THESE PAGES ARE PROVIDED PURELY FOR CLASS-ROOM USE BY
CS 3100 STUDENTS TAKING MY FALL 2011 OFFERING
PLEASE DISCARD AFTER THIS CLASS IS OVER. THANKS.

EXCERPTS FROM MY SPRINGER 2006 BOOK
"COMPUTATION ENGINEERING: APPLIED AUTOMATA THEORY AND LOGIC"

12

The 'Pumping' Lemma

Theorem 10.3 reiterates why regular languages are so called – their strings are "regular" in length. This fact can be taken advantage of in reasoning about languages. The specific approach taken is based on the fact that long strings meandering through finite-state structures cannot avoid revisiting states. Hence, if such a string goes from the start state to a final state, one can traverse the loop described by the re-visitation an arbitrary number of times, including zero times, and find other strings that also go from the start to the same final state. Expressed rigorously, this idea forms the basis of showing that certain languages are *not* regular, and takes the curious name of the "Pumping" Lemma.

The most common usage comes in the form of *incomplete Pumping Lemmas* or *one-way Pumping Lemmas* that help prove that certain languages are *not* regular. There are also *complete Pumping Lemmas* that can help *prove* that certain languages *are* regular. We now discuss one incomplete Pumping Lemma in depth and show many usages of the same. This will be followed by a brief discussion of one *complete* Pumping[1] Lemma.

12.1 Pumping Lemmas for Regular Languages

One *incomplete Pumping Lemma* for regular languages is as follows. If a language L is regular, then there exists a number n (typically equal in magnitude to the number of states of the minimal DFA D of L) such that for any string $w \in L$ exceeding n in length, w will have a loop somewhere in it. More specifically, when a DFA makes n

[1] Rumor has it that this is the most favorite lemma of a certain California governor.

state transitions, it must go through $n + 1$ states;[2] all of these states cannot be distinct (since there are altogether only n states). This causes the DFA to revisit at least one state, thus describing a path such as $s_1, s_2, \ldots, s_i, \ldots, s_i, \ldots, s_{n+1}$. Now, break w into three distinct pieces. Let x be the maximal prefix of w in which no states repeat ($s_1, \ldots, s_i$ in our example). Following x, we will have a segment of w that begins and ends at some specific state; this segment would form a loop, such as $s_i, \ldots, s_i$ in our example. Call this segment y. Now, the rest of w is considered to be the string z, which leads w to one of the accepting states $s \in F$ of D. It is then clear that the portion y can be repeated any number $k \geq 0$ of times in going to s, thus ensuring that strings of the form xy^kz are also in L. Using mathematical logic, and following Lamport's style, discussed on Page 79 in Chapter 5, we write:

$$
\begin{array}{l}
\mathit{Regular}(L) \Rightarrow \\
\qquad \exists n \in N : \\
\qquad \forall w \in L \ : |w| \geq n \\
\qquad\qquad\quad \Rightarrow \\
\qquad\qquad\quad \exists x, y, z \in \Sigma^* \ : \\
\qquad\qquad\quad \wedge\ w = xyz \\
\qquad\qquad\quad \wedge\ y \neq \varepsilon \\
\qquad\qquad\quad \wedge\ |xy| \leq n \\
\qquad\qquad\quad \wedge\ \forall k \geq 0 \ : \ xy^kz \in L.
\end{array}
$$

Illustration 12.1.1 *(Quantifier alternation)* The Pumping Lemma resembles the following example English assertion: "A zoo Z is interesting if *forall* giraffes g in Z whose right rear leg is more than n feet in length, there *exists* a reticulation patch on g's skin of exactly $\sqrt{n}$ feet circumference, such that within this reticulation patch, *forall* hair h, the color of h is brown."

To show that Z is *uninteresting*, we have to find one giraffe of height $\geq n$ such that for all patches, either the patch is not $\sqrt{n}$ feet in circumference, or (it is, and) there exists a non-brown hair in it.

To use the incomplete Pumping Lemma in proving that a language $L_{suspect}$ is *non*-regular, we proceed as follows. Assume $\mathit{Regular}(L_{suspect})$. Then, use the incomplete Pumping Lemma, obtaining as a consequence, the following formula C:[3]

[2] If the n moves are compared to n webs on the foot of a duck, then the duck must have $n + 1$ digits!

[3] Note that C, being a fully quantified formula, or *sentence*, is either true or false.

$$\begin{array}{ll}
\exists n \in N : & \\
\forall w \in L_{suspect} & : |w| \geq n \\
& \Rightarrow \\
& \exists x, y, z \in \Sigma^* \ : \\
& \ \wedge\ w = xyz \\
& \ \wedge\ y \neq \varepsilon \\
& \ \wedge\ |xy| \leq n \\
& \ \wedge\ \forall k \geq 0 \ : \ xy^k z \in L_{suspect}.
\end{array}$$

Now, we try to show that formula C is false (or that $\neg C$ is true). If we succeed in doing so, we can conclude using proof by contradiction that $\neg Regular(L_{suspect})$. What does showing $\neg C$ involve? Let $D = \neg C$. We can now write D as follows:

$$\begin{array}{ll}
\forall n \in N : & \\
\exists w \in L_{suspect} & : |w| \geq n \\
& \wedge \\
& \forall x, y, z \in \Sigma^* \ : \\
& \ \vee\ w \neq xyz \\
& \ \vee\ y = \varepsilon \\
& \ \vee\ |xy| > n \\
& \ \vee\ \exists k \geq 0 \ : \ xy^k z \notin L_{suspect}.
\end{array}$$

Now, our goal is to show that D is true (if we were to achieve this goal, we would have proved $\neg C$, or that $\neg Regular(L_{suspect})$), which is our original proof goal. To make D true, we must clearly satisfy the "bullet disjunction" embedded in it. That disjunction would be made true by making *any one of the following disjuncts* true *for every* $x,y,z \in \Sigma^*$:

1. pick x, y, z such that $w \neq xyz$,
2. pick $y = \varepsilon$,
3. pick x, y such that $|xy| > n$, or
4. find a $k \geq 0$ such that $xy^k z \notin L_{suspect}$.

Now, for many x, y, z, it will be possible to satisfy one of disjuncts 1 or 3. This is clear because we can quite easily find $xyz \neq w$, find $y = \varepsilon$, or find xy, such that $|xy| > n$. *So we don't even bother with these selections of* x, y, z *in the rest of this sequel.* What about x, y, z that falsify disjuncts 1 through 3? For that case, we *must* find a $k \geq 0$ such that $xy^k z \notin L_{suspect}$. That surviving case is now spelled out fully, below. This listing incorporates the fact that the first three disjuncts are false.

Pumping recipe: These steps below constitute the *Pumping recipe* we shall follow in attacking problems using the Pumping Lemma.

PR1: Consider x, y, z such that $w = xyz$ and $y \neq \varepsilon$ (thus falsifying disjuncts 1 and 2), and ensure that $|xy| \leq n$ (look for a loop within the first n moves in w), thus falsifying disjunct 3.

PR2: Find a $k \geq 0$ such that $xy^kz \notin L_{suspect}$ (thus satisfying disjunct 4).

One should, however, bear in mind the following frequently committed mistake, and avoid it:

If, instead of showing that formula C of page 206 is false, one ends up showing C, i.e, that C is true, then we cannot draw *any* conclusion about $L_{suspect}$. It could either be regular or non-regular! Refer to the discussion on page 74 around proving $5 = 5$.

We shall now illustrate these steps as well as related methods with several examples.

Illustration 12.1.2 Example: Consider $L = \{0^m10^m1 \mid m \geq 0\}$. To show L is not regular:

PR1:
1. Choose $w = 0^n10^n1$.
2. Choose $y \neq \epsilon$.
3. Choose $x, y,$ *such that* $|xy| \leq n$ – aha! Observe that y must contain a 0.

PR2:
1. Now, does there exist a $k \geq 0$ such that $xy^kz \notin L$?
2. Sure! For $k = 0$, we lose one 0, giving rise to a string of the form 0^m1^n where $m < n$. This satisfies the “D formula” associated with this example. Hence, L is not regular.

Illustration 12.1.3 Example: Consider $L = \{10^m10^m \mid m \geq 0\}$. To show L is not regular:

PR1:
1. Choose $w = 10^n10^n$.
2. Choose $y \neq \epsilon$.
3. Choose $x, y,$ *such that* $|xy| \leq n$.

PR2:
1. We have three choices for y:
   a) $y = 1$,

   b) $y = 10^l$ *for* $l < n$, or
   c) $y = 0^l$ *for* $0 < l < n$.
2. Does there exist a $k \geq 0$ such that $xy^kz \notin L$ for all these choices?
3. Sure!
   a) For $y = 1$, choose $k = 0$ (other choices work too; see Exercise 12.5).
   b) $y = 10^l$ *for* $l < n$, choose $k = 0$ (other choices work too).
   c) $y = 0^l$ *for* $0 < l < n$ also, choose $k = 0$ (other choices work too).
   d) In all these cases, the assertion $\exists\ k \geq 0$ such that $xy^kz \notin L$ is satisfied.
   e) Therefore, we get *full contradiction*, and hence L is not regular.

12.1.1 A stronger incomplete Pumping Lemma

There is a stronger version of the Pumping Lemma which allows strings "in the middle" to be pumped. We now state this Pumping Lemma semiformally, and illustrate its power on a simple example:

$$
\begin{array}{l}
\mathit{Regular}(L) \Rightarrow \\
\qquad \exists n \in N : \\
\qquad \forall w \in L\ : |w| \geq n \\
\qquad\qquad\qquad \Rightarrow \\
\qquad\qquad\qquad \forall x, y, z \in \Sigma^*\ : \\
\qquad\qquad\qquad\ \wedge\ w = xyz \\
\qquad\qquad\qquad\ \wedge\ |y| \geq n \\
\qquad\qquad\qquad\quad \exists u, v, w \in \Sigma^*\ : \\
\qquad\qquad\qquad\quad \wedge\ v \neq \varepsilon \\
\qquad\qquad\qquad\quad \wedge\ \forall k \geq 0\ :\ xuv^kwz \in L.
\end{array}
$$

As can be seen, the pumping can occur "in the middle."

Illustration 12.1.4 Consider the language L_{if} defined on page 212. By applying the ordinary Pumping Lemma, we cannot derive a contradiction starting from string ab^nc^n because the possibilities include $x = \varepsilon$, $y = a$, and $z = b^nc^n$, and by pumping y, we do not go outside the language. However, with the stronger Pumping Lemma, we can pick x, y, and z suitably, with $|y| \geq n$. Observe that by letting x be a, we can situate u, v, w in the b^nc^n region, obtaining a violation in all cases.

12.2 An adversarial argument

The Pumping Lemma provides a concrete setting to understand adversarial arguments. Consider proving, *directly using the Pumping Lemma*, that the language

$$L = \{0^i 1^j \mid i \neq j\}$$

is not regular. Here is how the proof goes as an adversarial argument. Suppose an adversary (Y) claims that this is a regular language. You (U) want to prove it is not. Here is how you can argue and win:

1. U: "OK if L is regular, you have a DFA D with you right?"
2. Y: "Yes."
3. U: "How many states in it?"
4. Y: "n".
5. U: "OK, describe to me the sequence of states that D goes through upon seeing the first n symbols of the string $0^n 1^{(n+n!)}$." Here, n is chosen to be the number of states in D. Since $n \neq (n + n!)$, this string surely must be in L. (The choice of $(n + n!)$ as the exponent of 1s is rather purposeful — and *very astute* on the part of U — as we shall see momentarily).
6. Y (Straight-faced): "It visits s_0, s_1, ..., *all of which are different from one another.*"
7. U: (Red-faced): "Lie! If there are n states in your DFA, then seeing n symbols, the DFA must have traversed a loop, and hence you must have listed two states that are the same. Don't you know that this follows from the pigeon-hole principle?"
8. Y: (Blue-faced): "OK, you are right, it is "$s_0, s_1, \ldots, s_i, s_{i+1}, s_{i+2}, \ldots, \ldots, s_i, s_j \ldots$." Notice that s_i is repeating in such a sequence.
9. U: "Aha! I'm going to call the pieces of the above sequence as follows."
   a) the piece that leads up to the loop, $s_0, s_1, \ldots, s_i$, will be called x,
   b) the piece that traverses the loop, $s_i, s_{i+1}, s_{i+2}, \ldots, s_i$, will be called y, and
   c) the piece that exits the loop and visits the final state, $s_i, s_j \ldots$, will be called z.

   "You may pick any such x, y, z and I'm going to confound you."
10. Y: "How?"
11. U: "Watch me!" (private thoughts of U now follow...)
    a) Since I have no idea what $|y|$ is, I must ensure that by pumping y, no matter what its length, I should be able to create a string of 0s equal in length to $(n + n!)$.

b) So, by pumping, if I can create an overall string $0^{(n+n!)}1^{(n+n!)}$, I would have created the desired contradiction.
c) The initial distribution of 0s along the path xyz is as follows:
   i. x has $|x|$ 0s,
   ii. y has $|y|$ 0s, and
   iii. z has $(n - |y| - |x|)$ 0s.
d) Hence, by pumping y k-times, for integral k, we must be able to attain $n + n! = |x| + k \times |y| + (n - |y| - |x|)$.
e) Simplifying, we should be able to satisfy $n! = (k - 1)|y|$. Since $|y| \leq n$, such a k exists!

12. U now begins his animated conversation: "See the above argument. I can now pump up the y of your string k times where $k = n!/|y| + 1$. Then you get a string $0^{n+n!}1^{n+n!}$ that is not in L. This path also exists in your DFA. So your DFA cannot be designed exactly for L— it also accepts illegal strings. Admit defeat!"
13. Y: Tries for an hour, furiously picking all possible x, y, z and goes back to step 8. For each such choice, U defeats Y[4] in the same fashion. Finally Y admits defeat.
14. U: "Thank you. Next victim please."

12.3 Closure Properties Ameliorate Pumping

The use of closure properties can simplify the application of the Pumping Lemma. However, caution is to be exercised to avoid unsound arguments. We now provide a few illustrations and exercises. First, let us rework Problem 12.1.3 as follows:

1. The reverse of $L = \{10^m10^m \mid m \geq 0\}$ is $L^{'} = \{0^m10^m1 \mid m \geq 0\}$
2. Now, $L^{'}$ was proved to be not regular in Problem 12.1.2
3. Since *reverse* preservers regularity, the original language L isn't regular either.

As a general approach, here is how we use regularity preserving operations to help make our arguments:

1. Suppose $M \cap L(0^*\ 1^*) = N$
2. Suppose we can show (thru Pumping Lemma) that N is not regular
3. *Then* we can conclude that M is not regular.

[4] I promise to make Y win in my next *two* books—and meanwhile, offer to put replacement pages on my web-page for the benefit of anyone wishing that Y trounce U in this very book!

Here is an **abuse** of the incomplete Pumping Lemma (from [111]). Consider the language

$$L_{if} = \{a^i b^j c^k \mid i \geq 0,\ j,k > 0,\ \textit{and if}\ i = 1\ \textit{then}\ j = k\}.$$

While this language is not regular, we can still show that the C formula of page 206 that results from the incomplete Pumping Lemma will be a tautology (“can pump k without causing any violations”). This is because

PR1: for *every* choice of w of the form $a^i b^j c^k$, and a way to split it into x, y, z that abide by the PR1 conditions,

PR2: we must find a $k \geq 0$ such that $xy^kz \notin L_{suspect}$. Basically, for any such x, y, z, there must always be a choice of y such that pumping causes us to stay in the language $L_{suspect}$, thus deriving no contradictions. Exercise 12.8 asks you to spell out this argument, and offers another attack on the same problem.

12.4 Complete Pumping Lemmas

There are many *complete* Pumping Lemmas of the form “$Regular(L)$ if and only if *conditions*,” i.e., a language is regular if and only if certain conditions hold. We present two popular versions, one due to Jaffe [65] and the other due to Stanat and Weiss [113]. Possible uses of these *complete* Pumping Lemmas include showing that certain languages *are* regular (we do not pursue such proofs of regularity in this book).

12.4.1 Jaffe’s complete Pumping Lemma

For a language L over a finite alphabet Σ, Jaffe’s Pumping Lemma is the following:

$$\begin{array}{l}
Regular(L) \Leftrightarrow \\
\qquad \exists k \in N : \\
\qquad \forall y \in \Sigma^* \ : |y| = k \\
\qquad\qquad\quad \Rightarrow \\
\qquad\qquad\quad \exists u, v, w \in \Sigma^* \ : \\
\qquad\qquad\quad \wedge\ y = uvw \\
\qquad\qquad\quad \wedge\ v \neq \varepsilon \\
\qquad\qquad\quad \wedge\ \forall z \in \Sigma^* \ : \\
\qquad\qquad\qquad\quad \forall i \in N \ : \ (yz \in L \ \Leftrightarrow\ uv^iwz \in L).
\end{array}$$

Notice that for a "long string" $y = uvw$ with a pump-able middle portion v, it is expressed that we can follow the original string y with an arbitrary z and stay within L, if and only if we can pump the middle and still follow it with that same z and stay within L. In [65], a proof of this Pumping Lemma is provided.

12.4.2 Stanat and Weiss' complete Pumping Lemma

For a language L over a finite alphabet Σ, Stanat and Weiss' Pumping Lemma is the following:

$$\begin{array}{l} \mathit{Regular}(L) \Leftrightarrow \\ \qquad \exists p \in N : \\ \qquad \forall x \in \Sigma^* \; : |x| \geq p \\ \qquad\qquad\qquad \Rightarrow \\ \qquad\qquad\qquad \exists u, v, w \in \Sigma^* \; : \\ \qquad\qquad\qquad \wedge\; x = uvw \\ \qquad\qquad\qquad \wedge\; v \neq \varepsilon \\ \qquad\qquad\qquad \wedge\; \forall r, t \in \Sigma^* \; : \\ \qquad\qquad\qquad\qquad \forall i \in N \; : \; (rut \in L \; \Leftrightarrow \; ruv^i t \in L). \end{array}$$

Notice that this Pumping Lemma does not require the pump-able string to be part of the prefix; an arbitrary string r can lead off, and an arbitrary tail t can follow. In [113], a proof of this Pumping Lemma using Jaffe's Pumping Lemma is provided.

Chapter Summary

We discuss the so-called Pumping Lemmas that characterize regular sets. We also discuss operations that *preserve* regularity; given one or more sets, these operations are guaranteed to deliver only regular sets. This chapter shows how one may exploit these facts to disprove that certain languages are *not* regular. For the sake of completeness, we also very briefly discuss the so-called *complete* Pumping Lemmas that actually help establish that certain languages *are regular*. While we do not utilize these complete Pumping Lemmas to carry out any proofs, the fact that such lemmas exist is important to know.

Exercises

12.1. Argue that L_{badd} is non-regular, where L_{badd} is almost similar to L_{add} of Exercise 10.6 except for what is shown below:

$$L_{badd} = \{a_0 a_1 \ldots a_{k-1} b_0 b_1 \ldots b_{k-1} c_0 c_1 \ldots c_{k-1} \;\mid\; \ldots \mathit{same} \ldots\}$$

12.2. Consider the example language of Section 12.2 again:

$$L = \{0^i 1^j \mid i \neq j\}.$$

Using closure properties, show that this set is not regular.

12.3. Prove that if L is not regular, then LL is also not regular.

12.4. $L_{0n1n} = \{0^n 1^n \mid n \geq 0\}$ is easily shown to be non-regular. Now, show

$$L_{eq} = \{x \mid x \in \{0,1\}^* \textit{ and } \#_0(x) = \#_1 x\}$$

is not regular. (Hint: intersection with `0* 1*`).

12.5. Show how to solve the problems presented in Illustration 12.1.2 and 12.1.3 by choosing a $k \neq 0$. Write out the complete proof using such k values.

12.6. What's wrong with this argument?

1. Suppose $M \cap L(0^*\ 1^*) = N$.
2. Suppose we can show (through Pumping Lemma) that M is not regular.
3. Conclude that N is not regular.

12.7. Among the assertions below, identify those that are true, and justify them. For those that are false, provide a counterexample.

1. The union of two non-regular sets is always non-regular.
2. The intersection of two non-regular sets is always non-regular.
3. A regular set can have a non-regular subset.
4. A regular set can have a non-regular superset.
5. Every regular set has a regular subset.
6. The star of a non-regular set can never be regular.
7. The prefix-closure of a non-regular set can never be regular.
8. The reverse of a non-regular set can never be regular.
9. The union of a regular and a non-regular set can be regular.
10. The union of a regular and a non-regular set can be non-regular.
11. The concatenation of a regular and a non-regular set can sometimes be regular.
12. There is a finite non-regular set.
13. It is possible to apply a homomorphism to turn a non-regular set into a regular set.

12.8. Consider the language L_{if} of page 12.3. Show that using the initial incomplete Pumping Lemma, we can pump, *i.e.*, prove the Pumping Lemma condition C to be true, thus being unable to conclude anything. Now, apply a closure property and use the incomplete Pumping Lemma to show that L_{if} is non-regular.

12.9. Using the stronger incomplete Pumping Lemma of Section 12.1.1, show that L_{if} is non-regular.

12.10. Prove the following languages to be non-regular:

1. $L_{sq} = \{0^{i^2} \mid i \geq 0\}$.
2. $L_{()} = \{x \mid x \in \{(,)\}^* \; and \; x \; is \; \mathrm{well-parenthesized}\}$.
3. The set of palindromes over $\{0,1\}^*$ is not regular.

The definition of well-parenthesized is as follows (see also page 224):

1. The number of (and) in x is the same.
2. In any prefix of x, the number of (is greater than or equal to the number of).