Homework 3: Regular Expressions and Non-Regular Languages

Will Griffin

February 12, 2026

1. Regular Expressions vs Unix Regular Expressions

(a) Unix regular expressions have *quantifiers*: if α is a regular expression, $a^{\{m,n\}}$ is a regular expression that matches at least m and no more than n strings that match α. More formally, it matches all strings $w^{(1)} \cdots w^{(l)}$ where $m \leq l \leq n$, and for all such i such that $1 \leq i \leq l$, $w^{(i)}$ matches α. Prove that for any regular expression with quantifiers, there is an equivalent regular expression without quantifiers.

A regular expression with a quantifier is written as such: $a^{\{m,n\}}$. To write this without quantifiers, we must understand that a quantifier $\{m, n\}$ on a regular expression matches the preceeding expression i times, where $m \leq i \leq n$.

There are three cases for this expression:

Case 1 ($m = n = 0$): $\alpha^{\{0,0\}}$ means that α will be matched both at least and at most 0 times (or 0 times total), which means that this regular expression only matches the empty string. Therefore, $\alpha^{\{0,0\}}$ can be written as ε.

Case 2 ($m = 0, n > 0$): $\alpha^{\{0,n\}}$ means that α will be matched at least 0 times and at most n times. Since any string w can be written concatenated with the empty string (since $w = w\varepsilon = \varepsilon w$), the original regular expression can be written as $(\alpha \cup \varepsilon)^n$, as that will match either the expression α or the empty string ε n times.

Case 3 ($0 < m \leq n$): $\alpha^{\{m,n\}}$ means that α will be matched at least m times and at most n times. This can be broken up into two sections: matching α at least m times and matching the potential additional $n - m$ times. Matching α at least m times can be written as α^m, and matching the potential additional $n - m$ times can be written as $(\alpha \cup \varepsilon)^{n-m}$. Concatenating these regular expressions together gives us the original regular expression without a quantifier: $\alpha^m(\alpha \cup \varepsilon)^{n-m}$.

Therefore, any regular expression with a quantifier can be rewritten without a quantifier.

(b) Unix regular expressions have *backreferences*. Give an example of a Unix regular expression that uses backreferences to describe a non-regular language, and prove that this language is not regular. Please write a full proof.

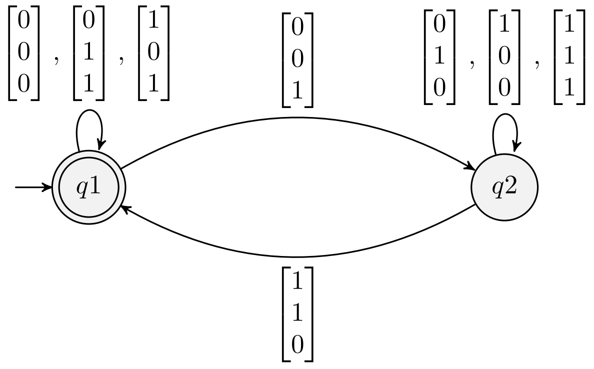
Unix regular expression: ([0-1]*)\1

This can be written as the non-regular language $L = \{ww \mid w \in \{\mathtt{0}, \mathtt{1}\}\}$. To prove this language is non-regular, assume for the sake of contradiction the language is regular. Let p be the pumping length given by the pumping lemma. Let a string $s = \mathtt{0}^p\mathtt{1}\mathtt{0}^p\mathtt{1}$. Since $s \in L$ and $|s| > p$, the pumping lemma lets s be written as $s = xyz$, where $y \neq \varepsilon$ and $|xy| \leq p$. Then, because $|xy| \leq p$, y only contains 0s. Let another string $s' = xyyz$. Since $|xy| \leq p$ and y only contains 0s, $z = 0^{p-k}10^p1$, where $|xy| = k$. Since we concatenate another y in s' compared to s, the first grouping of 0s has more 0s than the second grouping. Therefore, this string cannot be perfectly split in half, where the two substrings are equal. Therefore, $s' \notin L$. But, this contradicts the pumping lemma, where $s' \in L$. By proof of contradtiction, L is non-regular.

2. Binary Addition

(a) Show that the following is regular by writing an NFA for it:

$$B = \{w \in \Sigma_3^* \mid \text{the bottom row of } w \text{ is the sum of the top two rows}\}$$


$\begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\1\end{bmatrix}, \begin{bmatrix}1\\0\\1\end{bmatrix}$
$\begin{bmatrix}0\\0\\1\end{bmatrix}$
$\begin{bmatrix}0\\1\\0\end{bmatrix}, \begin{bmatrix}1\\0\\0\end{bmatrix}, \begin{bmatrix}1\\1\\1\end{bmatrix}$
$q1$
$q2$
$\begin{bmatrix}1\\1\\0\end{bmatrix}$

(b) Let $\Sigma = \{0, 1, +, =\}$, and prove that the following is not regular:

$$ADD = \{x = y + z \mid x, y, z \in \{0, 1\}^* \text{ and } x = y + z \text{ is true}\}$$

I will prove that ADD is not regular by proof by contradiction. Suppose ADD is regular. The pumping lemma states that all regular languages have a pumping length p.

Let a string $s = 1^p = 0 + 1^p$. The pumping lemma states that s can be written as $s = xyz$, where $|y| > 0$ and $|xy| \leq p$. The lemma also states that any string $s' = xy^iz \in ADD$ for any $i \geq 0$. Since $|xy| \leq p$, y contains only 1s.

Let a new string $s' = xz$, that is, the case where $i = 0$. This string can be also written as $1^{p-k} = 0 + 1^p$, where $k = |y|$. However, for this to be recognized by ADD, the equation must be true, and since the 1 terms on each side of the equation are not equal, this equation is clearly not true. Therefore, $s' \notin ADD$.

However, this contradicts the pumping lemma, which states that any $xy^iz \in ADD$ for any $i \geq 0$. Therefore, ADD is not a regular language by proof of contradiction.

3. Similar But Different

(a) Let $B = \{1^k w \mid w \in \{0, 1\}^* \text{ and } w \text{ contains at least } k \text{ 1s, for } k \geq 1\}$. Show that B is a regular language.

From this language, we get the pattern that every string in B must start with a 1 and contain at least 1 other 1 in the string. This can be written using the following regular expression:

$$1(0 \cup 1)^* 1(0 \cup 1)^*$$

This regular expression accepts any number of 1s followed by any number of 0s followed by at least one 1, followed by any number of 0s or 1s. This expression accepts all $1^k w$ where w contains at least k 1s. Since the minimum requirement of $k = 1$ is met by any string accepted by that regular expression, the language B is regular.

(b) Let $C = \{1^k w \mid w \in \{0,1\}^* \text{ and } w \text{ contains at most } k \texttt{ 1s, for } k \geq 1\}$. Prove that C is not a regular language.

Assume for the sake of contradiction that C is a regular language. Therefore, by the pumping lemma, we get a certain pumping length p.

Let a string $s = 1^p 1^p$. Since $s \in C$, by the pumping lemma, s can be written as $s = xyz$, where $|y| > 0$, $|xy| \leq p$, and $xy^i z \in C$ for all $i > 0$.

Let another string $s' = xz$, the case where $i = 0$. Since $|xy| \leq p$, xy must consist of 1s entirely from the first substring of 1s. Therefore, s' can also be written as $s' = 1^{p-j}1^p$, where $j = |y|$. Since $p - j < p$, $s' \notin C$. This contradicts the pumping lemma, which stated that $s' \in C$.

By contradiction, C is not a regular language.