A Brief Review of Functions

A **function** f from A to B takes as input an element from the set A, and returns an element from the set B. This is often abbreviated as $f : A \to B$.

Example: f takes as input a nonnegative integer, and outputs its square. We could say $f : \mathbb{N} \to \mathbb{N}$. We could also say $f : \mathbb{N} \to$ perfect squares.

- $f : A \to B$ is an **injective** function if every input maps to a different output. That is, if $f(a) = f(b)$, then $a = b$. This is also called a **one-to-one** function.
- $f : A \to B$ is a **surjective** function if there is a way to produce every element in B as output. That is, if $b \in B$, then there is an $a \in A$ such that $f(a) = b$. This is also called a **onto** function.
- $f : A \to B$ is a **bijective** function if it is both one-to-one and onto. This is also called a **one-to-one correspondence**.

Question 1. Given the above definitions, suppose we have two sets A and B. What can we say about the existence of a function from A to B in each of the following circumstances? Recall that $|X|$ represents the *cardinality* of the set X.

- $|A| = |B|$; that is, A and B are *equinumerable*.
- $|A| \leq |B|$
- $|A| < |B|$

Countable Sets and Establishing Bijections

We use the symbol $\mathbb{N}$ to represent the set of *natural* numbers. Depending on which math book(s) you read, you might find slightly different definitions of the symbol. *For purposes of CompSci 162*, this is the set of non-negative integers, or $\{0, 1, 2, \ldots\}$. This set is *infinite* in size. When you hand-write aspects for this class, you may wish to simply use N.

Any set A that can be put into one-to-one correspondence with $\mathbb{N}$ is said to be *countably infinite*. When we write the size of any such set, we write $|A| = \aleph_0$, pronounced "aleph null."

Let's determine the size of the following sets. **In this class, for all graded artifacts, unless otherwise stated, you will need to justify any such answer.**

Question 2. What is the size of the set of all *odd natural numbers*. That is, $A = \{1, 3, 5, \ldots\}$?

Question 3. What about the set of integers, $\mathbb{Z}$? Recall that $\mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$.

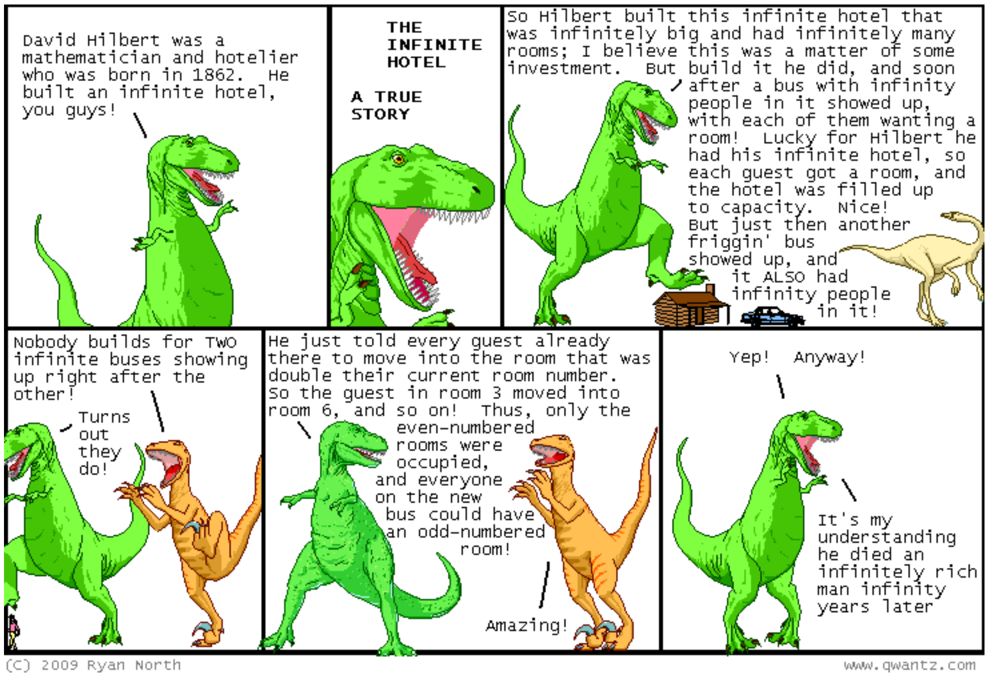

David Hilbert was a mathematician and hotelier who was born in 1862. He built an infinite hotel, you guys!
THE INFINITE HOTEL
A TRUE STORY
So Hilbert built this infinite hotel that was infinitely big and had infinitely many rooms; I believe this was a matter of some investment. But build it he did, and soon after a bus with infinity people in it showed up, with each of them wanting a room! Lucky for Hilbert he had his infinite hotel, so each guest got a room, and the hotel was filled up to capacity. Nice! But just then another friggin' bus showed up, and it ALSO had infinity people in it!
Nobody builds for TWO infinite buses showing up right after the other!
Turns out they do!
He just told every guest already there to move into the room that was double their current room number. So the guest in room 3 moved into room 6, and so on! Thus, only the even-numbered rooms were occupied, and everyone on the new bus could have an odd-numbered room!
Amazing!
Yep! Anyway!
It's my understanding he died an infinitely rich man infinity years later
(C) 2009 Ryan North
www.qwantz.com

Figure 1: Dinosaur Comics # 1575: https://www.qwantz.com/index.php?comic=1575

Recall that the **Cartesian Product** of two sets $A \times B = \{(a, b) | a \in A \wedge b \in B\}$. That is, the set of all possible ordered pairs (a, b) where $a \in A$ and $b \in B$.

If $A = \{1, 2\}, B = \{1, 3, 4\}$ then $A \times B = \{(1, 1), (1, 3), (1, 4), (2, 1), (2, 3), (2, 4)\}$.

Question 4. What about the set of all *ordered triples* of $\mathbb{N}$? That is, $\mathbb{N} \times \mathbb{N} \times \mathbb{N}$?

Schröder-Bernstein Theorem

The Schröder-Bernstein Theorem has two parts:

1. Let A and B be arbitrary sets. If there exist injections $f : A \to B$ and $g : B \to A$, then there exists a bijection from A to B.
2. Let A and B be arbitrary sets. If $|A| \leq |B|$ and $|B| \leq |A|$, then $|A| = |B|$.

Question 5. Use the Schröder-Bernstein Theorem to answer the previous question.

Question 6. Show that the set of rational numbers, $\mathbb{Q}$, is countable.

Question 7. How large is $\mathbb{R}$, the set of *real* numbers?

The *power set* of A, written as $\mathcal{P}(A)$ or 2^A, is the set of all subsets of A.
Question 8. *Cantor's Theorem* : The power set of any set has cardinality strictly greater than the original set.

What is a Formal Language?

What is a String?

I assume you are familiar with a `std::string` in C++ and its equivalents in other programming languages. Please be sure to keep these concepts separate.

An *alphabet* is a non-empty finite set whose elements are referred to as *letters* or *symbols.* A *string* is a finite (possibly empty) sequence of symbols over some alphabet. Strings can be concatenated with other strings or letters.

The symbol Σ is typically used to denote an alphabet. The number of letters in string w is $|w|$ and $w[i]$ is the ith symbol of w. Unlike `std::string`s, the first symbol of w is $w[1]$. We use w^R to denote the reversal of w. The empty string is denoted ε ("varepsilon" for those of you using LaTeX).

A string or symbol with a natural number exponent, such as a^n, denotes the string consisting of n concatenated copies of a.

Σ^* is the *Kleene Star* of Σ, the set of all strings over Σ.

What's a Language?

Note that this is different from the definition you might get if you wanted to describe English or Spanish as a "language."

A *language* is a set of strings. If Σ is an alphabet, then every $L \subseteq \Sigma^*$ is a language.

Here are some example languages over the alphabet $\Sigma = \{a, b\}$.

$$\begin{aligned}
L_1 &= \{a, abb, aaaa\} \\
L_2 &= \{a^n | n \in \mathbb{N} \text{ is prime}\} \\
L_3 &= \{b^n a^n b^m \mid n, m \in \mathbb{N} \textit{ and } n \equiv m \pmod 3 \ \} \\
L_4 &= \text{The set of all } w \in \Sigma^* \text{ with at most three } a\text{'s} \\
L_5 &= \{a^n \mid n \in \mathbb{N} \text{ and } \exists x, y, z \in \mathbb{N} - \{0\} \text{ such that } x^n + y^n = z^n\}
\end{aligned}$$

Question 9. Which of the above five languages are finite? Which are infinite? How infinite?

Note that even the infinite languages above have a *finite representation.* We can think of the definitions themselves as strings in a *meta language*, in this case, a superset of English and formal mathematics.

Question 10. Are all languages (over a given alphabet) defined by *some* string in a suitable meta-language?

Operations on Languages

Since languages are sets, the traditional set operations can be used. If L and L' are languages, then so are:

$$\begin{aligned} L - L' &= \{w \in L \mid w \notin L'\} \\ \overline{L} &= \Sigma^* - L \\ L^* &= \{w_1, w_2, \ldots w_n \mid n \in \mathbb{N}, w_i \in L\} \\ LL' &= \{uv \mid u \in L, v \in L'\} \end{aligned}$$

Deterministic Finite Automata

Associated reading: Sipser, 1.1, Lewis/Papadimitriou 2.1

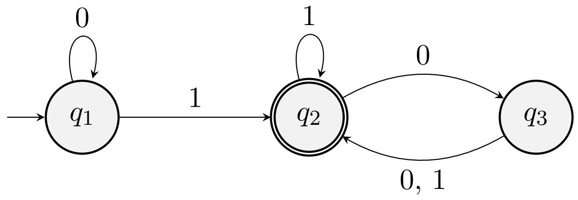


Figure 2: Figure 1.4 from Sipser text; a finite automaton with three states

Question 11. What happens if we feed the input string "1101" into the machine in Figure 2?

Formal Definition

A **finite automaton** is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where:

1. Q is a finite set called the *states*
2. Σ is a finite set called the *alphabet*
3. $\delta : Q \times \Sigma \to Q$ is the *transition function*
4. $q_0 \in Q$ is the *start state*
5. $F \subseteq Q$ is the *set of accept states*

Formal definition of computation

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a finite automaton and let $w = w_1 w_2 \ldots w_n$ be a string where each w_i is a member of the alphabet Σ. Then we say that M accepts w if a sequence of states $r_0, r_1, \ldots, r_n$ in Q exists with three conditions:

1. $r_0 = q_0$
2. $\delta(r_i, w_{i+1}) = r_{i+1}$ for $i = 0 \ldots n-1$
3. $r_n \in F$

Any language recognized by some finite automaton is called a *regular language.*

Question 12. Formally show that the machine in Figure 2 accepts the string "1101" – this will likely be the only time you do such a thing.

Question 13. Design a DFA over the alphabet $\Sigma = \{a, b\}$ that accepts all strings with an odd number of instances of the letter a.

Regular Languages and Regular Expressions

Given an alphabet Σ, we can represent a class of languages over Σ by using a special syntax called a *regular expression*[1]. Any language defined by such is called a *regular language*, the first important category we are going to cover in CompSci 162. We assume for this that any alphabet we consider does not contain the symbols '(', ')', '*', '$\cup$', or '$\emptyset$.

There are four rules for a regular expression:

1. Any given character in the language, or $\emptyset$, is a valid regular expression. This means if you want to include any single character, or the empty string, you may do so.
2. If u and v are both valid regular expressions, $(u \cup v)$ is a valid regular expression. This means that the regular language we are describing includes everything that can be described by regular expression u or v (or both).
3. If u and v are both valid regular expressions, (uv) is a valid regular expression. This is *concatenation*, and means that this regular expression accepts any string described by regular expression u that is immediately followed by one described by regular expression v.
4. If u is a valid regular expression, $(u)^*$ is a valid regular expression. This means that if u is a valid regular expression, then if you repeat it zero or more times (including none), it is included by this regular expression[2].

This all seems very abstract, so let's do some exercises instead. For each of these, we assume the alphabet Σ is $\{a, b\}$.
Question 14. Describe the set of strings accepted by this regular expression: a^*ba^*.

Question 15. Describe the set of strings accepted by this regular expression: $a\Sigma^*a \cup b\Sigma^*b \cup a \cup b$

Question 16. Give a regular expression for the language that includes every string that has "aab" as a substring.

[1]Note that some programming languages have a feature they call a "regular expression" that does not always match these. In CompSci 162, you'll need to use this syntax, or something similar to it, when asked for a regular expression. Sometimes, a programming language's "regular expression" is actually a Context-Free Grammar, which we'll discuss in the next unit.

[2]Some programming language have shorthand ways to say "two or more of this RegEx" but we won't use that here. Sometimes $u(u)^*$ is abbreviated u^+, which is okay to use in CompSci 162.

Question 17. Give a regular expression for the language that includes every string wherein every a is followed by at least one b.

Non-Deterministic Finite Automata (NFA)

We saw *deterministic* finite automata (DFAs). In these, there was always exactly one valid transition at each step of computation. We are going to generalize this into the concept of non-deterministic finite automata, wherein a transition might not be fully determined by the state of the machine and the next symbol. In fact, it is possible for the machine to be in one, many, or even no states. Transitions can be symbols or strings, including the empty string. It is also valid to have an ambiguous transition. I will draw what this looks like in lecture. Lastly, it is also possible to not have a transition specified.

We say that an NFA M *accepts* a string if and only if there is a sequence of choices made while reading the string that leaves the machine in an accept state after reading the string.

Let's see an example of when this definition helps us.

Question 18. Give an NFA that recognizes the set of strings over $\Sigma = \{a, b\}$ where the third-to-last character in the string is a b. For example, $aaabaa$ is accepted but $aabb$ is not.

Question 19. Give a DFA that recognizes the same set of strings.

Question 20. What does the technique I used to answer the previous question tell you about the relationship between the set of *NFA-acceptable languages* (those for which there is an NFA that accepts it) and the set of *DFA-acceptable languages*?

Question 21. Give an NFA over the alphabet $\Sigma = \{a\}$ which accepts all strings whose length is a multiple of 2 or a multiple of 3 (or both). Do you believe you can draw a DFA that accepts these? Why or why not?

Equivalence of NFA/DFA-acceptable and Regular Expressions

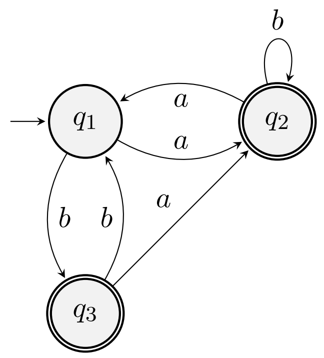
Question 22. Given a regular expression, draw an NFA that accepts the same set of strings.

Question 23. Given a DFA, how can we convert it to a regular expression?

Question 24. What does the result of the two previous questions, combined with what we learned earlier this week, tell us about the relationship between languages accepted by NFAs, DFAs, and regular expressions?

Question 25. Make an NFA that accepts $(ab \cup a)^*$.

Question 26. Convert the following DFA to a regular expression



Question 27. Prove that regular languages are closed under complementation. That is, if L is a regular language, then $\overline{L} = \Sigma^* - L$ is also a regular language.

What are not Regular Languages?

It may seem so far that lots of families of strings are regular languages. What sorts of things are not? After all, there are uncountably infinite languages over a given alphabet, and only countably infinite descriptors. We saw already how to prove a language **is** regular, now we need to know how to prove when a language **is not** regular.

Question 28. Given a regular language, how can you tell (in principle) if it is *infinite*?

Question 29. Given a regular language, how can you tell (in practice) that it is infinite?

Question 30. What is the pumping lemma[3]?

Question 31. Use the pumping lemma to prove that $\{a^n b^n \mid n \geq 1, n \in \mathbb{N}\}$ is **not** a regular language.

[3]This will be printed in the slides, but I recommend writing it down and referencing *what you wrote* when you see the work for the next question. There will be a question on exam 1 that uses it in the same manner that the next question does. I can't check that you approach the question this way "during lecture," but I very much believe you will learn the material better if you do.