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. $\Sigma$ is a finite alphabet,
3. $\delta : Q \times \Sigma \rightarrow \mathcal{P}(Q)$ is the transition function,
4. $q_0 \in Q$, is the start state, and
5. $F \subseteq Q$ is the set of accept states.

NFAs are similar to DFAs, except that NFAs are allowed to have $\epsilon$ 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 \( \epsilon \)-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.

Next, we look at an example with binary parallel addition.

**Example 2** Let \( \Sigma_3 \) be the alphabet of all columns of 0s and 1s of height three,
e.g. \( \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \). A string of symbols in \( \Sigma_3 \) gives three rows of 0s and 1s.
Consider each row to be a binary number and let

\[
T = \{ w \in \Sigma_3^* \mid \text{the bottom row of } w \text{ is the sum of the top two rows} \}.
\]

Show that \( T \) is regular.

**Solution 2** When we check whether the bottom row is the sum of the above
rows, we can go digit by digit and keep track of the bottom digit’s correctness,
as well as carries from addition. This is simplest to do by reading the string in
reverse. Since regular languages are closed under reversal, it suffices to show
that \( T^R \) is regular. So, we build a DFA with that reads from the rightmost
digit to leftmost digit.

Let \( q_0 \) denote the state of having no carry, \( q_1 \) be the state of having carry
1, and \( q_{\text{bad}} \) be a “dead” state that invalid symbols will transition to. \( q_0 \) will
be the the start state and also accept state.

![Diagram of DFA for Example 2](https://via.placeholder.com/150)

Since this DFA recognizes \( T^R \), \( T \) is regular. This approach can be used for simi-
lar problems like addition in other bases or multiplication by a given constant.
Pumping Lemma

The pumping lemma is used to prove that a given language $A$ is not regular. Note that we have to show that there is no FA $M$ such that $L(M) = A$. We use proof by contradiction relying on a property for regular languages given by the pumping lemma.

**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. $|xy| \leq p$

**Example 3** Show that $A_1$ containing strings of the form $0^n1^n0^n$ for $n \geq 0$ is non-regular.

**Solution 3** Assume for contradiction that $A_1$ is regular. Then $A_1$ must have a pumping length $p$. Consider the string $s = 0^p1^p0^p$. Note that $s \in A_1$ and $|s| = 2p + 1 \geq p$. For every partition $s = xyz$, 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.

**Example 4** Show that $A_2$ containing strings of the form $0^n1^m$ for $n \geq m$ is non-regular.

**Solution 4** 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 $|s| = 2p \geq p$. For every partition $s = xyz$, 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.

**Example 5** Show that

$$A = \{w \mid w \text{ has an equal number of } 0\text{s and } 1\text{s or } w \text{ has odd length}\}$$

is nonregular.
Solution 5 Assume for contradiction that A is regular. Consider B = $(\Sigma \Sigma)^*$, the language of all even-length strings. By closure under intersection, the language $C = A \cap B$ must be regular. C is exactly the language of strings with equal number of 0s and 1s, which we know to be nonregular, contradiction.

Recall that C can be shown to be nonregular with a similar argument. Let language D contain strings where 0s come before 1s and E contain strings in the form $0^n 1^n$. D can be recognized by a DFA, and E can be shown to be nonregular by the pumping lemma. Note that $C \cap D = E$. So if C was regular, E would have be regular too, contradiction.