Problem Set 5

This fifth problem set explores the computability topics from this week – DFAs and NFAs. As a reward for all the hard work you've put in over the past few weeks with the initial problem sets and the midterm, this problem set is designed to be easier than those in the past. I hope that this gives you a chance to play around with the material without costing you sleep and sanity.

Start this problem set early. It contains five problems (plus one survey question), several of which require a fair amount of thought. I would suggest reading through this problem set at least once as soon as you get it to get a sense of what it covers.

As much as you possibly can, please try to work on this problem set individually. That said, if you do work with others, please be sure to cite who you are working with and on what problems. For more details, see the section on the honor code in the course information handout.

In any question that asks for a proof, you **must** provide a rigorous mathematical proof. You cannot draw a picture or argue by intuition. You should, at the very least, state what type of proof you are using, and (if proceeding by contradiction, contrapositive, or induction) state exactly what it is that you are trying to show. If we specify that a proof must be done a certain way, you must use that particular proof technique; otherwise you may prove the result however you wish.

If you are asked to prove something by induction, you may use weak induction, strong induction, the well-ordering principle, or structural induction. In any case, you should state your base case before you prove it, and should state what the inductive hypothesis is before you prove the inductive step.

As always, please feel free to drop by office hours or send us emails if you have any questions. We'd be happy to help out.

This problem set has 125 possible points. It is weighted at 7% of your total grade. The earlier questions serve as a warm-up for the later problems, so do be aware that the difficulty of the problems does increase over the course of this problem set.

Good luck, and have fun!

Due Friday, November 4th at 2:15 PM

Problem One: Constructing DFAs (35 Points)

For each of the following languages over the indicated alphabets, construct a DFA that accepts precisely those strings that are in the indicated language. Your DFA does not have to have the fewest number of states possible, though the grading staff would appreciate it if you made an effort to keep your DFAs small. ☺ Unless otherwise noted, you should specify your DFA as either a state-transition diagram (the graphical representation we've seen in class) or as a table.

i. For the alphabet $\Sigma = \{0, 1, 2\}$, construct a DFA for the language L = { w | w contains exactly two 2s. }

ii. For the alphabet $\Sigma = \{0, 1\}$, construct a DFA for the language L = { w | w does not contain both 0 and 1. }

iii. For the alphabet $\Sigma = \{0, 1\}$, construct a DFA for the language L = { w | w contains the same number of instances of the substring 01 and the substring 10 }

iv. For the alphabet $\Sigma = \{a, b, c, \ldots, z\}$, construct a DFA for the language L = { w | w contains the word "cocoa" as a substring }. Remember that as a shorthand, you can specify multiple letters in a transition by using set operations on Σ (for example, $\Sigma - \{a, b\}$) *

v. For the alphabet $\Sigma = \{a, b, c, \ldots, z\}$, construct a DFA for the language L = { w | w contains the word "pirate" or the word "spire" as substrings }.†

vi. Suppose that you are playing the game of Nim with two piles, each of which has 2 stones in it at the beginning of the game. Let the alphabet $\Sigma = \{1_a, 2_a, 1_b, 2_b\}$, where 1_a represents the move "remove one stone from pile a," 1_b represents the move "remove one stone from pile b," etc. We can represent a game of Nim as a string of moves, where it is implied that the first symbol corresponds to the first player's first move, the second symbol corresponds to the second player's first move, the third symbol corresponds to the first player's second move, etc. For example, in the game $1_a\ 2_b\ 1_a$, the first player removes one stone from pile A, the second player removes two stones from pile B, and the first player then wins the game by removing one stone from pile A. Construct a DFA for the language L = { w | w is a string describing a legal game of Nim starting with two stones in each pile }.

vii. Suppose that you are taking a walk with your dog along a straight-line path. Your dog is on a leash that has length two, meaning that the distance between you and your dog can be at most two units. You and your dog start at the same position. Consider the alphabet Σ = {you, dog}. A string in Σ* can be thought of as a series of events in which either you or your dog moves forward one unit. For example, the string "you you dog dog" means that you take two steps forward, then your dog takes two steps forward. Let L = { w | w describes a series of steps that ensures that you and your dog are never more than two units apart }. Construct a DFA for this language.

---

* DFAs are the basis of a fast algorithm called the *Knuth-Morris-Pratt algorithm* for finding whether a string contains a given substring. The algorithm works by automatically constructing an automaton like the one you'll be building in this problem, then running the automaton on the string to be searched.

† DFAs are also the basis of the *Aho-Corasick algorithm* for finding all instances of a *set* of strings in some larger piece of text. The algorithm automatically constructs an automaton like the one you'll build in this problem.

Problem Two: Constructing NFAs (25 Points)

For each of the following languages over the indicated alphabets, construct an NFA that accepts precisely those strings that are in the indicated language. Unless otherwise noted, you should specify your NFA as either a state-transition diagram (the graphical representation we've seen in class) or as a table. Your NFA may use ε-transitions if you wish.

i. For the alphabet $\Sigma = \{0, 1, 2\}$, construct an NFA for the language { w | w ends in 0, 11, or 222. }

ii. For the alphabet $\Sigma = \{a, b, c, d, e\}$, construct an NFA for the language { w | some character of Σ is not present in w } *(Hint: You may find it very useful to use ε-transitions here.)*

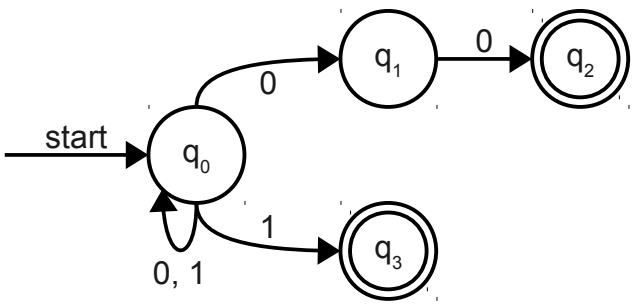
iii. For the alphabet $\Sigma = \{a, b, c, d, e\}$, construct an NFA for the language { w | the last character of w appears nowhere else in the string, and $|w| \geq 1$ }

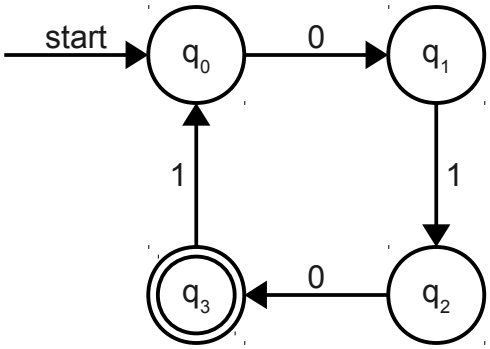
iv. Suppose that you are given a directed graph $G = (V, E)$ where $V = \{v_1, v_2, \ldots, v_n\}$. Consider the alphabet $\Sigma = V$ (that is, the set of symbols is equal to the set of nodes in the graph), so a string in this language corresponds to series of nodes in G. Let L = { w | w is a legal path in G }. Show how to define an NFA that accepts L by writing out a five-tuple for L and giving a specification of Q, Σ, δ, q_0, and F.

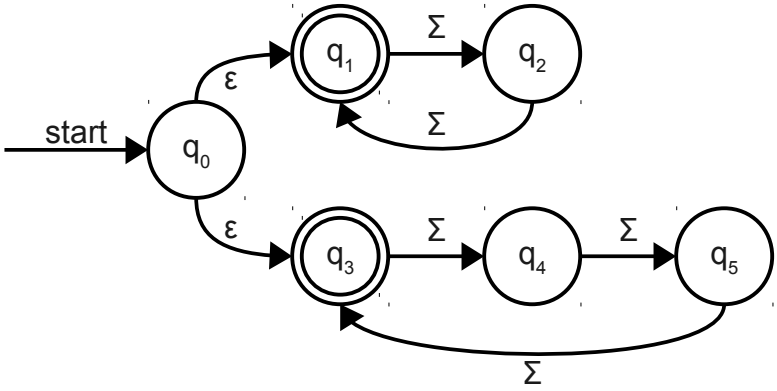
v. Suppose that you are given a set of buckets $B = \{b_1, b_2, \ldots, b_n\}$. You start adding objects into buckets one at a time, and are interested in seeing whether some bucket ends up holding more than one object. Given Σ = B, a string in Σ* corresponds to a series of objects put into buckets, where each symbol in the string indicates what bucket an object is put into. For example, the string $b_0\ b_1\ b_3\ b_0$ represents putting some object into bucket b_0, then putting an object into bucket b_1, then putting an object into bucket b_3, and then putting another object into bucket b_0. Let L = { w | w describes a series of objects distributed into buckets where some bucket contains multiple objects }. Show how to define an NFA that accepts L by writing out a five-tuple for L and giving a specification of Q, Σ, δ, q_0, and F.

Problem Three: The Subset Construction (20 Points)

Below are three NFAs over the language $\Sigma = \{0, 1\}$. For each NFA, convert that NFA into an equivalent DFA using the subset construction. You may represent the new DFA as either a state-transition diagram or as a table, though it is *much* easier to do so using a table. You must use the subset construction to convert the NFA to a DFA; if you simply provide an equivalent DFA that wasn't produced by the construction, you will receive no credit.







Problem Four: Finite Languages (20 Points)

A **finite language** is a language L such that |L| is finite, meaning that it contains only finitely many strings. In this problem, you will prove that any finite language is regular. This shows that DFAs and NFAs are powerful enough to recognize any finite set of words, including natural languages like English, legal sequences of moves in most games, etc.

i. Prove that Ø is a regular language.

ii. Prove that if *w* is a string, then {*w*} is a regular language.

iii. Using your results from (i) and (ii), prove that any finite language is regular (a finite language is a language L such that |L| is finite).

Problem Five: Inherent Complexity (20 Points)

Consider the language L = { w | |w| is a multiple of k } for any alphabet Σ and any positive natural number k. Prove that if D = (Q, Σ, δ, q_0, F) is a DFA such that L(D) = L, then |Q| ≥ k. This shows that there are some languages that require inherently large DFAs to accept. *(Hint: The pigeonhole principle might be useful here.)*

Problem Six: Course Feedback (5 Points)

We want this course to be as good as it can be, and we'd really appreciate your feedback on how we're doing. For a free five points, please answer the following questions. We'll give you full credit no matter what you write (as long as you write something!), but we'd appreciate it if you're honest about how we're doing.

i. How hard did you find this problem set? How long did it take you to finish?

ii. Does that seem unreasonably difficult or time-consuming for a five-unit class?

iii. Did you attend Monday's problem session? If so, did you find it useful?

iv. How is the pace of this course so far? Too slow? Too fast? Just right?

v. Is there anything in particular we could do better? Is there anything in particular that you think we're doing well?

Submission Instructions

There are four ways to submit this assignment:

1. Hand in a physical copy of your answers at the start of class.

2. Submit a physical copy of your answers in the filing cabinet in the open space near the handout hangout in the Gates building. If you haven't been there before, it's right inside the entrance labeled "Stanford Venture Fund Laboratories." There will be a clearly-labeled filing cabinet into which you can submit your homework.

3. Submit a physical copy of your answers in the drop-off box in the Gates basement. We'll send out an email announcement about the location of this box on the night of Friday, November 4.

4. Send an email with an electronic copy of your answers to **cs103@cs.stanford.edu**

For those of you who are interested in playing around with more advanced material, here are some challenge problems that will give you a chance to explore automata theory. Each of these problems are worth ten extra credit points, so there's a total of twenty points extra credit you can earn by working through these problems. Good luck, and have fun!

Extra Credit Problem: Redefining Acceptance

In a traditional DFA or NFA, the automaton accepts a string if, once all the characters of the string have been consumed, the automaton is in an accepting state. However, we may want to consider what strings, when run through a DFA, end up passing through some accepting state, or perhaps end up passing through some specific set of accepting states.

i. Suppose that $D = (Q, \Sigma, \delta, q_0, F)$ is a DFA. Prove that the language $L = \{ w \mid$ when D is run on w, the machine at some point enters an accepting state$\}$ is regular.

ii. Suppose that $D = (Q, \Sigma, \delta, q_0, F)$ is a DFA. Prove that the language $L = \{ w \mid$ when D is run on w, the machine at some point enters every accepting state $\}$ is regular.

Extra Credit Problem: Universal Nondeterminism

An NFA is a nondeterminstic automaton that uses *existential nondeterminism*, which says that the automaton accepts a string *w* if there is some possible choice of transitions it can make that would result in the automaton accepting the string. However, there is another type of nondeterminism called *universal nondeterminism* that says that the automaton accepts a string *w* if **for every** possible choice of transitions the automaton accepts the string. We can define a new type of automaton called a *UFA* (for **universal finite automaton**) that is similar to an NFA but uses universal nondeterminism instead of existential nondeterminism.

i. Consider the problem description for Problem 2.v (putting objects into buckets). Draw a state transition diagram for a UFA that accepts the language $L = \{ w \mid$ w describes a series of objects put into buckets such that no bucket contains more than one object $\}$, assuming that there are five buckets.

ii. Using the definition of the language of an NFA as a starting point, come up with a formal definition for the language of a UFA and justify why it is correct.

iii. Prove that if L is a regular language, there is some UFA that accepts L.

iv. Prove that if $L = L(U)$ for some UFA, then L is a regular language. *(Hint: Consider a modification of the subset construction)*

The UFA is a special case of a more powerful automaton called an AFA (for **alternating finite automaton**) that combines the NFA and UFA by allowing states to specify whether their transitions use existential or universal nondeterminism. As with the NFA and UFA, the AFA is equivalent in power to a standard DFA and thus accepts just the regular languages.