CS 3350 Automata, Computability, and Formal Languages  
Fall 2018, Test 1

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 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!

1-4. In class, we studied an automaton for recognizing valid Java identifiers. This automaton has 3 states: start (s), identifier (i), and error (e). Start is the starting state, identifier is the only final state.

The transitions are as follows:

- from s, any letter (a, ..., z, A, ..., Z) leads to i, any other symbol leads to e;
- from i, any letter, any digit (0, ..., 9), or an underscore symbol _ lead back to i, while any other symbol leads to e;
- from e, every symbol leads to e.

1. Is any of the three states a sink state? Explain your answer.

2-3. Trace, step-by-step, how this finite automaton will check whether the following two words (sequences of symbols) represent a valid Java identifier:

- the word number2 (which this automaton should accept) and
- the word 2ndNumber (which this automaton should reject).

4. Write down the tuple \(<Q, \Sigma, \delta, q_0, F>\) corresponding to this automaton:

- \(Q\) is the set of all the states,
- \(\Sigma\) is the alphabet, i.e., the set of all the symbols that this automaton can encounter;
- \(\delta: Q \times \Sigma \rightarrow Q\) is the function that describes, for each state \(q\) and for each symbol \(s\), the state \(\delta(q, s)\) to which the automaton that was originally in the state \(q\) moves when it sees the symbol \(s\) (you do not need to describe all possible transitions this way, just describe two of them);
- \(q_0\) is the staring state, and
- \(F\) is the set of all final states.
Yes. The error (e) state is a sink state because any symbol would lead back to e, meaning that it is impossible to leave or move from this state.

-3. number 2

\[
| n | u | m | b | l | e | r | 2 |
\]

- 2nd Number

\[
| 2 | n | i | d | N | u | l | m | b | l | e | r |
\]

1. \(Q = \{\text{start, identifier, error}\}\)

\(\Sigma = \{a, \ldots, z, A, \ldots, Z, 0, \ldots, 9, \text{any other symbol}\}\)

\(\delta(\text{start}, a) = \text{identifier}\)

\(\delta(\text{start}, +) = \text{error}\)

\(q_0 = \text{start}\)

\(F = \{\text{identifier}\}\)
5. Draw an automaton for recognizing all possible binary signed integers. Trace this automaton on the example of numbers +10 (that it should accept), -10 (also accepted), and 10 (should be rejected).

- +10
  
  + + + i 0
  s+ s1 i (i)

- -10
  
  - - + i 0
  s+ s1 i (i)

- 10
  
  i i i 0
  s+ c c
6-8. Let A be an automaton described in Problem 1. Let B be the following automaton that accepts all the strings that contain only letters but not any other symbols. This automaton has two states: the start state which is also a final state, and the sink state. The transitions are as follows:

- from the start state, any letter leads back to the start state, every other symbol leads to sink;
- from the sink state, any symbol leads back to sink.

6. Use the algorithm that we had in class to describe the following two new automata:

- the automaton that recognizes the union A U B of the two corresponding languages, and
- the automaton that recognizes the intersection of the two languages A and B.

7-8. Test these two new automata step-by-step on the following words:

- test the union automaton on the example of the words Var (that it should accept) and 2words (that it should reject);
- test the intersection automaton on the example of the words Var (that it should accept) and Var2 (that it should reject).
7 - 8.

- Var

\[
\begin{array}{ccccc}
V & a & r & l \\
\text{s} & \text{st} & \text{st} & \text{st} & \text{st}
\end{array}
\]

2 words

\[
\begin{array}{cccccccc}
2 & w & 0 & r & d & i & s \\
\text{s} & \text{st} & \text{c/sinh} & \text{c/sinh} & \text{c/sinh} & \text{c/sinh} & \text{c/sinh}
\end{array}
\]

- Var

\[
\begin{array}{ccccc}
V & a & r & l \\
\text{s} & \text{st} & \text{st} & \text{st} & \text{st}
\end{array}
\]

Var 2

\[
\begin{array}{ccccccc}
V & a & r & l & 2 \\
\text{s} & \text{st} & \text{st} & \text{st} & \text{st} & \text{st} & \text{c/sinh}
\end{array}
\]
9-10. Use the general algorithm that we learned in class to design a non-deterministic finite automaton that recognizes the language \((0 \cup 1)(1 \cup 2)\):

- first, describe the automata for recognizing 0, 1, and 2;
- then, combine them into the automata for recognizing the unions \(0 \cup 1\) and \(1 \cup 2\);
- finally, combine the two union automata into an automaton for recognizing the composition \((0 \cup 1)(1 \cup 2)\) of the two union languages.