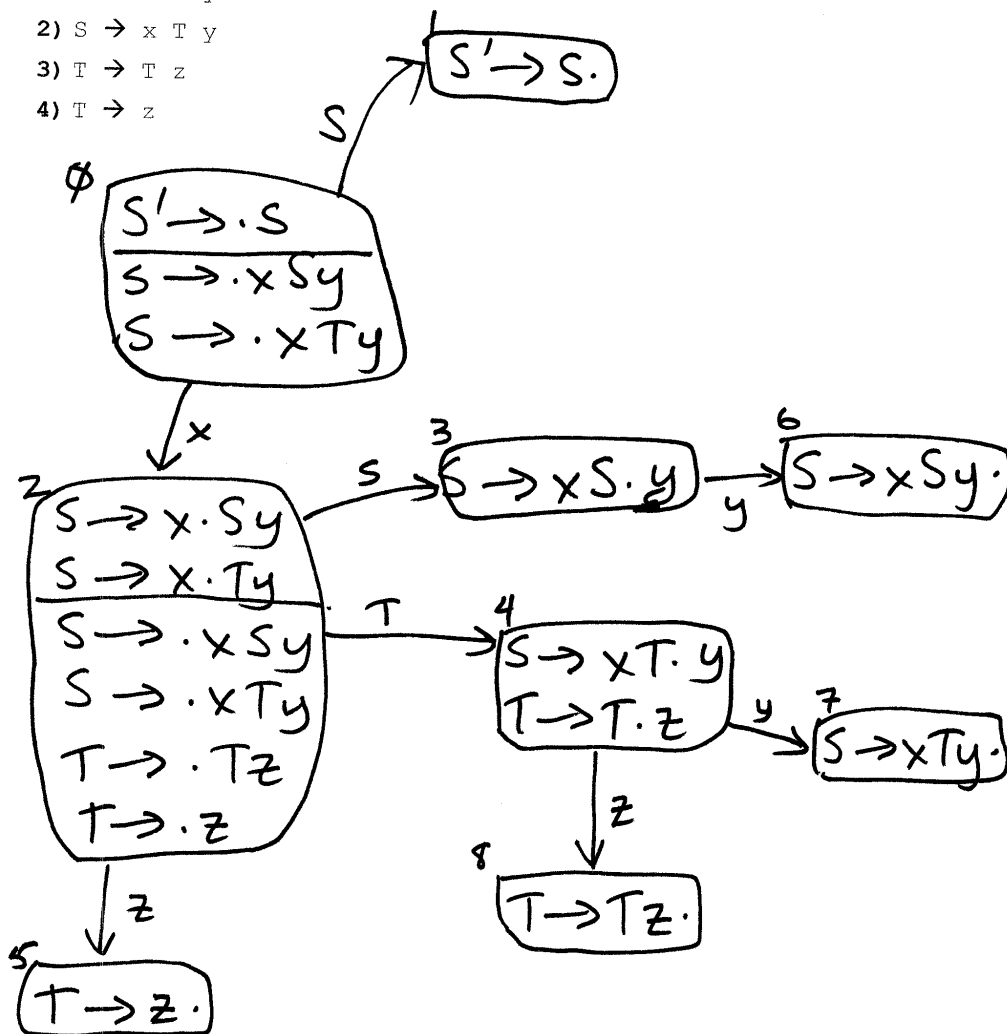
Two derivations
for "xxxxxx"

Many other correct answers.

Problem 7 [13 pts.]

Build the DFA for recognizing viable prefixes for an SLR parser for the grammar below. Show the set of items for each state. Note: the grammar has already been augmented for you.

- 0) $S' \rightarrow S$
- 1) $S \rightarrow x S y$
- 2) $S \rightarrow x T y$
- 3) $T \rightarrow T z$
- 4) $T \rightarrow z$



Problem 8 [10 pts.]

Write semantic rules for the grammar below to compute the boolean semantic attribute $S.moreb$, which is true iff the sentence parsed has more b's in it than a's. Note: the subscripts below are only to distinguish multiple instances of a non-terminal in a production.

Hint: you can use additional semantic attributes.

You may not use any global variables in your answer.

idea: count a's + b's + compare

$S \rightarrow A$ $S.moreb = A.numb > A.numa;$

$A \rightarrow a B d$ $A.numa = 1;$
 $A.numb = B.numb;$

$A \rightarrow a A_1 d$ $A.numa = 1 + A_1.numa;$
 $A.numb = A_1.numb;$

$B \rightarrow b c$ $B.numb = 1;$

$B \rightarrow b B_1 c$ $B.numb = B_1.numb + 1;$

Alt solution (idea: count b's, subtract a's from this to get the difference)

$S \rightarrow A$ $S.moreb = A.diff > 0;$
 $A \rightarrow a B d$ $A.diff = B.num - 1;$
 $A \rightarrow a A_1 d$ $A.diff = A_1.diff - 1;$
 $B \rightarrow b c$ $B.num = 1;$
 $B \rightarrow b B_1 c$ $B.num = B_1.num + 1;$