IT 328, Introduction to the Theory of Computation

Programming Assignment 2+3 (Team work)

NFA $\to$ DFA $\to$ Minimized DFA

Due date: April 30, 2023, Sunday, 11:55 PM   100 points (90 on programs, 10 on report)

As the first programming assignment, form a team of 3 students and decide who will be the corresponding student for this programming assignment. If you are content with your previous team, you don't have to change your team, but A/B/C roles and the corresponding student don't have to be the same. However, you still have to use the ReggieNet to join the team for my records (there is a new *join-able set* for this assignment). The major programming responsibility is divided as follows.

- **Student A** should write a method that takes an NFA from a text file and convert it to an equivalent DFA. The DFA should be shown on the screen and written to a text file for Student B and Student C to use.
- **Student B** should write a method that takes a DFA from a text file created by Student A and minimize it to an optimal DFA. The optimal DFA should be shown on the screen and written to a text file for Student C to use.
- **Student C** should write a method that takes a DFA from students A and B and use it to parse strings attached to the original NFA file (alternatively, strings in a DFA file if conversion fails).

- The task for Student C seems more straightforward. The team may, but doesn't have to, ask Student C to be in charge of facilitating supporting tools, such as to design a Set class that can perform basic set operations like union, intersection, membership and equality tests, and add/remove elements.

NFA file format: $(Q, \Sigma, \delta, q_0, A)$ Consider file `X.nfa` as an example, we have

```
|Q|:  5
Sigma:  a b c d
----------------------------------------
0:  {1} {2} {} {} {0,1}
1:  {1} {1,2,4} {1} {1} {1}
2:  {2,4} {0} {1,3} {2} {2}
3:  {0} {1,3} {3} {2,4} {3}
4:  {3} {} {1} {3,4} {0,2,4}
----------------------------------------
Initial State:  3
Accepting Sate(s):  3,4

-- Input strings for testing -----------

aabacadbdcacbacadbacaabacdcbbcadbcadbacabacabacdacabca
....
```

- The first and second lines indicate the number of states and Σ. In this case, `X.nfa` has 5 states, indexed from 0 to 4, and $\Sigma = \{a, b, c, d\}$.

- The transition function, $\delta : Q \times \Sigma \bigcup \{\lambda\} \to 2^Q$, is presented in a transition table between two dash lines, where the number followed by a colon is the index of the state, then each set is the set of next states on each input alphabet corresponding to the alphabets in the same order shown in the `Sigma` line. The extra set at the end of each line is the λ-transition of the state. For example, consider q_4 of `X.nfa` above. We have: $\delta(q_4, a) = \{q_3\}, \delta(q_4, b) = \{\}, \delta(q_4, c) = \{q_1\}$, $\delta(q_4, d) = \{q_3, q_4\}$, and $\delta(q_4, \lambda) = \{q_0, q_2, q_4\}$. Note that, the last set is referred to the λ-transition, and it always contains the state itself, i.e., q_0 is included in $\delta(q_0, \lambda)$; likewise $q_1 \in \delta(q_1, \lambda)$ and $q_2 \in \delta(q_2, \lambda)$, and so on.
- After the transition table, the initial sates and accepting states are specified . In this example, q_3 is the initial state and the set of accepting states A is $\{q_3, q_4\}$.
- The rest of the file contains strings as the input strings for testing (after convert to a DFA and minimization. Note that: There are 30 strings and the first string is always the empty string.

DFA file format: $(Q, \Sigma, \delta, q_0, A)$ Consider `X.dfa` s an example, we have

```
  |Q|:    6
 Sigma:    a     b     c     d
 ------------------------------
     0:    1     2     0     3
     1:    4     3     4     4
     2:    1     5     2     3
     3:    5     3     2     5
     4:    4     3     4     4
     5:    5     5     2     5
 ------------------------------
Initial State:  0
Accepting Sate(s):  0,2,3,5

-- Input strings for testing -----------

aabacadbdcacbacadbacaabacdcbbcadbcadbacabacabacdacabca
....
```

- The format and presentation are similar to DFA file except that the type of the transition function becomes $\delta : Q \times \Sigma \to Q$, that allow us to make a better alignment for the transition table. Also, there is no λ-transition in DFA.

What to do

1. Make a new directory `~/IT328/nfadfa` on your unix account, where ~ is your home directory. All files related to this assignment should be prepared in this directory.
2. From `/home/ad.ilstu.edu/cli2/Public/IT328/nfadfa`, copy every files into to your own `nfadfa` directory. There are some NFA and DFA files, where the file name's extension indicates it is an NFA or a DFA. Note that `X.dfa` is an equivalent DFA version of `X.nfa`. Every given NFA/DFA file contains 30 test strings. `A.nfa`'s DFA and minimized versions are given for you to compare.

3. Name your main program as `NFA2DFA.java` that will take one argument from the command line. I will compile and run your program from the Unix command line as follows:

```
javac NFA2DFA.java
java NFA2DFA X.nfa
```

The argument is an NFA file as the aforementioned `X.nfa`. Your program should check every string attached to the NFA file (including an empty string) and print Yes or No on the screen for each string to indicate whether the NFA accepts the string. Your program will not parse the input string using the NFA directly, which is rather inefficient since we have to rely on back-tracking. Instead, the NFA should be converted to an equivalent DFA, and use it to parse every string attached to the NFA file before and after minimization.

The output of your program should follow the exact format shown in the sample output in the next page, including the following results on the screen:

(1) An equivalent DFA (before minimization) as shown in the sample output.

(2) A list of answers of the DFA on each of the 30 strings attached at the end of the NFA file. Arrange your program to show exactly 15 answers in a line on the screen.

(3) Minimize the DFA and show the optimal DFA using the same format on the screen.

(4) A list of answers of the optimal DFA on the same 30 strings. If your program is correct, the DFA and its minimized version should give the same results.

Incomplete Works: In case some of the jobs fail, you may get partial credit up to 70% depending on how much the team has accomplished. You have to skip the failing part and don't let your program terminate with an error exception. In general, there are 3 main tasks in this assignment. If one task fails, you may get up to 70% of the credit; if two fail, you may get up to 50% of the credit. In the following two cases, you have to make some extra adjustment for me to know.

(1) (Up to 70%) If Student B can successfully minimize a DFA to an optimal one but Student A fails to convert an NFA to a DFA for B, then don't submit `NFA2DFA.java`. In stead, submit a program named **`minimizeDFA.java`** that will read in a DFA file and minimize it.

```
javac minimizeDFA.java
java minimizeDFA X.dfa
```

The output should be the same except that the first line massage should be "`Minimize X.dfa`" and the testing strings should be the 30 string attached at the end of the DFA file.

(2) (Up to 50%) If both Student A and Student B fail, the team should submit a program named `parse.java` that will read in a DFA file and run it on the attached 30 strings.

```
javac parse.java
java parse X.dfa
```

The output should be the same up to the minimization section except that the first line massage should be "`Run X.dfa`" and the testing strings should be those attached at the end of `X.dfa`.

4. **Final Step on Unix Account** & **Report** are same as the previous assignment. Submit your report through corresponding student's ReggieNet and use bash script `submit328.sh` to prepare your program on your Unix account.

```
bash /home/ad.ilstu.edu/cli2/Public/IT328/submit328.sh peekapoo nfadfa
```

   Note that your programs have to be in the required directory ∼`/IT328/nfadfa/` as described in step 1 in order to let this script program correctly copy your programs into the secret directory, where your programs will be tested. Don't compile and run your programs from the directory where `submit328.sh` copies to, since it will mess up the permission. It is ok to modify your program after submission before the deadline, but you have to run `submit328.sh` again. Also, don't copy `submit328.sh` to your directory, since I may change it from time to time.

Sample output: (You **don't have to** print the program outputs on your report, they are lengthy; I will run your program and check.)

```
java NFA2DFA X.nfa strings.txt

NFA X.nfa to DFA X.dfa:

 Sigma:     a     b     c     d
 ------------------------------
     0:     1     2     0     3
     1:     4     3     4     4
     2:     1     5     2     3
     3:     5     3     2     5
     4:     4     3     4     4
     5:     5     5     2     5
 ------------------------------
0:  Initial State
0,2,3,5:  Accepting State(s)

Parsing results of strings attached in X.nfa:
 Yes  No   No   Yes  No   Yes  Yes  No   No   Yes  No   Yes  No   Yes  Yes
 No   Yes  Yes  Yes  No   No   Yes  Yes  No   No   Yes  Yes  No   No   Yes

Yes:16 No:14


Minimized DFA from X.dfa:

 Sigma:     a     b     c     d
 ------------------------------
     0:     1     2     0     3
     1:     1     3     1     1
     2:     1     3     2     3
     3:     3     3     2     3
 ------------------------------
 0:  Initial State
 0,2,3:  Accepting State(s)

Parsing results of strings attached in X.nfa:
 Yes  No   No   Yes  No   Yes  Yes  No   No   Yes  No   Yes  No   Yes  Yes
 No   Yes  Yes  Yes  No   No   Yes  Yes  No   No   Yes  Yes  No   No   Yes

Yes:16 No:14

|Q| 6 -> 4
```

The last line indicates the number of states reduced from 6 to 4.