Recitation 02: NFAs, Pumping Lemma

In this recitation, we'll continue our discussion of regular languages. We'll introduce NFAs (nondeterministic finite automata) as an equivalent model of computation to DFAs (deterministic finite automata). Next, we'll practice techniques for showing that languages are *not* regular. That means, for a given language A, we have to show that *no* finite automaton M recognizes A. This seems harder than proving a language is regular, where we only have to construct *some* FA recognizing A. This is where the **pumping lemma** and closure properties of regular languages come in handy.

Nondeterministic Finite Automata

Definition 1 *A nondeterministic finite automaton is a 5-tuple* $(Q, \Sigma, \delta, q_0, F)$*, where*

1. Q *is a finite set of states,*
2. Σ *is a finite alphabet,*
3. $\delta : Q_\epsilon \times \Sigma_\epsilon \to \mathcal{P}(Q)$ *is the transition function,*
4. $q_0 \in Q$*, is the start state, and*
5. $F \subseteq Q$ *is the set of accept states.*

$\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$. This means that an NFA can have ϵ-arrows in its transition function moving from one state to another *without reading any input*.

The range of δ is $\mathcal{P}(Q)$, the power set of Q. This means that, for a given state/symbol pair (q, a), an NFA will *nondeterministically* transition to the set of states $\delta(q, a) \in \mathcal{P}(Q)$, resulting in branches of computation running *in-parallel*.
If a branch of computation encounters the scenario $\delta(q, a) = \emptyset$ (i.e. there are no transition arrows for input a on a given state q), then it simply *dies*.

NFAs are similar to DFAs, except that NFAs are allowed to have ϵ transitions and multiple transitions for a given symbol. This means that an NFA can have multiple branches of computation running in parallel for the same input, and if and only if any of them accept, the NFA accepts. With that, let's do a practice example.

Example 1 *Show that regular languages are closed under reversal. Reversal of a language A is defined as follows:*

$$A^R = \{w^R \mid w \in A\}$$

Solution 1 *Let A be a regular language. Then $L(M) = A$ for some DFA M. The idea is to construct an NFA N that recognizes A^R. A natural first*

step is to reverse the direction of arrows of M. Using this approach, note that if we had multiple arrows going into a state q in the original DFA M, we'll have multiple arrows going out of q in the NFA N. But that's OK, as we're building an NFA. For example, if $\delta_M(q_1, a) = \delta_M(q_2, a) = q$*, we would now have* $q_1, q_2 \in \delta_N(q, a)$*.*

The next step would be to make M's accept states become start states, and vice versa. This doesn't work immediately, as M could have no or multiple accept states, but an FA should have exactly one *start state. We fix this by adding an extra state* q_0 *to be the start state of N. We then add* ϵ*-transitions from* q_0 *to all the states that were accept states in M. Another way to think about this is that, since we're looking at a string w in reverse order, we're guessing which accept state M uses to accept w and go backwards from there.*

Pumping Lemma

The pumping lemma is most directly a property that all regular languages have, which informally states that "regular languages can be pumped." However, we mostly use it in the contrapositive sense to prove a language is *not* regular by showing that it *cannot* be pumped.

Lemma 1 *If A is a regular language, then there exists a number p (the pumping length) where, for any string* $s \in A$ *of length at least p, then s may be divided into three pieces,* $s = xyz$*, satisfying the following conditions:*

1. *for each* $i \geq 0$*,* $xy^iz \in A$*,*
2. $|y| > 0$*, and*
3. $|xy| \leq p$

To proceed with a proof by contradiction, we have to find **some** string s where $s \in A$ and $|s| \geq p$ such that **for every** partition $s = xyz$, **at least** one of the three conditions above is violated.

Example 2 *Show that* $A_1 = \{0^n1^m0^n\}$ *for non-negative integers* n, m *is non-regular.*

Solution 2 *Assume for contradiction that* A_1 *is regular. Then* A_1 *must have a pumping length p. Consider the string* $s = 0^p10^p$*. Note that* $s \in A_1$ *and* $|s| = 2p + 1 \geq p$*. Consider* ***any*** *partition* $s = xyz$ *that satisfies the three conditions of the pumping lemma. By condition 3, we must have* $|xy| \leq p$*. This implies that y is all 0's. As* $|y| > 0$ *by condition 2, if we pump up (i.e. use condition 1 with* $i > 1$*), then* xy^iz *will have more 0's than 1's and can't be in* A_1*, contradiction.*

One strategy is to incorporate the pumping length p into string s so that y, which is within the first p symbols, is forced to be under some condition, e.g. y has to be all 0's.

Example 3 *Show that* $A_2 = \{0^n1^m \mid n \geq m\}$ *is non-regular.*

Solution 3 *Assume for contradiction that* A_2 *is regular. Then* A_2 *must have a pumping length p. Consider the string* $s = 1^p0^p$*. Note that* $s \in A_2$ *and*

Example 3 illustrates that **pumping down** is also a valid strategy for a proof by contradiction.

*$|s| = 2p \geq p$. Consider **any** partition $s = xyz$ that satisfies the three conditions of the pumping lemma. By condition 3, we must have $|xy| \leq p$. This implies that y is all 1's. As $|y| > 0$ by condition 2, if we pump down (i.e. use condition 1 with $i = 0$), then xy^iz will have more 0's than 1's and can't be in A_2, contradiction.*

Closure under Intersection

We can also use closure properties of regular languages to show a language cannot be regular. Recall that the class of regular languages is closed under the regular operations (union, star, concatenation), as well as intersection, reversal (this recitation), and complement (Problem 0.1 on the first Problem Set).

Example 4 *Show that $B = \{a^m b^n \mid m \neq n\}$ is non-regular.*

Solution 4 *Using the pumping lemma to directly prove B is not regular is quite difficult, though it can be done. Instead, the strategy we'll use is the following: we assume B is a regular language; then we mold B using operations that regular languages are closed under into a non-regular language C, which we can more easily prove is non-regular using the pumping lemma; and finally we reach a contradiction because if B were regular, we've only used operations that the class of regular languages are closed under, so the final resulting language C must have been regular.*

Assume for contradiction that B is regular. Then the complement of B, which we denote B^c, must also be regular. Note that B^c consists of all strings of the form $a^m b^n$ where $m = n$ as well as all strings of a's and b's that don't fit in the form $a^m b^n$ (e.g. abba). This means that if we intersect B^c with the language $D = \{a^m b^n \mid$ any value of $m, n\}$, we get $C = \{a^m b^m\}$. Note that D is a regular language, as a simple DFA/NFA can recognize strings of the form $a^m b^n$. Then we can write $C = B^c \cap D$. The right hand side must be regular since we've taken regular languages and applied operations that the class of regular languages are closed under. However, we know that C is not regular (this is a simple pumping lemma exercise), which is a contradiction.