EC3204: Programming Languages and Compilers (Fall 2024)
Mid-term Exam

Open-book Test

Date and Time: 10/28, 16:00 – 17:15
Place: EECS Building B201

Student ID: ______________________________

Name: ___________________________________

- You cannot use electronic devices during the exam.
- No partial points will be given for any subproblem.
- Leave the score table blank.

	Score Range	Your Score
Problem 1	0 – 20	
Problem 2	0 – 15	
Problem 3	0 – 10	
Problem 4	0 – 15	
Problem 5	0 – 20	
Problem 6	-20 – 20	
Total	0 – 100	

Problem 1. (20pt) String Recognition

Recall that, both NFA and DFA recognize a string w iff there is a path from the start state to one of the final states covered by w. Consider the below algorithms for recognizing strings using NFA and DFA, which may contain errors. Identify and fix the wrong parts in the algorithms, if any. For example, if X at line 3 is wrong and must be replaced with Y, write "Line 3: $X \rightarrow Y$". If you find no errors in the algorithm(s), write "OK" in the corresponding box(es). Suppose, given a string $w = xy_1 \cdots y_n\$$, $\mathsf{first}(w)$ returns x, and $\mathsf{rest}(w)$ returns $y_1 \cdots y_n\$$.

1. (10pt)

Algorithm 1 Simulation of NFA (potentially wrong)

Input: An NFA $(N, \Sigma, \delta_N, n_0, N_A)$, an input string w
Output: Accept or Reject

1: $S \leftarrow \{n_0\}$
2: $x, w \leftarrow \mathsf{first}(w), \mathsf{rest}(w)$
3: **while** $x \neq \$$ **do** ▷ \$: end marker of w
4: $\quad S \leftarrow \epsilon\text{-}\mathsf{closure}(\mathsf{next}(S, x))$ where $\mathsf{next}(S, x) = \delta_N(S, x)$
5: $\quad x, w \leftarrow \mathsf{first}(w), \mathsf{rest}(w)$
6: **if** $S = N_A$ **then return** Accept
7: **else return** Reject

Your answer:

2. (10pt)

Algorithm 2 Simulation of DFA (potentially wrong)

Input: A DFA $(D, \Sigma, \delta_D, d_0, D_A)$, an input string w
Output: Accept or Reject

1: $s \leftarrow d_0$
2: $x, w \leftarrow \mathsf{first}(w), \mathsf{rest}(w)$
3: **while** $x \neq \$$ **do** ▷ \$: end marker of w
4: $\quad s \leftarrow \mathsf{next}(s, x)$ where $\mathsf{next}(s, x) = \delta_D(s, x)$
5: $\quad x, w \leftarrow \mathsf{first}(w), \mathsf{rest}(w)$
6: **if** $\{s\} = D_A$ **then return** Accept
7: **else return** Reject

Your answer:

Problem 2. (15pt) NFA to DFA

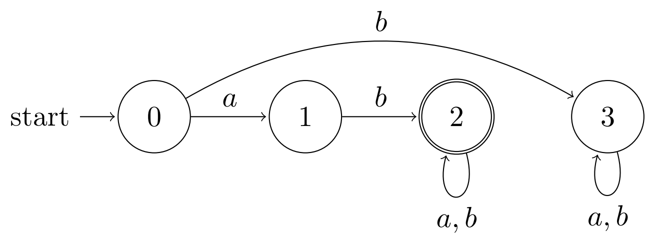
Consider the subset construction algorithm below that we learned in class.

Algorithm 3 Subset Construction

Input: An NFA $(N, \Sigma, \delta_N, n_0, N_A)$
Output: An equivalent DFA $(D, \Sigma, \delta_D, d_0, D_A)$

1: $d_0 \leftarrow \epsilon\text{-closure}(\{n_0\})$
2: $D \leftarrow \{d_0\}$ ▷ D: a set of DFA states
3: $W \leftarrow \{d_0\}$ ▷ W (workset): a set of DFA states to process
4: **while** $W \neq \emptyset$ **do**
5:     pick and remove q from W
6:     **for** $x \in \Sigma$ **do** ▷ consider each input symbol
7:         $t \leftarrow \epsilon\text{-closure}(\bigcup_{s \in q} \delta_N(s, x))$
8:         $D \leftarrow D \cup \{t\}$
9:         $\delta_D(q, x) \leftarrow t$
10:         **if** t has not been added to W before **then**
11:             $W \leftarrow W \cup \{t\}$
12: $D_A \leftarrow \{q \mid q \in D, q \cap N_A \neq \emptyset\}$
13: **return** $(D, \Sigma, \delta_D, d_0, D_A)$

1. (10pt) Given the below NFA as input, draw a DFA that will be produced by running Algorithm 3.



Your answer:

2. (5pt) Suppose we replaced the instruction at line 1 with $d_0 \leftarrow \{n_0\}$. Provide an NFA that will produce an incorrect output (i.e., semantically different DFA) when running the modified algorithm.

Your answer:

Problem 3. (10pt) Context-free Grammar

Consider the grammar below:

$$S \rightarrow aSbbS \mid bS \mid \epsilon$$

where S is a non-terminal symbol, and a, b are terminal symbols.

1. (3pt) Give a leftmost derivation for the string abb.
Your answer:

2. (3pt) Give a rightmost derivation for the string abb.
Your answer:

3. (4pt) Is the grammar ambiguous? Justify your answer.
Your answer:

Problem 4. (15pt) Top-Down Parsing

Consider the grammar where the start variable is A and the terminal symbols are $\{x, y, z, (,)\}$.

$$A \to AxB \mid B, \qquad B \to ByC \mid C, \qquad C \to z \mid (A)$$

1. (5pt) List the *First* and *Follow* sets for the grammar above.

$$\begin{aligned} First(A) &= \{ \qquad\qquad\qquad \}, & Follow(A) &= \{ \qquad\qquad\qquad \} \\ First(B) &= \{ \qquad\qquad\qquad \}, & Follow(B) &= \{ \qquad\qquad\qquad \} \\ First(C) &= \{ \qquad\qquad\qquad \}, & Follow(C) &= \{ \qquad\qquad\qquad \} \\ First(AxB) &= \{ \qquad\qquad\qquad \}, & First(ByC) &= \{ \qquad\qquad\qquad \} \end{aligned}$$

2. (5pt) Complete the LL(1) parsing table for the grammar.

	x	y	z	(	)	\$
A						
B						
C						

3. (5pt) Is the grammar in LL(1)? Justify your answer based on the parsing table you obtained. Your answer:

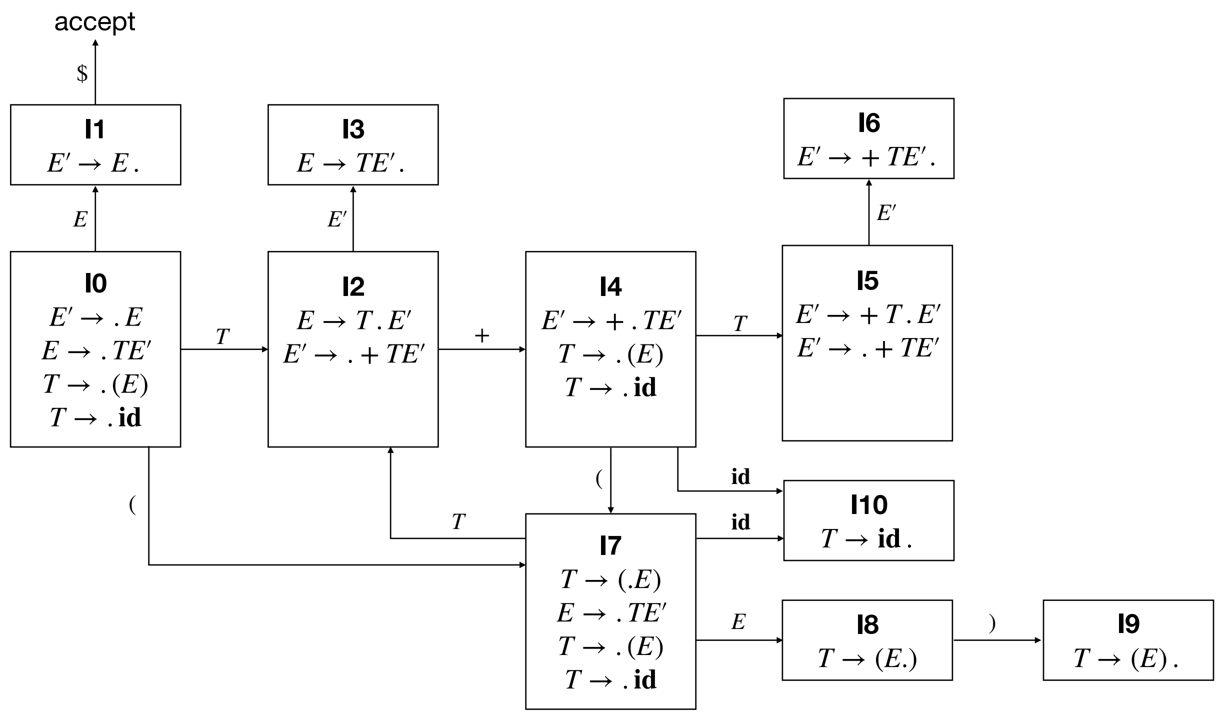
Problem 5. (20pt) Bottom-Up Parsing

Consider the expression grammar G:

$$\begin{array}{llllllllllll} (1) & E & \rightarrow & TE' & (2) & E' & \rightarrow & +TE' & (3) & E' & \rightarrow & \epsilon \\ (4) & T & \rightarrow & (E) & (5) & T & \rightarrow & \mathbf{id} & & & & \end{array}$$

where E is the start variable, and terminal symbols are $\{+, (,), \mathbf{id}\}$.

1. (10pt) Complete the below LR(0) automaton for the grammar G. Fix the wrong parts and add missing parts, if any.



2. (10pt) Construct an SLR parsing table for the grammar G. Extend the template if necessary.

State	**id**	+	*	(	)	$	E	E'	T	F
0										
1						acc				
2										
3										
4										
5										
6										
7										
8										
9										
10										

Problem 6. (20pt) O/X Questions

You will get 2 points for each correct answer. You will lose 2 points for each wrong answer.

(1) The string recognition of NFA terminates for every finite input string. (O, X)

(2) In Algorithm 3 from Problem 2, the condition ("t has not been added to W before") at line 10 is true iff $t \notin W$. (O, X)

(3) In Algorithm 3 from Problem 2, the condition ("t has not been added to W before") at line 10 is true iff $t \notin D$. (O, X)

(4) During the execution of the while-loop (lines 4–11) in Algorithm 3 from Problem 2, $W \subseteq D$ always. (O, X)

(5) There exists a regular expression that accepts the strings in $L = \{(1+2), ((1+2)+3), (1+(2+3)), (((1+2)+3)+4)\}$. (O, X)

(6) The language of a context-free grammar is the set of all sentential forms. (O, X)

(7) The language of a context-free grammar is the set of strings that can be parsed by some automatic algorithms. (O, X)

(8) The below grammar, where a and b are terminal symbols, is in SLR. (O, X)

$$S \rightarrow aSbS \mid bS \mid \epsilon$$

(9) In SLR parsing algorithm, the goto action always occurs after a reduce. (O, X)

(10) In SLR parsing algorithm, if a terminal symbol is on the top of the stack, we must take a reduce action. (O, X)