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: __________________________________________________

<table>
<thead>
<tr>
<th>Problem</th>
<th>Max points</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>TOTAL</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>
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)

\[
\begin{align*}
S & \rightarrow aA \\
& \quad | \quad cA \\
A & \rightarrow \varepsilon \\
& \quad | \quad dB \\
B & \rightarrow bA \\
& \quad | \quad bB
\end{align*}
\]
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) \[A \to a \mid BaB \mid C\]
\[B \to b \mid A\]
\[C \to c \mid BcB\]

**Answer:**
All CFGs. The grammar cannot be described by an unambiguous CFG, because the grammar is ambiguous. Since \(C\) can be replaced by \(BcB\), then \(A\) can go to either \(BaB\) or \(BcB\). Given a string “\(BaBcB\)”, there are two possible trees: one ‘\(a\)’ lower in the tree and one with ‘\(c\)’ lower in the tree.

(b) \[A \to aaC \mid C\]
\[B \to bB \mid a\]
\[C \to c \mid B\]

**Answer:**
The CFG is SLR(1). It is not in LR(0) because there is a shift-reduce conflict, which can be resolved in SLR(1) since \(\text{Follow}(B) = \{\$\}\) does not contain ‘\(a\)’. 

![Diagram of context-free grammar](image-url)
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, \( \text{val}(\varepsilon) = 0 \), \( \text{val}(100) = 2^2 = 4 \), \( \text{val}(100010) = 2^1 - 2^5 = -30 \), and \( \text{val}(1100001) = 2^0 - 2^5 + 2^6 = 33 \).

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

\[
S \rightarrow T \\
\quad\mid \varepsilon \\
T \rightarrow 0T \\
\quad\mid 1T \\
\quad\mid 0 \\
\quad\mid 1
\]

Answer:

\[
S \rightarrow T \$
\]

\[
S \rightarrow \varepsilon \$
\]

\[
\\$.val = 0;\\$
\]
$$.tmp = 2.\tmp * 2;
$$.val = 2.\val;

}

T \rightarrow 1T \{

$$.tmp = 2.\tmp * (-2);
$$.val = 2.\val + $$.tmp;

}

T \rightarrow 0 \{

$$.tmp = -1;
$$.val = 0;

}

T \rightarrow 1 \{

$$.tmp = 1;
$$.val = 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.

First(A) = \{a, b\}
First(B) = \{a, b\}
First(D) = \{d, \varepsilon\}

First(B) − \{\varepsilon\} ⊆ Follow(a)
   d ∈ Follow(B)
First(D) − \{\varepsilon\} ⊆ Follow(b)
   Follow(B) ⊆ Follow(D)
Follow(A) ⊆ Follow(D)
First(D) − \{\varepsilon\} ⊆ Follow(B)
   Follow(B) ⊆ Follow(B)
Follow(A) ⊆ Follow(B)
First(D) − \{\varepsilon\} ⊆ Follow(b)
   Follow(B) ⊆ Follow(a)
Follow(B) ⊆ Follow(B)
Follow(D) ⊆ Follow(d)

Answer:

\[
\begin{align*}
A & \rightarrow aBd \mid BD \\
B & \rightarrow bD \mid a \\
D & \rightarrow d \mid \varepsilon
\end{align*}
\]

Note: some variations here are possible. E.g., instead of \(A \rightarrow aBd\), you could have \(A \rightarrow aB\).
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 automatons, 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:

$$A \rightarrow \varepsilon$$

\[
\begin{array}{c}
S' \rightarrow .A \\
A \rightarrow . \\
A \rightarrow . \\
S' \rightarrow A. \\
\end{array}
\]
(b) An automaton (and corresponding CFG) with a minimal number of states, without loops, where one state has two incoming transitions.

Answer:

\[
A \rightarrow BB \\
B \rightarrow b
\]

(Note: We accepted any solution with at most 7 states as close enough to minimal.)