CS 3350 Automata, Computability, and Formal Languages
Fall 2019, Test 2

Last 4 digits of your UTEP ID number: 

General comments:

- you are allowed up to 5 pages of handwritten notes;
- if you need extra pages, place the last 4 digits of ID number on each extra page;
- the main goal of most questions is to show that you know the corresponding algorithms; so, if you are running out of time, just follow the few first steps of the corresponding algorithm;
- each question will be graded on its own merit; so, for example, if when answering to the first part of the question, you got a wrong automaton, but on the second part, you correctly traced the new automaton, you will get full credit for the second part.

Good luck!

1. Use a general algorithm to construct a (non-deterministic) pushdown automaton that corresponds to the following context-free grammar with the starting variable $S$: $S \rightarrow Bcd$, $B \rightarrow Sdc$; $S \rightarrow \varepsilon$. Show, step by step, how the word $dced$ will be accepted by this automaton. Its derivation in the given grammar is straightforward: $S \rightarrow Bcd \rightarrow Sdcd \rightarrow dcd$. 

file:///Q:/cs3350.19a/test2.html
1) $S \rightarrow Bcd$
   $B \rightarrow Sdc$
   $S \rightarrow \varepsilon$

![Diagram]

- Transition labels: $1 \rightarrow \varepsilon, \varepsilon \rightarrow S$
- Input sequence: $dcccddddd$
2. Design a finite automaton for recognizing binary sequences that have odd number of a's or odd number of b's. Assume that the input strings contain only symbols a and b. The easiest is to have 4 states:

- a state ee in which we have even number of a's and even number of b's,
- a final state eo, in which we have even number of a's and odd number of b's, and
- similarly defined states oe (odd-even, final) and oo (odd-odd, not final).

You just need to describe transitions between these states, and which states are final. Show, step-by-step, how your automaton will accept the string abaa. Show how the general algorithm will produce a context-free grammar that generates all the words accepted by this automaton -- and only words generated by this automaton. On the example of a word abaa accepted by this automaton, show how the tracing of acceptance of this word by the finite automaton can be translated into a generation of this same word by your context-free grammar.
2)

\[ \begin{array}{c}
\text{States} \\
\text{ee} \rightarrow \text{EE} \\
\text{eo} \rightarrow \text{EO} \\
\text{oe} \rightarrow \text{OE} \\
\text{oo} \rightarrow \text{OO}
\end{array} \]

\[ \begin{array}{c}
\text{CFG} \\
\text{EE} \rightarrow \text{EO} \\
\text{EO} \rightarrow \text{EE} \\
\text{EO} \rightarrow \text{EO} \\
\text{OE} \rightarrow \varepsilon
\end{array} \]

\[ \text{EE} \rightarrow \text{aOE} \rightarrow \text{abOO} \rightarrow \text{abaEO} \rightarrow \text{abaaOO} \rightarrow \text{abaa} \]

abaa?
3. Use the general algorithm to transform the grammar from Problem 1 into Chomsky normal form.

\[
\begin{align*}
S & \rightarrow Bcd \\
B & \rightarrow VdVc \\
S & \rightarrow E
\end{align*}
\]

**Step 0**

\[
\begin{align*}
S_0 & \rightarrow E \\
B & \rightarrow VdVc
\end{align*}
\]

**Step 1**

\[
\begin{align*}
S_0 & \rightarrow Bcd
\end{align*}
\]

**Step 2**

\[
\begin{align*}
Vc & \rightarrow c \\
Vd & \rightarrow d \\
B & \rightarrow SVdVc \\
S & \rightarrow NVdVc \\
B & \rightarrow VdVc
\end{align*}
\]

**Step 3**

\[
\begin{align*}
Vc & \rightarrow BVc \\
Vd & \rightarrow VdVc \\
S_0 & \rightarrow VdVc \\
N & \rightarrow SVdVc \\
B & \rightarrow VdVc
\end{align*}
\]
4. Use the general algorithm to transform the following pushdown automaton into a context-free grammar. This automaton has 4 states:

- the starting state s,
- the reading state r,
- the checking state c, and
- the final state f.

The transitions are as follows:

- From s to r, the transition is:
  - $\varepsilon, \varepsilon \rightarrow \$;
- From r to r, the transitions are:
  - a, $\varepsilon \rightarrow b$;
  - b, $\varepsilon \rightarrow a$;
- From r to c, we have a jump $\varepsilon, \varepsilon \rightarrow \varepsilon$.
- From c to c, the transitions are:
  - a, a $\rightarrow \varepsilon$
  - b, b $\rightarrow \varepsilon$
- From c to f, the only transition is:
  - $\varepsilon, \$ $\rightarrow \varepsilon$.

Show, step-by-step, how the resulting grammar will generate the sequence $abab$. 

$$
\begin{align*}
\varepsilon & \rightarrow \$ \\
A_0 \rightarrow \varepsilon A_0 c \varepsilon \\
a_0, \varepsilon \rightarrow b \\
A_0 \rightarrow a A_0 c b \\
A_0 \rightarrow b A_0 c a \\
A_0 \rightarrow c A_0 c \varepsilon \\
A_0 \rightarrow \varepsilon \\
A_0 \rightarrow \varepsilon & 
\end{align*}
$$
CFG

\[ S \rightarrow \text{start word.} \]

\[ A \rightarrow S \rightarrow Arc \]
\[ Arc \rightarrow a Arc b \]
\[ Arc \rightarrow b Arc a \]
\[ Arc \rightarrow 1Acc \]
\[ Acc \rightarrow \epsilon \]
\[ S \rightarrow AsF \]

Diagram:

- \( S \)
  - \( AsF \)
  - \( Arc \)
    - \( a \)
    - \( b \)
  - \( Arc \)
    - \( b \)
    - \( a \)
  - \( Acc \)
    - \( \epsilon \)

Input: \( 1abab \)
5. Show, step by step, how the stack-based algorithm will transform the expression $3 - (5 - 7)$ into a postfix expression, and then how a second stack-based algorithm will compute the value of this postfix expression.
6. For the grammar from Problem 1, show how the word $dcd$ can be represented as $uvxyz$ in accordance with the pumping lemma for context-free grammars. Show that the corresponding word $uvxuyz$ will be generated by this grammar.