Advanced Exploration 4: Mathematical Models for Computation

MATH2603: Discrete Mathematics

Overview: In this Advanced Exploration, you'll learn about some mathematical models for computation, and explore some of their limitations. Many problems in computer science ask for a computer algorithm that performs a certain task (e.g., sort a list, or count the length of a string). You will learn about some simple mathematical structures (finite state machines, and Turing machines) that are abstract models of computation. It would not be practical to use these models to actually solve computer science problems. Rather, they were conceived by mathematician to study *the limits of computers*; there are problems that **cannot be solved by any computer program.** To emphasize, that doesn't mean no one has been smart enough to devise a program to solve these problems yet; it means that no matter how hard you try, no matter how smart you are, you can't write a program to solve these problems.

Instructions: Read Chapter 17 in the textbook (Languages, Automata, Turing Machines); you may have seen some of this material in other classes. Next, complete the exercises below. Your submission should be self-contained; so you should explain what the questions asked, and give context to your work before answering any questions. Your solutions should be complete, clear, and correct. Your solutions should also be accompanied by explanations written in complete sentences that justify your work.

Instructions for submitting your work, and information on how the EMRN rubric will be applied to evaluate this exploration, are at the end of the assignment.

Background: An *alphabet* Σ is a set of letters (with which one can make strings); as we have seen in class, given an alphabet Σ, the set of all strings consisting of letters from the alphabet is denoted by Σ^*. A **language** $\mathcal{L}$ is simply a subset of Σ^*; elements of $\mathcal{L}$ are the "words" in the language.

We will focus on computational models that take a string as input, and output either `YES` or `NO`. The book explains (17.1.6) how to construct a finite state machine that takes as input a string s in $\{0,1\}^*$, then accepts if s has even parity, and rejects if s has odd parity. By definition, the set of all strings accepted by this machine (i.e., those strings which have even parity) form a language in the alphabet $\{0,1\}$. In fact, given an alphabet Σ and a FSM F, one can define a language as

$$\mathcal{L}_{\mathcal{F}} = \{w \in \Sigma^* \mid F \text{ answers } \texttt{YES} \text{ on input } w\}$$

Below, you will study the connection between some computational models and the languages that they accept.

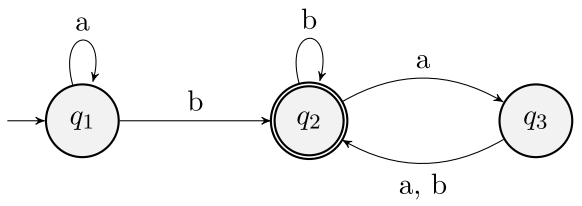
The Problem: Thasper has been learning about computational models in his computer science class, but has really been struggling. In his last class, the instructor defined a **regular language** $\mathcal{L}$ as a language for which there exists a FSM F, such that $\mathcal{L}_F = \mathcal{L}$: so a regular language is one that is "the language accepted by a FSM." We say that F **recognizes** $\mathcal{L}$ or that it **accepts** $\mathcal{L}$.

1. Help Thasper understand regular languages by constructing a finite state machine which accepts the langauge in the alphabet $\Sigma = \{a, b\}$ defined by

$$\mathcal{L} = \{w \in \Sigma^* \mid w \text{ ends in an } a\}$$

You may depict your FSM with a decorated digraph as in the book. Be sure to explain how your FSM works and why it accepts precisely the correct language.

2. Next, consider the finite state machine depicted below



   Describe in complete English sentences the language $\mathcal{L}$ recognized by this FSM. Give examples of words that are accepted and words that are rejected. Then, give a recursive definition for the set $\mathcal{L} \subseteq \Sigma^*$.

3. If you have not already done so, read the section of the textbook on the Pigeonhole Principle. Consider the language defined by

$$\mathcal{H} = \{w \in \Sigma^* \mid w \text{ has the same number of } a\text{'s and } b\text{'s}\}$$

   Explain to Thasper why the language $\mathcal{H}$ cannot possibly be a regular language. *(Hint: The Pigeonhole principle is relevant)*

A new computational model: Thasper has just learned of a new type of computational model that, according to his instructor, is more powerful than a FSM, but not as powerful as a Turing machine. A **stack** is a data structure in computer science that acts like a queue or a stack of papers: you can "push" an item to the stack (placing it on top of the pile), and you can "pop" an item from the stack (remove it from the pile). There is no direct way to access items that are not on top of the stack: the user must remove (pop) items from the top until the desired item is reached.

A **pushdown automaton** or **PDA** is an automaton that has a stack as its memory. A PDA can push a letter to its stack, or pop a letter from its stack, but cannot read the tenth item in the stack without first removing the top nine. (You can read more about PDAs, including how to draw one, on wikipedia) A **deterministic context-free language** $\mathcal{L}$ is one that is accepted by some PDA (all machined discussed in this AE are deterministic).

4. Explain to Thasper why a PDA can accept the language

$$\mathcal{I} = \{w \in \Sigma^* \mid w = a^n b^n\}$$

   even though a FSM cannot. Be explicit, and use examples.

5. Convince Thasper that the language $\mathcal{H}$ defined above is context free by constructing a push-down automaton which accepts $\mathcal{H}$. Explain how your PDA works, and why it accepts the language you claim.

Submitting your work: Your work must be neatly typed up using a system that supports mathematical notation. For example, you can use MS Word and its equation editor; or you can write your work in a Jupyter notebook using Markdown and LaTeX. Once it is written up, the work must be saved as a PDF file and then uploaded as a PDF to the area on Blackboard where the original assignment is located. Remember that the work is not actually submitted until you upload the file and click the “Submit” button. Grading and feedback will take place entirely on Blackboard. The following are not allowed: Submissions outside Blackboard (for example through email); files that are not in PDF form; and work that contains any handwriting, though you may *draw a diagram* neatly by hand, scan it, and include it in your submission.

Evaluation: Like all Advanced Explorations, your work will be evaluated using the EMRN rubric. Please see the statement of this rubric in the syllabus for an explanation of how it is used. When applied to this Advanced Exploration, the following criteria help to assign the grade:

- **E:** The submission consists of a clear, correct, and complete solution. The solution contains no major errors (computation, logic, syntax, or semantic); it is also exceptionally clear and the writeup is professional in its look and style. The solution would be at home in a professional lecture or publication.
- **M:** The submission consists of a clear, correct, and complete solution. The solution contains no major errors (computation, logic, syntax, or semantic) and is neatly and professionally written up.
- **R:** The solution contains at least one, but not several, major errors (computation, logic, syntax, and/or semantic) that require revision. An “R” may also be given for writeups that do not expend sufficient effort to produce a good-looking writeup, or do not provide enough context for the submission to be self-contained.
- **N:** The solution has several significant errors; or the submission is missing large portions of the solution; or the solution is for a significantly altered version of the problem; or the submission is excessively cluttered, messy, difficult to read, or handwritten.