CS 5000: F21: Theory of Computability
Assignment 3

Vladimir Kulyukin
Department of Computer Science
Utah State University

September 18, 2021

Learning Objectives

1. Closure Properties of Regular Languages
2. Subset Construction Algorithm
3. Myhill-Nerode Theorem (aka DFA Minimization Algorithm)

Introduction

In this assignment, we'll work some more with the closure properties of regular languages, implement the subset construction algorithm which is fundamental FA theory, and apply the algorithm of the Myhill-Nerode Theorem to minimize a DFA.

Problem 0 (0 points)

Review the slides for Lectures 4 and 5 or your class notes. I've also included additional PDF reading materials in the lecture modules in Canvas. You may want to read these if the slides and/or your class notes are not sufficient or if you didn't/couldn't attend the F2F lectures.

Problem 1 ($\frac{1}{2}$ point)

Let Σ be an alphabet. A finite language over Σ is a finite set of strings over Σ, i.e., a finite subset of Σ^*. Let L_1 be a regular language over Σ and L_2 be a finite language over Σ. Show that $L_1 \cap L_2$ is regular.

Problem 2 ($\frac{1}{2}$ point)

Let L_1 and L_2 be two regular languages over Σ. Show that $L_2 - L_1$ is regular, where $L_2 - L_1$ is the set-theoretic difference between the two languages.

Problem 3 (1 point)

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA, where

1. $Q = \{q_0, q_1, q_2, q_3, q_4\}$;
2. $\Sigma = \{0, 1\}$;
3. $\delta(q_0, 0) = q_1; \delta(q_0, 1) = q_3; \delta(q_1, 0) = q_2; \delta(q_1, 1) = q_4;\ \delta(q_2, 0) = q_1; \delta(q_2, 1) = q_4; \delta(q_3, 0) = q_2; \delta(q_3, 1) = q_4;\ \delta(q_4, 0) = q_4; \delta(q_4, 1) = q_4$.
4. q_0 is the start state;
5. $F = \{q_2, q_4\}$.

Minimize M with the algorithm of the Myhill-Nerode Theorem (Lecture 5) and draw your minimal DFA. Make sure that you clearly specify the start state and the end states. You don't have to write the table.

Problem 4 (2 points)

Now that we've had some experience with DFAs in Assignment 2 (recall `Mod3DFA.py` and `Mod5DFA.py`), let's put it to use and implement the subset construction algorithm. We'll work with the NFA from Lecture 4.[1] We used it in the subset construction example worked out on the board in class. We'll respresent the transition function of the NFA as a Python dictionary where keys are 2-tuples of states and alphabet symbols and the corresponding values are the sets of states. For example, in this NFA, we have $\delta(q_0, 1) = \{q_0, q_1\}$. In Python, we can implement it as follows.

```
NFA_DELTA_01 = {}
NFA_DELTA_01[('q0', '1')] = set(['q0', 'q1'])
```

The first line in the above code segment defines an empty dicitionary in `NFA_DELTA_01`. The second line specifies a transition from q_0 on 1 to $\{q_0, q_1\}$. Note that the dictionary key is defined as a tuple of two strings, i.e., `('q_0', 'q_1')`. This is an opportune moment to draw your attention to an important Python subtlety (in case you don't know it already): if we use a dictionary to represent transitions, we cannot use lists to represent sets of states, because lists are not hashable (i.e., cannot be keys in a dictionary).

In case you want, for consistency, to have the strings in your tuples lexicographically sorted, before you turn them into a tuple, you can use the `sorted()` function. Here's how.

```
>>> tuple(sorted(['q0', 'q1', 'q3', 'q4', 'q2']))
('q0', 'q1', 'q2', 'q3', 'q4')
```

Let's define the entire transition function for the NFA on slide 7 of Lecture 4.

[1] See Slide 7 in CS5000_F21_Lecture_04_SubsetConstructionWithQueueOptimization.pdf in Canvas.

```
NFA_DELTA_01 = {}
NFA_DELTA_01[('q0', '0')] = set(['q0'])
NFA_DELTA_01[('q0', '1')] = set(['q0', 'q1'])
NFA_DELTA_01[('q1', '0')] = set(['q2'])
NFA_DELTA_01[('q1', '1')] = set(['q2'])
```

Once we have the δ function defined, we can proceed to define the entire NFA as a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ as follows:

```
NFA_01 = (set(['q0','q1', 'q2']), ## this is Q - the set of states
          set(['0', '1']),        ## this is Sigma (i.e., the alphabet)
          NFA_DELTA_01,           ## this is the transition function
          'q0',                   ## this is the start state
          set(['q2'])             ## this is F - the set of final states
          )
```

The above NFA `NFA_01` is a 5-tuple of the set of states (i.e., `set(['q0','q1', 'q2'])`), the alphabet Σ (i.e., `set(['0', '1'])`), the transition function δ (i.e., `NFA_DELTA_01`), the start state (i.e., `q0`), and the set of final states (i.e., `set(['q2'])`).

To reiterate, the transition function δ is given in the dictionary `NFA_DELTA_01`. Each key is a 2-tuple that maps a state and symbol to the key's value, which is the set of states from where the NFA can go on the symbol from that state. In other words, `NFA_DELTA_01[('q0', '0')] = set(['q0'])` is equivalent to $\delta(q_0, 0) = \{q_0\}$. Alternatively, we could've represented each transition as a 3-tuple that consists of a start state, a symbol, and the set of states where the NFA can transition from the start state on the symbol. For example, `('q0', '1', set(['q0', 'q1']))` represents the same transition from q_0 to $\{q_0, q_1\}$ on 1. We could've filed away all these transition tuples into a list, but the list is not as efficient for lookup as a dictionary.

Implement the function `nfa_to_dfa(nfa)` that takes a 5-tuple NFA representation as defined above and produces a 5-tuple DFA representation by applying the subset construction algorithm. Implement the function `display_dfa(dfa)` that displays the resultant DFA in the format shown in the test run below. Save your solutions in `cs5000_f21_hw03.py` which has some starter code for you.

To help you with coding and debugging, I wrote two unit tests in `cs5000_f21_hw03_uts.py`. The first unit test converts to the DFA the NFA on slide 7 in the Lecture 4 PDF and displays it with `display_dfa()`. The second unit test does the same with the NFA in Problem 2 of Assignment 2. Here's what my output is when I run the unit tests.

```
>>> python cs5000_f21_hw03_uts.py

***** Assign 03: Subset Construction UT 01 ************
STATES: [('q0',), ('q0', 'q1'), ('q0', 'q1', 'q2'), ('q0', 'q2')]
SIGMA:  {'1', '0'}
START STATE: [('0', 'q')]
DELTA:
0) d(('q0',), 1) = ('q0', 'q1')
```

```
1) d(('q0',), 0) = ('q0',)
2) d(('q0', 'q1'), 1) = ('q0', 'q1', 'q2')
3) d(('q0', 'q1'), 0) = ('q0', 'q2')
4) d(('q0', 'q1', 'q2'), 1) = ('q0', 'q1', 'q2')
5) d(('q0', 'q1', 'q2'), 0) = ('q0', 'q2')
6) d(('q0', 'q2'), 1) = ('q0', 'q1')
7) d(('q0', 'q2'), 0) = ('q0',)
FINAL STATES: [('q0', 'q1', 'q2'), ('q0', 'q2')]
.
***** Assign 03: Subset Construction UT 02 ************
STATES: [('q0',), ('q1',), ('q0', 'q1'), ('q2',), ('q1', 'q2'), ('q0', 'q1', 'q2')]
SIGMA:  {'1', '0'}
START STATE: [('0', 'q')]
DELTA:
0) d(('q0',), 1) = ('q1',)
1) d(('q0',), 0) = ('q0', 'q1')
2) d(('q1',), 1) = ('q2',)
3) d(('q1',), 0) = ('q2',)
4) d(('q0', 'q1'), 1) = ('q1', 'q2')
5) d(('q0', 'q1'), 0) = ('q0', 'q1', 'q2')
6) d(('q2',), 1) = ('q2',)
7) d(('q2',), 0) = ('q2',)
8) d(('q1', 'q2'), 1) = ('q2',)
9) d(('q1', 'q2'), 0) = ('q2',)
10) d(('q0', 'q1', 'q2'), 1) = ('q1', 'q2')
11) d(('q0', 'q1', 'q2'), 0) = ('q0', 'q1', 'q2')
FINAL STATES: [('q1',), ('q0', 'q1'), ('q1', 'q2'), ('q0', 'q1', 'q2')]
.
----------------------------------------------------------------------
Ran 2 tests in 0.001s
OK
```

What to Submit

Save your solutions to Problems 1, 2, and 3 in `hw03.pdf`. I want to take this opportunity to thank all of you for the wonderful graphs that you drew in your homework 2 solutions by hand or with software tools. Save your solution to problem 4 as `cs5000_f21_hw03.py`. Zip your `hw03.pdf` and `cs5000_f21_hw03.py` in `hw03.zip` and upload it in Canvas.

Happy Thinking and Hacking!