Week 8: Nondeterministic Determination

Collaboration: You should work on the problems yourself, before discussing with others, including your cohorts at your cohort meeting. By the Assessed Cohort Meeting, you and all of your cohortmates, should be prepared to present and discuss solutions to all of the assigned problems. In addition to discussing with your cohortmates, you may discuss the problems with any other current CS3102 students you want, and use any resources you want except for any materials from previous offerings of this course or complete solutions that might be available on the web, which are not permitted. **At the end of your assessed cohort meeting, your Cohort Coach will assign one of these problems as a writeup. You may not collaborate on this writeup with your cohort-mates, but you may use notes taken before or during your assessed cohort meeting.**

Problem 2: Non-Deterministic Finite State Automata

Give a non-deterministic finite state automaton which decides each of the languages described below, using no more than the number of states indicated. Give both a drawing and a description.

(a) $\{x \in \{0,1\}^* \mid x$ contains the substring 0101$\}$, using no more than five states.

(b) The language described by $0^*1^*0^*$, using no more than three states.

(c) The language described by $1^*(001^*)^*$, using no more than three states.

(d) The language described by $((0 \mid 00)^*(1|\varepsilon))^*$, using no more than one state.

Problem 3: Regular Expression Matching in Software (programming)

The most common way to do regular expression matching in hardware is to convert the regular expression into a equivalent non-deterministic finite-state automaton, and then simulate the execution of that automaton on a particular string. While there is a wide diversity of techniques achieve the transformation and simulation (see Hyperscan, Grapefruit, for state-of-the-art examples), the core strategy is exactly the same.

I have written python code to perform regular expression matching using this strategy, employing the constructions exactly as mentioned in lecture. For this problem, your task is to use this tool to build a function that diagnoses Huntington's Disease.

Follow the instructions in the week 8 guideto complete this problem.

Problem 4: NFA Size

Regular expressions are represented by strings of text. Consider that we have a regular expression that is n characters long.

Show that the language represented by this regular expression can be computed by a NFA with $O(n)$ states (that is, the function from the length of the a regular expression to the number of states needed for an NFA that decides the language represented by that regular expression is in $O(n)$ where n is the length of the input). You may find the python code from the previous step to be helpful in forming an intuition for this problem.

Problem 5: Operations on Languages

In class, we showed that the Regular languages are closed under various operations. We showed, for example, that when regular languages L_1, L_2 are given as operands to Complement ($\overline{L_1}$), union ($L_1 \cup L_2$), and intersection ($L_1 \cap L_2$), the resulting language will also always be regular.

Show that the regular languages are closed under each of the operations below.

(a) NAND: a string x is in $NAND(L_1, L_2)$ if $\texttt{NAND}(x \in L_1, x \in L_2)$.

(b) NOR: a string x is in $NOR(L_1, L_2)$ if $\texttt{NOR}(x \in L_1, x \in L_2)$.

(c) XOR: a string x is in $XOR(L_1, L_2)$ if $\texttt{XOR}(x \in L_1, x \in L_2)$.

(d) Difference: a string x is in $SUBTRACT(L_1, L_2)$ if $x \in L_1$ and $x \notin L_2$.

Problem 6: Fixing Brunelle's Mistakes (Part 1)

In the *Equivalence of NFAs and DFAs* video, Professor Brunelle's method of converting an NFA to a DFA is not entirely correct. In particular, there is an error where he defines the DFA's transition function δ_D using the NFA's transition function δ as

$$\delta_D(s, \sigma) = \bigcup_{q \in s} \delta(q, \sigma).$$

The right-hand-side of this definition does not account for any ε transitions that might be present in the NFA.

(a) Give at least one NFA where converting using Brunelle's definition will result a DFA that computes a different language.

(b) Describe how we could correct this error (this may be informal).

(c) ($\star$) There's a least one other subtle problem in this construction, identify and correct that.

(One other subtlety with the definition is that sets of states in the DFA need to be mapped to labels for states in the NFA. The provided definition doesn't include the conversion between a set of DFA states and label for one NFA state. You can assume such conversions are done implicitly, that sets of DFA states are interpreted as you would expect as labels of NFA states.)

Problem 7: Fixing Brunelle's Mistakes (part 2)

In the Proving FSAs are as Powerful as Regular Expressions (Part 4: Kleene Star)video, there's a subtle but important error in the Kleene star construction. We made two modifications to a given NFA in order to make a new NFA to compute the Kleene star of the original language:

– Since we needed to return 1 on any string that was comprised of several strings on which the original NFA will return 1, we added empty transitions from every final state to the start state.

– Since we needed to return 1 on the empty string, we added a new final state to the machine, then added an empty transition from the start state to this new final state.

There is a flaw in the second modification mentioned above. While it achieves the goal of guaranteeing that the machine returns 1 when given the empty string as input, there are situations where applying both modifications to the machine results in it returning 1 for strings that do not belong to the Kleene star of the original language.

In other words: Consider we have a non-deterministic finite state automaton M which computes the language L. Let M' represent the result of applying the above modifications to M, and let L' be the language computed by M'. For some choices of M, $L^* \subset L'$.

For this problem, do both of the following:

a) Give an example of a machine where the above construction does not properly compute the Kleene star.

b) Fix the second modification above to correctly perform the Kleene star construction.

