1 More finite deterministic automata

Exercise. *Consider the following game with two players: Repeatedly flip a coin. On heads, player 1 gets a point. On tails, player 2 gets a point. A player wins (and the game ends) as soon as they are ahead by two points. Draw a DFA that recognizes the language of strings (with alphabet $\{H, T\}$) which represent a possible series of flips in which player 1 wins.*

Exercise. *Show that a DFA with n states accepts an infinite language if and only if it accepts some string of length at least n.*

This is also known as the Pumpkin lemma.[1] Happy Halloween!

2 More Myhill-Nerode

Recall the statement of the Myhill-Nerode theorem from class:

Theorem 1 (Myhill-Nerode)**.**
A language $L \subset \Sigma^*$ is regular if and only if there are only finitely many equivalence classes under the

[1] Er... did we say pumpkin? We meant pumping :)

following relation $\sim_L$ on Σ^*:

$$x \sim_L y \iff \forall z \in \Sigma^* : xz \in L \iff yz \in L.$$

Moreover, the minimum number of states in a DFA for L is exactly the number of such equivalence classes.

Proof. ($\implies$) Suppose L is regular. Then there is a DFA M that recognizes L, and for each $x \in \Sigma^*$ there is a state q_x such that on input x, M ends at state q_x. Then, define a relation $\sim_M$ on Σ^* by[2]

$$x \sim_M y \iff q_x = q_y$$

Clearly,

$$x \sim_M y \implies (\forall z \in \Sigma^* : xz \in L \iff yz \in L) \iff x \sim_L y.$$

It follows that each equivalence class of $\sim_M$ is contained entirely in some equivalence class of $\sim_L$, and hence each equivalence class of $\sim_L$ is a union of several equivalence classes of $\sim_M$. Thus, as there are finitely many of the latter, there must be finitely many of the former.

($\impliedby$) In the lecture notes! □

Exercise. *For each of the following languages, determine whether the language is regular or non-regular, and prove your answer:*

1. $\{a^n b a^n \mid n \geq 1\}$.
2. $\{a^{2^n} \mid n \geq 1\}^*$
3. $\{a^n b a^m b a^{m+n} \mid m, n \geq 0\}$

3 Closure properties

On this week's problem set, you will prove that regular languages are closed under homomorphisms and inverse homomorphisms. How might we go about showing closure properties under other natural operations, such as union, intersection, and set difference?

Exercise. *Prove that the complement of a regular language is also regular.*

[2] Especially if you are unfamiliar with equivalence relations, it is work checking that $\sim_M$ satisfies the conditions for one.

3.1 The product construction.

The product construction is a way to combine two DFAs into a single one that keeps track of both computations simultaneously and independently. Simply, it is a formal implementation of what you would do if you had to run two DFAs on the same string, but you could only pass over the word once (i.e., you're given the kind of access a single DFA would have to the string).

Given DFAs on the same alphabet $M = (Q, \Sigma, \delta, q_0, F), M' = (Q', \Sigma, \delta', q_0', F')$, we define a new DFA $M \times M'$ whose states are $Q_\times = Q \times Q'$, with initial state (q_0, q_0'), and transition function $\delta_\times((q, q'), \sigma) = (\delta(q, \sigma), \delta'(q', \sigma))$ (and let's leave the set of final states unspecified for now).

Then an easy induction shows that

Lemma 1.
On input w, if M, M' are in states q_i, q_i' after reading the first i symbols, $M \times M'$ is in state (q_i, q_i').

Getting the above statement was our initial motivation for coming up with the product construction, and it shows us how to construct DFAs that recognize $L(M) \cup L(M')$ and $L(M) \cap L(M')$.

3.2 Closure under union, intersection, and difference

Exercise. *How does the product construction help us establish closure under union, intersection, and difference?*

3.3 Concatenation and Kleene star

When you think about it for a while, for these closure properties, it is more convenient to use the NFA model, because we intuitively want to make ε transitions. Indeed, if you go back to the previous problem, you can find a fairly straightforward NFA construction for the union of two regular languages.

Exercise. *How would you make an NFA that recognizes the concatenation of the languages $L(N), L(N')$ for two NFAs N, N'?*

Exercise. *How would you make an NFA that recognizes the Kleene star of the language $L(N)$, for an NFA N?*

Exercise. *Let L be the language of all strings over the English alphabet that contain exactly one of the strings 'alan', 'mathison', 'turing'. Is L regular?*