CS 480 **Computational Theory**

Regular Languages—*Pumping Words*

Handout *from A. K. Dewdney's "The New Turing Omnibus"*

Besides programming languages, there are other, more abstract languages studied by computer scientists. Their names hint mildly at their natures:

Regular languages
Context-free languages
Context-sensitive languages
Recursively enumerable languages

In the classification of abstract languages, it is frequently useful to know when a given language is *not* in a particular class. One technique for doing precisely this is based on the *pumping lemma*, first published in 1961 by Y. Bar-Hillel, M. Perles, and E. Shamir. The pumping lemma for regular languages tells us that if we select a sufficiently long word from a regular language and "pump" it—as many times as we like—we always get a new word in that language.

What makes such knowledge useful is that sometimes, in dealing with a word from a language we know little about, we may find that a particular word in it cannot be pumped in any way without getting a word *not* in the language. In such a case, we immediately know that the language was not regular.

To *pump* a word (or string) s means to select a particular subword, say y, from s and to replace y by yy in s. Symbolically, if

$$s = xyz$$

then the result of pumping s (at y) once is

$$s' = xyyz$$

and the result of pumping s twice is

$$s'' = xyyyz$$

and so on. Indeed, we can pump s "negatively" by eliminating y altogether from s.

It is possible to discover both what the pumping lemma says and how it works, simultaneously, by examining a portion of the state-transition diagram for the finite automaton A that accepts a particular regular language L.

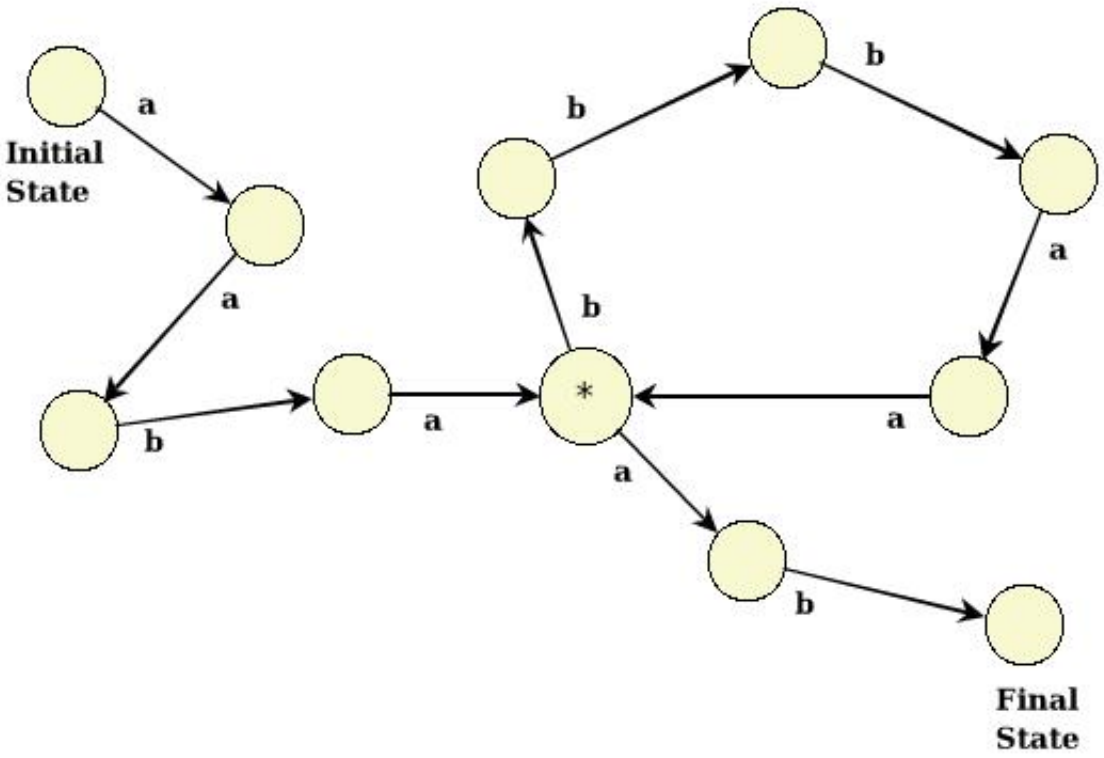

Figure 1: An input word leaves a trail

Here different states of the automaton are represented by circles, and transitions between states are represented by arrows labeled with the input symbol which causes that particular transition.

Each word in L corresponds to a trail through A's diagram. The trail leads from the initial state to one of A's final states. For example, the trail for the word $aababbbaaab$ is shown in the figure above. The trail is simply the sequence of transitions through which A must go in accepting the word.

The most interesting thing about the trail pictured above is that it contains a loop: having entered one of A's states, the trail returns to it later. This observation gives us an immediate idea for producing new words which A must accept: Simply repeat that portion of the word corresponding to the loop

$$aaba(bbbaa)ab$$

to get

$$aaba(bbbaa)(bbbaa)ab$$

In other words, $aababbbaabbbaaab$ must also be accepted by A and hence must be in L. By the same reasoning, we can go around the loop as many times as we like before going on to the final state. Each time, we get a new word in L.

Unfortunately, there is nothing to guarantee that the trail corresponding to an arbitrarily selected word in L has a loop! However, A has only a finite number of states, say n of them, and a word of length $\geq n$ which happens to be in L will certainly drive A through just such a loop: any word of length n or more must cause at least one state to be revisited. This simple observation is an example of [RMN: what well-known principle?]

One more observation will put us in a position to state the pumping lemma. We note that the subword

$$xy = \underbrace{aaba}_{x}\underbrace{bbbaa}_{y}$$

has total length $\leq n$ while the subword y has length > 0.

The Pumping Lemma for Regular Languages

If L is a regular language, then there is a constant p such that for each word s in L having length $\geq p$, there are words x, y, z such that

$$\begin{aligned} &s = xyz \\ &|xy| \leq p \\ &|y| > 0 \\ &xy^i z \in L \ (\forall i \geq 0) \end{aligned}$$

Here, y^i simply means i repetitions of the word y strung together in the fashion we are already familiar with.

It is [instructive] to use the pumping lemma to show that a particular language is *not* regular. For example, let L be the language consisting of all palindromes over the alphabet $\{a, b\}$. Such words are symmetric about their midpoints. For example,

$$abbababba$$

If L is regular, then the pumping lemma applies to L and there would be some constant p (unknown, in general) which we must use to measure the length of candidate words for the pumping lemma's application. The strategy to use here is to notice that no matter what value p might have, the word

$$s = a^p b a^p$$

must be in L according to the definition of palindromes. But, according to the pumping lemma, s can be written

$$s = xyz$$

so that, in particular, the length of xy, the initial part of s, is less than or equal to p. Thus, xy consists of nothing but a's, and y consists of at least one a. It now follows, by the pumping lemma, that the word

$$s = xy^2z = a^m b a^p$$

is also in L. Now a^m reflects the fact that some nonzero portion of the initial string a^p has been pumped, so that $m > p$. But in this case, $a^m b a^p$ cannot possibly be in L since it is *not* a palindrome! The only thing that can have gone wrong in all this reasoning is the initial assumption that L was regular. Evidently, it is not.