Lecture 7: The Pumping Lemma

Michael Engling

Department of Computer Science
Stevens Institute of Technology

CS 334 Automata and Computation
Fall 2015

OUTLINE: Non-Regular Languages

The Pigeonhole Principle

Contradiction? Don't Be Absurd!

The Language $L = \{0^n 1^n | n \geqslant 0\}$ is not regular

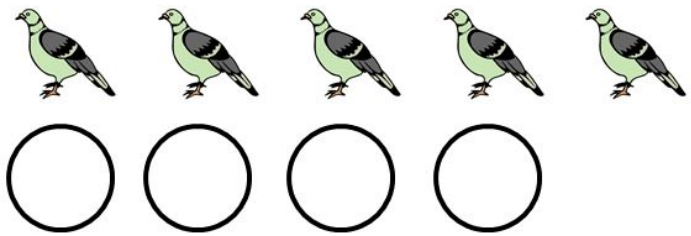
The Pumping Lemma

Another Non-Regular Language

Whence Non-Regularity?

The Pigeonhole Principle

10 pigeons, 9 pigeonholes



5 pigeons, 4 pigeonholes

The Pigeonhole Principle

The Pigeonhole Principle is simple: Given n pigeonholes and $n+1$ pigeons in those pigeonholes, there must be $(\exists)$ a pigeonhole with more than one pigeon in it.

Suppose each pigeon had its own pigeonhole. That would mean that $n+1 \leqslant n$. In turn, that implies $1 \leqslant 0$. That's false; a contradiction; reductio ad absurdum.

Our original assumption, that each pigeon is in its own pigeonhole, must be incorrect. Hence, there is a pigeonhole with more than one pigeon in it.

Does every pigeonhole have a pigeon in it? Maybe/maybe not. Are all the pigeons in one pigeonhole? Maybe/maybe not.

Contradiction? Don't Be Absurd!

That was perhaps the simplest example of "proof by contradiction" possible.

We start with an assumption, and then using arithmetic, definitions, rules of inference, etc. arrive at an absurdity, something we know to be false.

Since we are perfect arithmeticians and definers and logicians the error must lie in our original assumption. Whatever we assumed to be true ultimately leads to something we know to be false. Thus we must conclude that our assumption is also false.

We will now use "Reductio ad Absurdum" to show that a Non-Regular language exists.

The Language $L = \{0^n 1^n | n \geqslant 0\}$ is not regular

Suppose $\{L = 0^n 1^n | n \geqslant 0\}$ *is* regular. The L is recognized by a DFA.

This DFA is after all, a DFA, hence it has a finite number of states, say, p where $p \geqslant 1$.

Now, using this p, the number of states in our DFA, we know that $0^p 1^p$ is accepted by our DFA.

If computing the first half of this string (p symbols), we must have used $p + 1$ states:

$$q_0 - 0 - q_i - 0 - q_j - 0 - \cdots - 0 - q_{n-1} - 0 q_n - 1 \ldots$$

By the pigeonhole principle, two of those states must be the same. There are $p + 1$ names of states but only p actual states. We must have used one twice.

The Language $L = \{0^n 1^n | n \geqslant 0\}$ is not regular

$$q_0 - 0 - q_i - 0 - q_j - 0 - \cdots - 0 - q_{n-1} - 0 q_n - 1 \ldots$$

The duplicates might be anywhere. They might be q_0 and q_n in which case we're back where we started. They might be a stutterstep–perhaps q_j is repeated right after q_j. We don't know which state was repeated, but we know there is one that got repeated.

That repeated state forms a loop that comes back to itself. Maybe it includes every other state; maybe it just repeats itself like a hiccup, but from that state we deviated from a straight shot beginning to end and returned there.

What if we take that detour more than once?

The Language $L = \{0^n 1^n | n \geqslant 0\}$ is not regular

$$q_0 - 0 - q_i - 0 - q_j - 0 - \cdots - 0 - q_j - 0 q_n - 1 \ldots$$

What if we take that detour more than once?

$$q_0 - 0 - q_i - 0 - q_j - 0 - \cdots - 0 - q_j - 0 - \cdots - 0 - q_{n-1} - 0 q_n - 1 \ldots$$

The detour has some positive length–it computes some of our string. Call that length l. Now, if our DFA accepts $0^p 1^p$, it must also accept $0^{p+l} 1^p$ because a string that needs to take the loop twice. . . can.

If L is regular, then L is recognized by a DFA with $p \geqslant 1$ states. If that DFA accepts $0^p 1^p$ it also accepts $0^{p+l} 1^p$ for some $l > 0$. This latter string is *not* in L. This contradicts our assumption that L is recognized by a DFA.

Therefore, L is *not* recognized by a DFA.
$L = \{0^n 1^n | n \geqslant 0\}$ is not a regular language.

The Pumping Lemma

The idea we used here is the notion behind The Pumping Lemma (for regular languages).

We will prove languages are not regular by:

1. Assuming the language is regular, is recognized by a DFA with p states,
2. Cleverly choosing a (long, i.e. length p or more) string,
3. Using that string to show that some string *not* in the language is accepted by the DFA,
4. Concluding, by contradiction, that the language is *not* regular.

The only troublesome part is being clever enough to find a string to use the machine against itself.

The Pumping Lemma

This is the notion behind The Pumping Lemma.

If A is a regular language, then there is a number p (the pumping length) where if s is any string in A of length at least p, then s can be written as $s = alz$ satisfying the following conditions:

1. For each $i \geqslant 0$, $al^i z \in A$,
2. $|l| > 0$,
3. $|al| < p$

This suggests that any string long enough has an initial part, then a "loop" segment that can be pumped as many times as we like, and then a latter segment (which may be very long). It's the loop part, the l segment, that we will exploit to show languages are not regular.

Note:we may have $a = \varepsilon$ in which case the loop starts right up front, but in this construction we cannot have $z = \varepsilon$.

Another Non-Regular Language

Let A be the language of strings 0^k, where k is prime. A is not a regular language.

Consider the string 0^k where k is the smallest prime larger than p, the pumping length of A.

Now, $s = 0^k$ can be written as $s = alz$ as in the pumping lemma.

Let $k = |a| + |z|$, the length of the string without the loop part.

Consider the string $w = al^k z$. (We pump the loop k times.)

w, which must be in A, has length $|a| + |l|(|a| + |z|) + |z|$.

This is $(|a| + |z|)(1 + |l|)$. This number is clearly composite, i.e. not prime. Hence A is not regular.

One More Non-Regular Language

Let E be the language of strings $0^i 1^j$, where $i \geqslant j$.
E is not a regular language.

Consider the string $0^p 1^p \in E$.

Now, $s = 0^p 1^p$ can be written as $s = alz$ as in the pumping lemma.

Consider the string $w = al^0 z$.

This is an example of "pumping down". Instead of blowing up the loop, we pop it instead and remove symbols from our string. String w, which must be in E, now looks like $w = az$.

But by condition 3 of the pumping lemma, $|al| < p$, and thus $|al|$ consists only of 0s. In popping the loop, we have removed only 0s from s and have constructed a w with fewer 0s than 1s which must be in E. A contradiction.

Whence Non-Regularity?

What makes a language non-regular?

How can you tell a language *might* be non-regular?

What goes wrong?

Basically, our machines (DFAs & NFAs) have a bounded memory. They have n states, and are thus constrained to n possible recollections. They have no way to "think outside the box".

In the next chapters we will give our computing devices the ability to remember more than just what their individual states permit. Infinite memory may not be possible, but unbounded memory is within our grasp, and it will open new possibilities for us.