

CS143 Midterm

Spring 2022

- Please read all instructions (including these) carefully.
- There are 5 questions on the exam, some with multiple parts. You have 90 minutes to work on the exam.
- The exam is open note. You may use laptops, phones and e-readers to read electronic notes, but not for computation or access to the internet for any reason other than to access the class webpage.
- Please write your answers in the space provided on the exam, and clearly mark your solutions. Do not write on the back of exam pages or other pages.
- Solutions will be graded on correctness and clarity. Each problem has a relatively simple and straightforward solution. You may get as few as 0 points for a question if your solution is far more complicated than necessary. Partial solutions will be graded for partial credit.

NAME: _____

In accordance with both the letter and spirit of the Honor Code, I have neither given nor received assistance on this examination.

SIGNATURE: _____

Problem	Max points	Points
1	10	
2	25	
3	15	
4	25	
5	25	
TOTAL	100	

1. Regular Grammars

In class, we discussed how a CFG can be more expressive than a regular expression. However, a subset of CFGs we will call the *regular grammars* (RGs) have exactly the same expressive power as regular expressions.

A regular grammar is a CFG with any number of nonterminals in which every production follows one of three forms:

- $A \rightarrow \varepsilon$
- $A \rightarrow a$
- $A \rightarrow aB$

where a lowercase letter is a single terminal. Note that we allow the case where $B = A$.

For alphabet $\Sigma = \{a, b, c, d\}$, define an RG that is equivalent to the regular expression $(a|c)(dbb^*)^*$

Answer: (your answer may use at most 5 non-terminal symbols)

2. Context-Free Grammars

Given the following two CFGs over the alphabet $\Sigma = \{a, b, c\}$, what is the most restrictive language that can describe them among the following:

$$\text{LR}(0) \subset \text{SLR}(1) \subset \text{Unambiguous CFGs} \subset \text{All CFGs}.$$

For each CFG, explain why it cannot be expressed in the next more restrictive language. If the grammar is LR(0) then no explanation is needed.

(a) $A \rightarrow a \mid BaB \mid C$
 $B \rightarrow b \mid A$
 $C \rightarrow c \mid BcB$

Answer:

(b) $A \rightarrow aaC \mid C$
 $B \rightarrow bB \mid a$
 $C \rightarrow c \mid B$

Answer:

3. Syntax-Directed Translation

Consider a non-standard binary number system where the value of each binary number b is defined as the *alternating sum* of the decimal numbers that non-zero binary digits represent from right to left. For example, $val(\varepsilon) = 0$, $val(100) = 2^2 = 4$, $val(100010) = 2^1 - 2^5 = -30$, and $val(1100001) = 2^0 - 2^5 + 2^6 = 33$.

Given the following grammar for a non-standard binary numbers b , add semantic actions that computes $val(b)$. You must use the following attributes only: `int val` and `int tmp`. Use Bison syntax: `$i.val` refers to the `val` attribute of the i^{th} symbol of the production and `$$`.`val` refers to the `val` attribute of the production's result. You should not use any global variables or any attributes other than `val` and `tmp`.

$$\begin{array}{l} S \rightarrow T \\ \quad | \varepsilon \\ T \rightarrow 0T \\ \quad | 1T \\ \quad | 0 \\ \quad | 1 \end{array}$$

Answer:

```
S -> T {
```

```
}
```

```
S -> ε {
```

```
}
```

$T \rightarrow 0T \{$

$\}$

$T \rightarrow 1T \{$

$\}$

$T \rightarrow 0 \{$

$\}$

$T \rightarrow 1 \{$

$\}$

4. First and Follow Sets

We have lost our CFG, but luckily we have the First sets and all of the First/Follow relationships. Construct a grammar that is consistent with the following information:

Each nonterminal has exactly two productions.

$$\text{First}(A) = \{a, b\}$$

$$\text{First}(B) = \{a, b\}$$

$$\text{First}(D) = \{d, \varepsilon\}$$

$$\text{First}(B) - \{\varepsilon\} \subseteq \text{Follow}(a)$$

$$d \in \text{Follow}(B)$$

$$\text{Follow}(A) \subseteq \text{Follow}(d)$$

$$\text{First}(D) - \{\varepsilon\} \subseteq \text{Follow}(B)$$

$$\text{Follow}(A) \subseteq \text{Follow}(D)$$

$$\text{Follow}(A) \subseteq \text{Follow}(B)$$

$$\text{First}(D) - \{\varepsilon\} \subseteq \text{Follow}(b)$$

$$\text{Follow}(B) \subseteq \text{Follow}(D)$$

$$\text{Follow}(B) \subseteq \text{Follow}(b)$$

$$\text{Follow}(B) \subseteq \text{Follow}(a)$$

$$\text{Follow}(D) \subseteq \text{Follow}(d)$$

Answer:

5. Bottom-Up Parsing

Each of the following two subproblems describe a deterministic (i.e., DFA) LR(0) parsing automaton. Show your grammar and fill in the parsing automaton with transitions and each state labeled with its set of LR(0) items. You do *not* need to analyze the automaton to determine whether the grammar is LR(0) or SLR(1). The grammar and the automaton constitute a complete answer to each subproblem.

Assume that the first step of the automaton construction is to add a new production $S' \rightarrow S$ to the grammar, as described in class. This production should be included in your grammars, in your automata, and in your counts.

Give the simplest possible grammar (fewest productions *and* fewest terminals) that result in a parsing automaton satisfying the description.

- (a) An automaton (and corresponding CFG) with two states and one transition from the start state to the second state.

Answer:

- (b) An automaton (and corresponding CFG) with a minimal number of states, without loops, where one state has two incoming transitions.

Answer: