Closure Characterization of Regular Languages

We saw characterizations of regular languages using a variety of machines and devices. Given a language \( A \subseteq \Sigma^* \):

(a) \( A \) is regular iff there exists a DFA \( M \) such that \( L(M) = A \).

(b) \( A \) is regular iff there exists an NFA \( N \) such that \( L(N) = A \).

(c) \( A \) is regular iff there exists an \( \epsilon \)-NFA \( N \) such that \( L(N) = A \).

(d) \( A \) is regular iff there exists a regular expression \( \alpha \) such that \( L[\alpha] = A \).

There is another characterization, different in character, that is sometimes used and that I mentioned quickly in class, in terms of closure properties. That is the characterization that I want to make precise here.

Let \( \mathcal{REG}(\Sigma) \) be the set of all regular languages over alphabet \( \Sigma \):

\[
\mathcal{REG}(\Sigma) = \{ A \subseteq \Sigma^* \mid A \text{ is regular} \}
\]

For the sake of keeping the notation light, I’m usually going to simply write \( \mathcal{REG} \), leaving \( \Sigma \) implicit. Since \( \mathcal{REG} \) is a set of languages, \( \mathcal{REG} \subseteq 2^{\Sigma^*} \).

To avoid some slight ambiguities later on, I’m going to use the term language class for set of languages. A language class is just a subset of \( 2^{\Sigma^*} \). \( \mathcal{REG} \) is a language class.

**Theorem 1.** \( \mathcal{REG} \) is the smallest language class \( \mathcal{C} \) satisfying the following:

(i) \( \emptyset \in \mathcal{C} \)

(ii) \( \{\epsilon\} \in \mathcal{C} \)

(iii) \( \{a\} \in \mathcal{C} \) (for every \( a \in \Sigma \))

(iv) for all \( A, B \): if \( A, B \in \mathcal{C} \) then \( A \cup B \in \mathcal{C} \)

(v) for all \( A, B \): if \( A, B \in \mathcal{C} \) then \( A \cdot B \in \mathcal{C} \)

(vi) for all \( A \): if \( A \in \mathcal{C} \) then \( A^* \in \mathcal{C} \)
What do I mean by the “smallest language class”? Smallest with respect to \( \subseteq \): if \( \mathcal{U} \) is any language class satisfying (i–vi), then \( \mathcal{REG} \subseteq \mathcal{U} \). (In other words, if \( \mathcal{U} \) is any language class satisfying (i–vi), then \( \mathcal{U} \) contains all the regular languages.)

To prove Theorem 1, we need to prove two things: that \( \mathcal{REG} \) satisfies (i–vi), and that \( \mathcal{REG} \) is the smallest language class that does. We prove those two facts.

**Lemma 2.** \( \mathcal{REG} \) satisfies (i–vi).

**Proof.** We already saw in class that if \( A \) and \( B \) are regular, then:

- \( A \cup B \) is regular (using a variant of the product construction for DFAs),
- \( A \cdot B \) is regular (by connecting end to end an \( \epsilon \)-NFA that accepts \( A \) and an \( \epsilon \)-NFA that accepts \( B \)), and
- \( A^* \) is regular (by looping the end and start states of an \( \epsilon \)-NFA for \( A \) via a new start state that is also final).

It remains to show that \( \emptyset, \{ \epsilon \}, \) and \( \{ a \} \) (for every \( a \in \Sigma \)) are regular. But there are obvious DFAs that accept these languages. This shows that the regular languages (i.e., \( \mathcal{REG} \)) satisfy (i–vi).

Slightly more interesting is showing that \( \mathcal{REG} \) is the smallest such language class. It turns out that the heavy lifting for this result has already been done in showing that for every regular languages, we can find a regular expression \( \alpha \) with \( L[\alpha] = A \).

Here is something that is rather immediate.

**Lemma 3.** Let \( \mathcal{U} \) be a language class satisfying (i–vi). For all regular expressions \( \alpha \), \( L[\alpha] \in \mathcal{U} \).

**Proof.** We’re going to prove this by induction over regular expressions. Define the height of a regular expression, \( h[\alpha] \), via the following recursive function:

\[
\begin{align*}
    h[0] &= 0 \\
    h[1] &= 0 \\
    h[a] &= 0 \\
    h[\alpha + \beta] &= 1 + \max(h[\alpha], h[\beta]) \\
    h[\alpha \cdot \beta] &= 1 + \max(h[\alpha], h[\beta]) \\
    h[\alpha^*] &= 1 + h[\alpha]
\end{align*}
\]

(The height of a regular expression is basically the height of the tree when writing the regular expression as an expression tree.)
The proof proceeds by induction on the height of regular expressions.

**Base case:** If \( h[\alpha] = 0 \), then \( \alpha \) is one of 0, 1, or \( a \) (for some \( a \in \Sigma \)). We consider each case:

- If \( \alpha \) is 0: then \( L[\alpha] = L[0] = \emptyset \), and by property (i), \( \emptyset \in \mathcal{U} \), as required.
- If \( \alpha \) is 1: then \( L[\alpha] = L[1] = \{ \epsilon \} \), and by property (ii), \( \{ \epsilon \} \in \mathcal{U} \), as required.
- If \( \alpha \) is \( a \) (with \( a \in \Sigma \)): then \( L[\alpha] = L[a] = \{ a \} \), and by property (iii), \( \{ a \} \in \mathcal{U} \), as required.

**Inductive step:** Assume that for all regular expressions \( \alpha \) with \( h[\alpha] \leq k \), \( L[\alpha] \in \mathcal{U} \).\(^\dagger\) (This is the induction hypothesis.)

We want to prove that for a regular expression \( \alpha \) with \( h[\alpha] = k + 1 \), \( L[\alpha] \in \mathcal{U} \).

Let \( \alpha \) be a regular expression with \( h[\alpha] = k + 1 \). Since \( h[\alpha] > 0 \), then \( \alpha \) is either of the form \( \alpha_1 + \alpha_2 \), \( \alpha_1 \cdot \alpha_2 \), or \( \alpha_1^* \). We consider each case:

- If \( \alpha \) is of the form \( \alpha_1 + \alpha_2 \): then \( L[\alpha] = L[\alpha_1] \cup L[\alpha_2] \). Since \( k + 1 = h[\alpha] = 1 + \max(h[\alpha_1], h[\alpha_2]) \), we have \( h[\alpha_1] \leq k \) and \( h[\alpha_2] \leq k \). By the induction hypothesis, we have \( L[\alpha_1] \in \mathcal{U} \) and \( L[\alpha_2] \in \mathcal{U} \). By property (iv), we get that \( L[\alpha_1] \cup L[\alpha_2] \in \mathcal{U} \), as required.
- If \( \alpha \) is of the form \( \alpha_1 \cdot \alpha_2 \): then \( L[\alpha] = L[\alpha_1] \cdot L[\alpha_2] \). Since \( k + 1 = h[\alpha] = 1 + \max(h[\alpha_1], h[\alpha_2]) \), we have \( h[\alpha_1] \leq k \) and \( h[\alpha_2] \leq k \). By the induction hypothesis, we have \( L[\alpha_1] \in \mathcal{U} \) and \( L[\alpha_2] \in \mathcal{U} \). By property (v), we get that \( L[\alpha_1] \cdot L[\alpha_2] \in \mathcal{U} \), as required.
- If \( \alpha \) is of the form \( \alpha_1^* \): then \( L[\alpha] = L[\alpha_1]^* \). Since \( k + 1 = h[\alpha] = 1 + h[\alpha_1] \), we have \( h[\alpha_1] \leq k \). By the induction hypothesis, we have \( L[\alpha_1] \in \mathcal{U} \). By property (vi), we get that \( L[\alpha_1]^* \in \mathcal{U} \), as required.

\( \square \)

Once we have Lemma 3, the following result follows easily:

**Lemma 4.** \( \mathcal{RE} \mathcal{G} \) is the smallest language class satisfying (i–vi).

**Proof.** Let \( \mathcal{U} \) be a language class satisfying (i–vi).

We need to show \( \mathcal{RE} \mathcal{G} \subseteq \mathcal{U} \), that is, for all \( A \in \mathcal{RE} \mathcal{G} \), \( A \in \mathcal{U} \).

Let \( A \in \mathcal{RE} \mathcal{G} \). In other words, \( A \) is regular. Therefore, there exists a regular expression \( \alpha_A \) such that \( L[\alpha_A] = A \). By Lemma 3, \( L[\alpha_A] \in \mathcal{U} \), that is, \( A \in \mathcal{U} \), as required.

\( \square \)

Lemmas 2 and 4 together give us a proof of Theorem 1.

\( \dagger \)This is a slightly different form of induction than the one we saw in class, in which at the inductive step we assume that the property we’re trying to prove holds for every natural number \( \leq k \) and try to prove the property for \( k+1 \). This is sometimes called strong induction, but the name is misleading, since it turns out to be exactly equivalent to the usual induction principle. (In fact, it is proved using the induction principle...)