Limitations of regular languages

Informatics 2A: Lecture 7

Alex Simpson

School of Informatics
University of Edinburgh
als@inf.ed.ac.uk

1 October, 2013

Non-regular languages

We have hinted before that not all languages are regular. E.g.

- The language $\{a^n b^n \mid n \geq 0\}$.
- The language of all *well-matched* sequences of brackets (,). N.B. A sequence x is well-matched if it contains the same number of opening brackets '(' and closing brackets ')', and no initial subsequence y of x contains more ')' than '('.
- The language of all prefixes of well-matched sequences of brackets (,). A string x is in this language if no initial subsequence y of x contains more ')' than '('.

But how do we know these languages aren't regular?

And can we come up with a general technique for proving the non-regularity of languages?

The basic intuition: DFAs can't count!

Consider $L = \{a^n b^n \mid n \geq 0\}$. Just suppose, hypothetically, there *were* some DFA M with $\mathcal{L}(M) = L$.

Suppose furthermore that M had just processed a^n, and some continuation b^m was to follow.

Intuition: M would need to have *counted* the number of a's, in order to know how many b's to expect.

More precisely, let q_n denote the state of M after processing a^n. Then for any $m \neq n$, the states q_m, q_n must be different, since b^m takes us to an accepting state from q_m, but not from q_n.

In other words, M would need infinitely many states, one for each natural number. Contradiction!

Three clicker questions

For each of the following languages over $\{a, b\}$, decide whether they are regular or not.

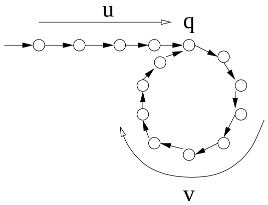
Press 1 for regular, 2 for non-regular.

1. Strings with an odd number of a's and an even number of b's.
2. Strings containing strictly more a's than b's.
3. Strings such that (no. of a's) * (no. of b's) $\equiv 6 \pmod{24}$

Put slightly differently...

Suppose there were some DFA M for $L = \{a^n b^n \mid n \geq 0\}$. Then M would have some finite number of states, say k.

Now consider what happens when we feed M with the string a^k. It passes through a sequence of $k+1$ states (including the initial state). So there *must* be some state q that's visited twice or more:



This means the string a^k can be decomposed as uvw, where

- u takes M from the initial state to q,
- v takes M once round the loop from q to q,
- w is whatever is left of a^k after uv.

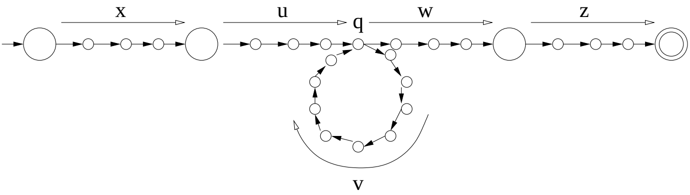
(Note that u and w might be ϵ, but v definitely isn't.)

More generally. . .

If L is *any* regular language, we can pick *some* corresponding DFA M, and it will have some number of states, say k.

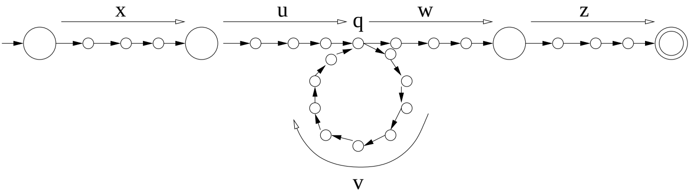
Not only must every string of length $\geq k$ cause a revisited state — so must every substring of length $\geq k$ *within* such a string.

Indeed, consider what happens when we run M on a string $xyz \in L$, where $|y| \geq k$. There must be at least one state q we visit twice in the course of processing y:



(There may be other 'revisited states' not indicated here.)

The idea of 'pumping'



So y can be decomposed as uvw, where

- xu takes M from the initial state to q,
- $v \neq \epsilon$ takes M once round the loop from q to q,
- wz takes M from q to an accepting state.

But now M will be oblivious to whether, or how many times, we go round the v-loop!

So we can 'pump in' as many copies of the substring v as we like, knowing that we'll still end in an accepting state.

The pumping lemma: official form

The pumping lemma basically summarizes what we've just said.

Pumping Lemma. Suppose L is a regular language. Then L has the following property.

> *(P) There exists $k \geq 0$ such that, for all strings x, y, z with $xyz \in L$ and $|y| \geq k$, there exist strings u, v, w such that $y = uvw$, $v \neq \epsilon$, and for every $i \geq 0$ we have $xuv^iwz \in L$.*

The pumping lemma: contrapositive form

Since we want to use the pumping lemma to show a language *isn't* regular, we usually apply it in the following equivalent but back-to-front form.

Suppose L is a language for which the following property holds:

> ($\neg P$) *For all* $k \geq 0$*, there exist strings* x, y, z *with* $xyz \in L$ *and* $|y| \geq k$ *such that, for every decomposition of* y *as* $y = uvw$ *where* $v \neq \epsilon$*, there is some* $i \geq 0$ *for which* $xuv^iwz \notin L$*.*

Then L is not a regular language.

N.B. The pumping lemma can only be used to show a language *isn't* regular. Showing L satisfies (P) doesn't prove L is regular!

To show that a language *is* regular, give some DFA or NFA or regular expression that defines it.

The pumping lemma: a user's guide

So to show some language L is not regular, it's enough to show that L satisfies $(\neg\mathrm{P})$.

Note that $(\neg\mathrm{P})$ is quite a complex statement: $\forall \cdots \exists \cdots \forall \cdots \exists \cdots$.

It's helpful to think in terms of how you would refute an opponent who claimed to have a DFA for L.

We'll look at a simple example first, then offer some advice on the general pattern of argument.

Example 1

Consider $L = \{a^n b^n \mid n \geq 0\}$.
We show that L satisfies $(\neg\mathrm{P})$.

Suppose $k \geq 0$.
(k is chosen by 'opponent' — we just have to cope.)

Consider the strings $x = \epsilon$, $y = a^k$, $z = b^k$. Note that $xyz \in L$ and $\mid y \mid \geq k$ as required.
(x, yz are cunningly chosen by 'us'.)

Suppose now we're given a decomposition of y as uvw with $v \neq \epsilon$.
(u, v, w chosen by 'opponent' — we have to cope.)

Let $i = 0$. (i chosen by 'us'.)
Then $uv^i w = uw = a^l$ for some $l < k$. So $xuv^i wz = a^l b^k \notin L$.
(And so we win!)

Thus L satisfies $(\neg\mathrm{P})$, so L isn't regular.

Use of pumping lemma: general pattern

- The opponent proposes a number $k \geq 0$.
  You don't get to choose k — you have to cope with what the opponent throws at you.
- You respond with a cunning choice of strings x, y, z, which might depend on k. These must satisfy $xyz \in L$ and $|y| \geq k$.
  Also, y should be chosen to 'disallow pumping' ...
- The opponent picks a decomposition of y as uvw with $v \neq \epsilon$.
  Again, you just have to cope with his choice.
- Finally, you have to choose i ($\neq 1$) such that $xuv^iwz \notin L$.
  Here i might depend on all the previous data.

Example 2

Consider $L = \{a^{n^2} \mid n \geq 0\}$.
We show that L satisfies ($\neg$P):

Suppose $k \geq 0$.
Let $x = a^{k^2-k}$, $y = a^k$, $z = \epsilon$, so $xyz = a^{k^2} \in L$.
Given any splitting of y as uvw with $v \neq \epsilon$, we have $1 \leq |v| \leq k$.
So taking $i = 2$, we have $xuv^2wz = a^n$ where $k^2 + 1 \leq n \leq k^2 + k$.
But there are no perfect squares between k^2 and $k^2 + 2k + 1$.
So n isn't a perfect square. Thus $xuv^2wz \notin L$.

Thus L satisfies ($\neg$P), so L isn't regular.

Reading and prospectus

Relevant reading: Kozen chapters 11, 12.

This concludes the part of the course on regular languages.

Next time, we start on the next level up in the Chomsky hierarchy: context-free languages.