Bosker Blog

Miscellaneous maundering

I hate the Pumping Lemma

2013-08-18 § 36 Comments

I hate the Pumping Lemma for regular languages. It's a complicated way to express an idea that is fundamentally very simple, and it isn't even a very good way to prove that a language is not regular.

Here it is, in all its awful majesty: for every regular language L, there exists a positive whole number *p* such that every string *w*∈L that has *p* characters or more can be broken down into three substrings *xyz*, where *y* is not the empty string and the total length of *xy* is at most *p*, and for every natural number *i* the string xy^iz is also in L.

Did you understand that statement? I'd be willing to bet you didn't, if you haven't seen it before. It has a ferociously intimidating logical structure, with no fewer than **five** alternating quantifiers: "for every L … there exists *p* … such that for every *w* … there exist *x*, *y* and *z* … such that for every *i* …". Beginning students of analysis are apt to struggle with the definition of continuity, because it takes a while to get used to having two nested quantifiers: for every epsilon > 0 there exists a delta > 0, etc. If two are a struggle, five is cruelty.

The real insult is that the actual underlying idea, and the proof, is shockingly simple. It is essentially the pigeonhole principle: the principle that if you put more than *n* pigeons into *n* holes then there must be a hole with more than one pigeon in. Take the regular language L, and express it as a deterministic finite automaton with *p* states. Any string in L determines a path through the automaton; so any string with *p* or more characters must visit the same state twice, forming a loop. This looped part can be repeated any arbitrary number of times to produce other strings in L.

If you understand the idea, it is easy to write down the incomprehensible formal statement of it. If you do not, the formal statement is not likely to lead you to enlightenment.

But the world is full of ways to express simple ideas in complicated ways. Why is this particular one foisted upon Computer Science students? Because it is used to show that some languages are not regular, which is important both in theory **and in practice (http://stackoverflow.com/questions/1732348/regex-match-open-tags-except-xhtml-self-contained-tags/1732454#1732454)**.

This leads to the second reason for my distaste for the Accursed Lemma: there are better ways to prove non-regularity, which are more powerful and give more insight into the nature of regularity, and which are more straightforward to state: they are omitted from the typical undergraduate curriculum, presumably because by the time the poor students have understood the hideous Pumping Lemma, there is no time left.

Here is my favourite: one might call it the Myhill-Nerode Theorem à la Brzozowski. Brzozowski invented the marvellous idea of *differentiating* formal languages. Let L be a language: not necessarily a regular language, just any set of strings; let *w* be a string, not necessarily in L. Then the derivative d/dw (L) is the language $\{ v \mid wv \in L \}$.

For example, if L is the set of English words then d/d'w' (L) is the words that start with 'w', with the 'w' removed. So it contains such strings as "ord" and "hy" and "ibble", and even "anker".

It's easy enough to see that any derivative of a regular language is again regular: taking a derivative just corresponds to changing the start state in a deterministic automaton. By the same argument, any regular language has only a finite number of different derivatives.

The really good thing is that it works the other way round as well: if a language has only a finite number of different derivatives, then it is regular; if it has infinitely many, it is not. Again the proof is easy: given a language L with a finite number of different derivatives, form a deterministic automaton with a state for each derivative. Put an edge labelled 'x' from A to B just when d/d'x' (A) = B. The start state is L itself, and any derivative that contains the empty string is marked as an accepting state.

So this condition completely characterises the regular languages, unlike the Pumping Lemma. There are **non-regular languages that are nevertheless pumpable (http://en.wikipedia.org/wiki/Pumping_lemma_for_regular_languages#Converse_of_lemma_not_tr** but they will still have infinitely many different derivatives.

In practice it is usually easy to use, as well. Take the classic example of $L = \{ a^i b^i \mid i \in \mathbb{N} \}$. Clearly $d/d(a^n)(L) = \{ a^i b^{n+i} \mid i \in \mathbb{N} \}$ for all n, and these are all different, so there are infinitely many derivatives and L cannot be regular.

Just to rub in how this is better than the Pumping Lemma, let's look at the example from Wikipedia of a language where the Pumping Lemma fails. It is $L = A \cup B \cup C$, over the alphabet {0,1,2,3}, where A is the strings that contain a doubled digit (00, 11, 22, 33), B is the strings that somewhere in them have the same digit twice with another digit in between (010, 020, 030, 101, 121, etc.); and C is the strings precisely 1/7 of whose digits are 3s.

We can express C as $\{ w \in \{0,1,2,3\}^* \mid n_{\{0,1,2\}}(w) = 6\, n_{\{3\}}(w) \}$. Here I am writing $n_S(w)$ to mean the number of characters of w that belong to S, where S is a subset of the alphabet.

Now for every natural number i, let's compute $d/d((0123)^i)(C)$: it is

$$\{ w \in \{0,1,2,3\}^* \mid n_{\{0,1,2\}}((0123)^i w) = 6\, n_{\{3\}}((0123)^i w) \}$$
$$= \{ w \in \{0,1,2,3\}^* \mid 3i + n_{\{0,1,2\}}(w) = 6i + 6\, n_{\{3\}}(w) \}$$
$$= \{ w \in \{0,1,2,3\}^* \mid n_{\{0,1,2\}}(w) = 3i + 6\, n_{\{3\}}(w) \}$$

These languages are all different, because for example for each i you can see that $d/d((0123)^i)(C)$ contains the string $(012)^i$, but none of the others do. What is more, none of the strings $(012)^i$ are in any $d/d((0123)^k)(A\cup B)$, therefore $d/d((0123)^i)(L)$ contains $(012)^j$ just when i=j, hence L has infinitely many derivatives and is not regular.

PS. I don't have anything against the pumping lemma for context-free languages.

PPS. This is not a complaint about my own education. I was taught finite automata by an enlightened lecturer who explained the simple idea behind the Pumping Lemma clearly and briefly, and had the good taste not to state it formally. It is driven rather by compassion for those less fortunate.

Update: For a dissenting view, see **this post by Lance Fortnow (http://blog.computationalcomplexity.org/2013/09/myhill-nerode-versus-pumping-lemma.html)**.

§ 36 Responses to *I hate the Pumping Lemma*

- ceyhuncanu (http://gravatar.com/ceyhuncanu) says:
  2013-08-18 at 2225 (http://bosker.wordpress.com/2013/08/18/i-hate-the-pumping-lemma/#comment-3074)
  "It is essentially the pigeonhole principle: the principle that if you put more than n pigeons into n holes then there must be a hole with more than one pigeon in."

  I guess there is a typo here. Number of pigeons should be more than number of holes to express the idea of **http://en.wikipedia.org/wiki/Pigeonhole_principle (http://en.wikipedia.org/wiki/Pigeonhole_principle)** .

  Reply (/2013/08/18/i-hate-the-pumping-lemma/?replytocom=3074#respond)
  - Robin Houston (http://bosker.wordpress.com/) says:
    2013-08-18 at 2227 (http://bosker.wordpress.com/2013/08/18/i-hate-the-pumping-lemma/#comment-3075)
    No, no typo. You might like to reread the sentence you quoted. :-)

    Reply (/2013/08/18/i-hate-the-pumping-lemma/?replytocom=3075#respond)
    - ceyhuncanu (http://gravatar.com/ceyhuncanu) says:
      2013-08-18 at 2237 (http://bosker.wordpress.com/2013/08/18/i-hate-the-pumping-lemma/#comment-3076)
      Yup, sorry. It seems I missed "_more than_" part. :-) I am having hard time understanding what I read these days.

      By the way, forgot to mention this is a real nice article, explaining such a hard-to-digest topic.

- bobt says:
  2013-08-19 at 0224 (http://bosker.wordpress.com/2013/08/18/i-hate-the-pumping-lemma/#comment-3078)

Can you please rewrite this article with a clear point? It's a complicated article to express an idea that is fundamentally unknown to readers.

Reply (/2013/08/18/i-hate-the-pumping-lemma/?replytocom=3078#respond)

- Atash says:
  2013-08-19 at 0309 (http://bosker.wordpress.com/2013/08/18/i-hate-the-pumping-lemma/#comment-3079)
  The first sentence of the article seemed pretty clear to me…

  Reply (/2013/08/18/i-hate-the-pumping-lemma/?replytocom=3079#respond)
- Brando Miranda says:
  2013-08-19 at 0350 (http://bosker.wordpress.com/2013/08/18/i-hate-the-pumping-lemma/#comment-3080)
  I hate the Pumping Lemma for regular languages. It's a complicated way to express an idea that is fundamentally very simple, and it isn't even a very good way to prove that a language is not regular.

  Thats what its about. Are you familiar with this topic?

  Reply (/2013/08/18/i-hate-the-pumping-lemma/?replytocom=3080#respond)

- Greg says:
  2013-08-19 at 0415 (http://bosker.wordpress.com/2013/08/18/i-hate-the-pumping-lemma/#comment-3081)
  "Clearly d/d(an)(L) = { aibn+i | i ∈ \mathbb{N} } for all n, and these are all different"

  Do you mean that there are an infinite number of derivatives because you can take d/d(a^1)(L), d/d(a^2)(L), … , d/d(a^n)(L), and for each n, the resulting derivative language is unique with respect to all other n?

  Whereas if L is regular, then all derivative functions will map to a finite set of languages?

  Just trying to make sure I understand correctly. Thank you for the interesting post.

  Reply (/2013/08/18/i-hate-the-pumping-lemma/?replytocom=3081#respond)
  - Robin Houston (http://bosker.wordpress.com/) says:
    2013-08-19 at 0709 (http://bosker.wordpress.com/2013/08/18/i-hate-the-pumping-lemma/#comment-3082)
    Yes, you understand perfectly. Thanks.

    Reply (/2013/08/18/i-hate-the-pumping-lemma/?replytocom=3082#respond)
- Stephan Wehner (@stephanwehner) (http://twitter.com/stephanwehner) says:
  2013-08-19 at 0751 (http://bosker.wordpress.com/2013/08/18/i-hate-the-pumping-lemma/#comment-3083)
  Are you saying that any language that can be shown to be non-regular using the Pumping Lemma can be shown to be non-regular without using the Pumping Lemma? There's a "plugin replacement" in each such proof?

  Of all the things one may not like one would think a mathematical fact would be the easiest to avoid.

  Cheers,

Stephan

Reply (/2013/08/18/i-hate-the-pumping-lemma/?replytocom=3083#respond)

- Robin Houston (http://bosker.wordpress.com/) says:
  2013-08-19 at 0801 (http://bosker.wordpress.com/2013/08/18/i-hate-the-pumping-lemma/#comment-3084)
  Yes. Every non-regular language has infinitely many derivatives. You never actually need to use the Pumping Lemma to show non-regularity.

  Reply (/2013/08/18/i-hate-the-pumping-lemma/?replytocom=3084#respond)

  - John Hughes says:
    2013-08-19 at 1353 (http://bosker.wordpress.com/2013/08/18/i-hate-the-pumping-lemma/#comment-3086)
    I'm not certain that this answers the question Stephan asked. Let me try to explain my doubts:

    Here are two properties of positive integers: (a) positive integers have unique factorizations. (b) For any positive integer n, the result of applying the predecessor function sufficiently often will be 0.

    The first of these can be used to prove various things about integers, as many elementary number theory texts demonstrate to excess. The second can be used to prove other things. You might ask "is everything provable using the first also provable using the second?" At some very basic level, the answer must be "yes", because when we say "Using the second," we really mean "Using the second and all the axioms for the integers," and using "all the axioms for the integers" we can prove the first, hence can prove anything derived from it.

    But that doesn't really capture the sense of the question. One way to think of proofs is via graph theory. At each moment in a proof, you're in some state of knowledge (call that a node), and there are tons of things you could say next. (call each of these an edge). If you pick the right sequence of edges, you reach a final node in which the knowledge you've established contains the conclusion of the theorem. So things proved by using unique factorization constitute certain paths in this large graph. The question is then something like "are there shorter paths to all (or most of) these things from the state in which you get to assume finite-predecessor-calls, with the paths not passing through the nodes that contain the knowledge of unique factorization?"

    To bring this back to languages, the fact that every non-regular language has some property (finitely many derivatives) and has some other property ("pumpingness") doesn't necessarily mean that there's a mechanism for replacing proofs involving the latter with "simpler" proofs involving the former, where by "simpler" I mean "not involving a hidden proof of the pumping lemma."

    I suspect that the answer to Stephan's question might well be "Yes," but I know little enough about the pumping lemma and its uses to have no confidence in my answer.

- Mark Mercer says:
  2013-08-20 at 0346 (http://bosker.wordpress.com/2013/08/18/i-hate-the-pumping-lemma/#comment-3092)

I think that the language {x: |x| is prime} should be easier to prove nonregular using the pumping lemma – Choose the long word to be a prime-length word and pump it up to a nonprime-length word. Generally if you have a language where membership only depends on the length of the input, you can skip the case analysis that normally comes with an application of the pumping lemma.

Reply (/2013/08/18/i-hate-the-pumping-lemma/?replytocom=3092#respond)

- eppstein says:
  2013-08-20 at 0734 (http://bosker.wordpress.com/2013/08/18/i-hate-the-pumping-lemma/#comment-3093)
  The pumping lemma proof is pretty easy, but here’s another easy proof using derivatives. For any integer x, let p(x) be the distance from x to the next prime number. Then, if p(x) ≠ p(y), then d/d(0^x) L ≠ d/d(0^y) L where L = {x : |x| is prime}. But p(x) can be arbitrarily large; for instance, p(n!+1) ≥ n. Therefore, L has infinitely many derivatives.

- Mark says:
  2013-08-20 at 1358 (http://bosker.wordpress.com/2013/08/18/i-hate-the-pumping-lemma/#comment-3096)
  eppstein: the claim ‘if p(x) ≠ p(y), then d/d(0^x) L ≠ d/d(0^y) L where L = {x : |x| is prime}. ‘ is almost certainly true, but it’s not obvious from the definition of primality. This has to be true for arbitrary prime x, y if you want to conclude immediately that you have infinitely many distinct derivatives.

- D. Eppstein (http://11011110.livejournal.com/) says:
  2013-08-20 at 2025 (http://bosker.wordpress.com/2013/08/18/i-hate-the-pumping-lemma/#comment-3098)
  You think it’s not obvious? I disagree. If p(x) ≠ p(y), we can assume without loss of generality that p(x) < p(y). Then x+p(x) is prime (the first prime after x), by definition of p, while y+p(x) is not prime (there is no prime until y+p(y), a larger number). Therefore the two languages d/d(0^x) L and d/d(0^y) L will differ on the input 0^{min(p(x),p(y)}.

- Mark says:
  2013-08-21 at 0022 (http://bosker.wordpress.com/2013/08/18/i-hate-the-pumping-lemma/#comment-3100)
  Ahh, I misunderstood what you were trying to do. So now you are finding an infinite set of pairs (x,y) that generate distinct derivatives, ok. But then how do you conclude that there are infinitely many derivatives? Consider the integral function f(x) = (-1)^x. There are infinitely many (x,y) pairs such that f(x) != f(y), but still the range of f is a constant 2.

- eppstein says:
  2013-08-21 at 0126 (http://bosker.wordpress.com/2013/08/18/i-hate-the-pumping-lemma/#comment-3101)
  Perhaps, in your zeal to find this argument non-obvious, you missed the sentence “But p(x) can be arbitrarily large.” Therefore there are infinitely many distinct values of p and infinitely many distinct derivatives.

- Mark says:
  2013-08-21 at 0309 (http://bosker.wordpress.com/2013/08/18/i-hate-the-pumping-lemma/#comment-3102)
  Yes, you are right. Sorry. Quite dense of me not to see that.

- Max Bucknell (http://gravatar.com/maxbucknell) says:
  2013-08-19 at 1305 (http://bosker.wordpress.com/2013/08/18/i-hate-the-pumping-lemma/#comment-3085)
  I understood the lemma, but not your explanation. But, I suppose I am an edge case, being a mathematics student who only dabbles in programming.

  Reply (/2013/08/18/i-hate-the-pumping-lemma/?replytocom=3085#respond)
- mac01021mac (http://gravatar.com/mac01021) says:
  2013-08-19 at 1531 (http://bosker.wordpress.com/2013/08/18/i-hate-the-pumping-lemma/#comment-3087)
  I'm not prepared to say I necessarily think your hatred is 100% misguided, but there are some point worth considering:

  1> Maybe there is some value in giving the students some practice thinking about logical propositions that are (structurally) nontrivial. With enough such practice, it might become second nature, in which case the students will have enhanced their abilities to reason.

  2> It might serve as an OK warm-up for the similar lemma about Context Free Languages.

  I expect that, if I were exposing the material to students, I would most likely begin from the informal appeal to the pigeonhole principle and the DFA with p states. But, once I was satisfied that everyone understood, I would still want to show how to express the theorem with rigor and formality and nested quantifiers, with all of which I would want the students to develop a great degree of comfort.

  Reply (/2013/08/18/i-hate-the-pumping-lemma/?replytocom=3087#respond)
- ulularem says:
  2013-08-19 at 1621 (http://bosker.wordpress.com/2013/08/18/i-hate-the-pumping-lemma/#comment-3088)
  "By the same argument, any regular language has only a finite number of different derivatives."

  From my naive reading, A* is a regular language with an infinite number of different derivatives (namely each A^n, for natural numbers n). For the obvious (canonical?) automaton of this language, taking these derivatives doesn't change the start state. Does that mean something is only a derivative if it changes the start state and removes a prefix from words in the language?

  Reply (/2013/08/18/i-hate-the-pumping-lemma/?replytocom=3088#respond)
  - Robin Houston (http://bosker.wordpress.com/) says:
    2013-08-19 at 1939 (http://bosker.wordpress.com/2013/08/18/i-hate-the-pumping-lemma/#comment-3089)
    Hi. It sounds as though you understand what's going on, but that you've got back-to-front what I meant by "different derivatives". If w∈A* then d/dw(A*) = A*, so there is only one derivative, namely A*.

    Or maybe you intended that A be a strict subset of the ambient alphabet, in which case there are two different derivatives, because d/dv(A*) = ∅ if v∉A*.

    In either case there are a finite number. As you say, this is the same as the number of states in the obvious automaton for the language.

    If you can say which part confused you, I'll try to make it clearer.

Reply (/2013/08/18/i-hate-the-pumping-lemma/?replytocom=3089#respond)

- ulularem says:
  2013-08-19 at 2050 (http://bosker.wordpress.com/2013/08/18/i-hate-the-pumping-lemma/#comment-3090)
  I'm not sure what would make it more clear: on re-reading, I see exactly what you're pointing out. Somehow I was counting derivatives as "distinct" if the strings used to generate them were distinct (and produced a non-empty language), but all your text points out that a derivative is a language, and hence A* = d/dv(A*), for all v.

  Thanks for the clarification.

- Barney says:
2013-08-19 at 2307 (http://bosker.wordpress.com/2013/08/18/i-hate-the-pumping-lemma/#comment-3091)
"If two are a struggle, five is cruelty."

Ha! That is why I switched from mathematics to computer science. In mathematics, you have to work through proofs that require many pages or even chapters to write. In computer science, if it takes more than half a page, you write a program to solve it for you, because that's usually simpler.

Reply (/2013/08/18/i-hate-the-pumping-lemma/?replytocom=3091#respond)

- Dave says:
2013-08-20 at 1248 (http://bosker.wordpress.com/2013/08/18/i-hate-the-pumping-lemma/#comment-3094)
Your last example looks to have some errors in it.

First, in your formula for d/d((0123)i)(C), I think you but the "+ 3i" on the wrong side of the equality. Shouldn't it be:

{ w∈{0,1,2,3}* | len(w∩{0,1,2}*) + 3i = 6 len(w∩{3}*)}

Second, I don't think it can be the case "that d/d((0123)^i)(C) contains the string (012)^i0123″. The number of 3's must be proportial to i (in fact, it must be 3*i), it cannot be constant. I think if you were to change this to "for each i, d/d((0123)^i)(C) contains the string (0123)^i or something similar, the rest of the argument goes through.

Reply (/2013/08/18/i-hate-the-pumping-lemma/?replytocom=3094#respond)

- Robin Houston (http://bosker.wordpress.com/) says:
  2013-08-20 at 2103 (http://bosker.wordpress.com/2013/08/18/i-hate-the-pumping-lemma/#comment-3099)
  Thank you, Dave. There were indeed some errors, though I don't think I quite agree about what they were.

  I was writing e.g. len(w∩{3}*) to mean the number of 3s in w, which doesn't really make sense. I've changed that notation to something ad hoc. (Is there a standard notation for this? Anyone?)

  The notational issue aside, I think the expression for d/d($(0123)^i$)(C) was correct. I have added a derivation, which I hope makes it clearer.

Yes, the “d/d($(0123)^i$)(C) contains the string $(012)^i$0123” part was wrong. It should have been $(012)^i$ without the extra 0123.

Thanks for prompting me to check this more carefully.

Reply (/2013/08/18/i-hate-the-pumping-lemma/?replytocom=3099#respond)

- Michael Tsai - Blog - The Pumping Lemma, The Pigeonhole Principle, and Differentiating Languages (http://mjtsai.com/blog/2013/08/21/i-hate-the-pumping-lemma/) says:
2013-08-21 at 2155 (http://bosker.wordpress.com/2013/08/18/i-hate-the-pumping-lemma/#comment-3104)
[…] Robin Houston (via @CompSciFact): […]

Reply (/2013/08/18/i-hate-the-pumping-lemma/?replytocom=3104#respond)

- Shali Jiang says:
2013-10-25 at 1253 (http://bosker.wordpress.com/2013/08/18/i-hate-the-pumping-lemma/#comment-3202)
I guess I’m one of the ‘less fortunate’ guys, I never thought of pumping lemma this way–it’s just wonderful! Thanks for sharing!

Reply (/2013/08/18/i-hate-the-pumping-lemma/?replytocom=3202#respond)

- Phil says:
2013-11-19 at 1804 (http://bosker.wordpress.com/2013/08/18/i-hate-the-pumping-lemma/#comment-3218)
“Take the regular language L, and express it as a deterministic finite automaton with p states. Any string in L determines a path through the automaton; so any string with p or more characters must visit the same state twice, forming a loop. This looped part can be repeated any arbitrary number of times to produce other strings in L.”

  Thank you, this statement did me some magic. Magic that no math/computer science professor could ever formulate into words. I really wish that they had more common sense in explaining material rather than regurgitating unnecessarilly confusing definitions from the textbook.

Reply (/2013/08/18/i-hate-the-pumping-lemma/?replytocom=3218#respond)

- emilio says:
2014-05-13 at 0442 (http://bosker.wordpress.com/2013/08/18/i-hate-the-pumping-lemma/#comment-3351)
The pumping lemma is also useful for proving decidability on the questions:
is the language accepted by a given finite automaton the empty language?
is the language accepted by a given finite automaton infinite?
are the languages accepted by two given finite automatons the same?

  how may the Myhill-Nerode thm be used to prove these?

Reply (/2013/08/18/i-hate-the-pumping-lemma/?replytocom=3351#respond)

  - Robin Houston (http://bosker.wordpress.com/) says:
2014-05-14 at 2213 (http://bosker.wordpress.com/2013/08/18/i-hate-the-pumping-lemma/#comment-3353)
Hi emilio. I’m not sure I entirely understand your question. If you can explain how you use the pumping lemma for these questions, I may be able to answer more usefully.

To find out if the language accepted by a given finite automaton is the empty language, it is sufficient to do a graph search (e.g. breadth-first search) from the start state and see if an accepting state is found. I'm not sure where the pumping lemma comes in?

One way to see if a finite automaton accepts an infinite language is to convert it to a regular expression and see if there are any stars in the regular expression. I can see how the pumping lemma could be used to prove that a finite automaton accepts an infinite language – by exhibiting an accepted string that has more characters than the automaton has states – but not how it could be used to decide the question.

I have no idea about the last one. This is a PSPACE-complete problem for non-deterministic finite automata or regular expressions, and I can't imagine how it's related to the pumping lemma.

Reply (/2013/08/18/i-hate-the-pumping-lemma/?replytocom=3353#respond)

- emilio says:
  2014-05-15 at 0240 (http://bosker.wordpress.com/2013/08/18/i-hate-the-pumping-lemma/#comment-3354)
  by using the pumping lemma you may prove:
  – the language accepted by an automaton with n states is not empty if and only if it accepts at least one word of length smaller than n
  – the language accepted by an automaton with n states is infinite if and only if it accepts at least one word of length l with n <= l < 2n

  you could try out every possible word of length 0 to n-1 to check if the language is empty, or n to 2n-1 to check if it is infinite. the number of words in each case is finite, and therefore you can always check this algorithmically.

  finally, checking if the language accepted by two given automatons M1 M2 is the same, can be done by constructing an automaton that accepts L(M1) – L(M2) and checking if it accepts only the empty language. such automaton can always be constructed algorithmically.

  of course these methods would not be efficient in practice.

- Robin Houston (http://bosker.wordpress.com/) says:
  2014-05-15 at 1159 (http://bosker.wordpress.com/2013/08/18/i-hate-the-pumping-lemma/#comment-3355)
  Ah, I see. Thanks for clarifying.

  Since these are problems that have better (and more efficient) solutions not using the pumping lemma, I don't really see why you would want to use Myhill-Nerode. Why not just use the better algorithms in the first place?

- emilio says:
  2014-05-15 at 1610 (http://bosker.wordpress.com/2013/08/18/i-hate-the-pumping-lemma/#comment-3356)
  well, from a pedagogical point of view, after learning finite automata, by means of a "shockingly simple" proof you get the pumping lemma and with it you get for free .) a tool for proving (many) languages are not regular, .) one liner proofs for decidability.

for these reasons it seems to me like a perfectly good choice over Myhill-Nerode for a normal course. Unless of course there were cool bonuses to it too.

- Lingadharini (http://thevicissitudeme.wordpress.com) says:
  2014-10-05 at 1112 (http://bosker.wordpress.com/2013/08/18/i-hate-the-pumping-lemma/#comment-3547)
  Reblogged this on **The Vicissitudious Me (http://thevicissitudeme.wordpress.com/2014/10/05/i-hate-the-pumping-lemma/)** and commented:
  Post I liked.

  Reply (/2013/08/18/i-hate-the-pumping-lemma/?replytocom=3547#respond)
- Lingadharini (http://thevicissitudeme.wordpress.com) says:
  2014-10-05 at 1239 (http://bosker.wordpress.com/2013/08/18/i-hate-the-pumping-lemma/#comment-3548)
  Nice post on 'Pumping Lemma for regular languages'. I used to wonder how people manage to understand it. And now I have successfully understood the concept of pumping lemma. But my basic question is, 'why state such a complex lemma to prove something is not regular? The language is not regular..we know. Then why take all these risks?'. Actually no one seems to have an answer to my question..!

  Reply (/2013/08/18/i-hate-the-pumping-lemma/?replytocom=3548#respond)