The Pumping Lemma

A Technique for Proving that Languages are Nonregular

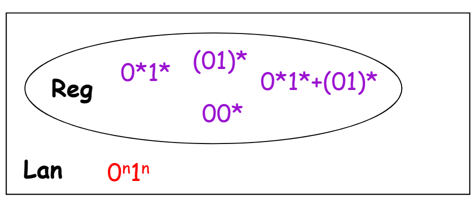
Friday, October 22, 2010
Reading: Sipser 1.4, Stoughton 3.13

CS235 Languages and Automata

Department of Computer Science
Wellesley College

Nonregular Languages: Overview

1. Not all languages are regular! As an example, we'll show the language $\{0^n1^n \mid n \text{ in Nat}\}$ is not regular.



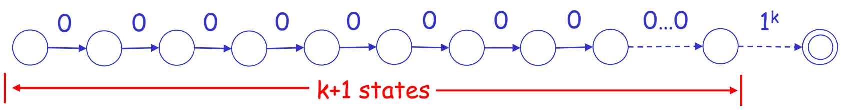
2. Generalize the technique for #1 by developing the **pumping lemma**.

3. Give examples of using the pumping lemma (sometimes in conjunction with closure properties of regular languages) to prove-by-contradiction that certain languages aren't regular.

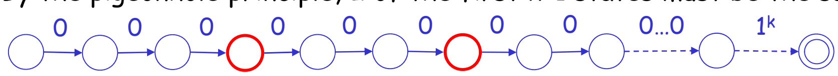
0^n1^n is Not a Regular Language

Proof by Contradiction: Suppose 0^n1^n ***is*** a regular language.
Then it is accepted by a DFA. Suppose the DFA has k states.

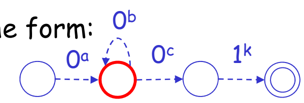
Now consider the labeled path for accepting the string 0^k1^k:



By the pigeonhole principle, 2 of the first k+1 states must be the same:



So the path has the form:



where $a + b + c = k$ and $b > 0$

This means the DFA also accepts strings $0^a0^{ib}0^c1^k$ for any $i \in Nat$.
But for $i \neq 1$, these strings do not have the form 0^n1^n for some n.
This contradicts the assumption that there is a DFA for 0^n1^n. ✗

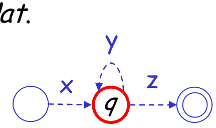
Generalizing the Technique: Intuition

Suppose L is an infinite regular language.

Any regular expression for L must contain a "nontrivial" *
(i.e., after weak simplification).

So it is accepted by an FA (and a DFA) with at least one loop.

Any sufficiently long string $s \in L$ must traverse some loop,
and so can be decomposed into xyz, where y is nonempty and
$xy^iz \in L$ for any $i \in Nat$.



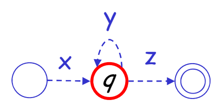
We say that the substring y of s can be **pumped**.

Generalizing the Technique: The Pumping Lemma

The Pumping Lemma
If L is a regular language, there is a number p (the pumping length) such that any string $s \in L$ with length $\geq p$ can be expressed as xyz, where:

1. $|y| > 0$
2. $|xy| \leq p$
3. $xy^iz \in L$ for each $i \in Nat$.

Proof sketch: Let p be the number of states in a DFA for L and q be the first repeated state in the path for s (which must exist by the pigeonhole principle). Use q to divide s into xyz.



Using the Pumping Lemma to Prove L Nonregular

The pumping lemma says every sufficiently long string in a regular language has a parse that can be pumped and still be in the language.

To prove a language nonregular, we just need to find **one counterexample string!**

Towards a contradiction, assume L is regular.

By the pumping lemma, there is a p such that all strings $s \in L$ with length $\geq p$ can be pumped.

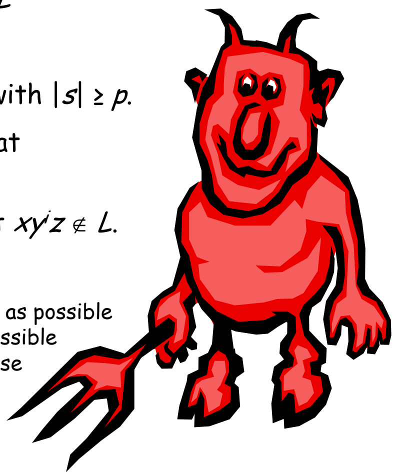
Find ***some*** string $s \in L$ with length $\geq p$ for which pumping is problematic. I.e., ***every*** decomposition of s into xyz with $|y| > 0$ and $|xy| \leq p$ leads to a string $xy^iz \notin L$ for ***some*** $i \in Nat$.

Therefore, the assumption that L is regular is false. ✗

Game vs. Demon

Using the pumping lemma to prove a language nonregular can be viewed as a game vs. a demon:

1. *You:* give the demon the language L
2. *Demon:* gives you p
3. *You:* give the demon string $s \in L$ with $|s| \geq p$.
4. *Demon:* divides s into xyz such that $|y| > 0$ and $|xy| \leq p$
5. *You:* give the demon an i such that $xy^iz \notin L$.



Notes:

- The demon will make your task as difficult as possible in step #4. *He* gets to chose the worst possible parse of s into xyz. *You* do **not** et to choose a parse that happens to be good for you.
- A clever choice of s in step #3 can tie the demon's hands in step #4, and make your life much easier in step #5.

$L_1 = \{0^n1^n \mid n \in Nat\}$ revisited

Viewed as game vs. a demon:

1. *You:* give the demon the language L_1
2. *Demon:* gives you p
3. *You:* give the demon a string $s \in L_1$ with $|s| \geq p$. E.g.:

   $s_1 = 0^{p/2}1^{p/2}$ (for simplicity, assume p is even)

   $s_2 = 0^p1^p$
4. *Demon:* divides s into xyz such that $|y| > 0$ and $|xy| \leq p$.
5. *You:* give the demon an i such that $xy^iz \notin L_1$

Moral: Since you get to pick string s, choose one that saves you work!

How to Write a Pumping Lemma Proof

Here's how to write a formal proof that L_1 is not regular.

Towards a contradiction, suppose L_1 were regular.

By the pumping lemma for regular languages, there is a pumping length p such that the string $s = 0^p1^p$ in L_1 would be pumpable --- i.e., parsable into xyz such that y is nonempty, $|xy| \leq p$, and $xy^iz \in L_1$ for all $i \in$ Nat.

s must be parsed as $x = 0^a$, $y = 0^b$, $z = 0^c1^p$, where a,b,c $\in$ Nat, a + b + c = p, and b $\geq$ 0.

But $xy^iz = 0^{a+bi+c}1^p = 0^{p+b(i-1)}1^p$, which $\notin L_1$ for any $i \neq 1$.
So L_1 cannot be regular.

You should write pumping lemma proofs on PS7 in this format!

L_2 = {w | w has equal # of 0s and 1s}

1. *You:* give the demon the language L_2
2. *Demon:* gives you p
3. *You:* give the demon a string $s \in L_2$ with $|s| \geq p$.
   Which ones below work?

   $s_1 = 0^{p/2}1^{p/2}$

   $s_2 = 0^p1^p$

   $s_3 = (01)^p$
4. *Demon:* divides s into xyz such that $|y| > 0$ and $|xy| \leq p$
5. *You:* give the demon an i such that $xy^iz \notin L_2$

Moral: not all strings s work! (But just need one.)

L_2 : A Simpler Approach using Closure Properties

Suppose L_2 is regular.

Then $L_2 \cap$ 0*1* is regular. Why?

So L_2 can't be regular. Why?

Moral: Closure properties of regular languages are helpful for proving languages nonregular!

Intuition: Regular Languages "Can't Count"

Intuitively, the pumping lemma says that regular languages (equivalently, finite automata) can't count arbitrarily high – they'll get confused beyond k = the number of states.

This is why L_1 and L_2 aren't regular:

$L_1 = \{0^n1^n \mid n \in \text{Nat}\}$

L_2 = {w | w has equal # of 0s and 1s}

But be careful! This intuition can sometimes lead you astray!

For example, the following languages ***are*** regular:

{w | w in {0,1}* and has equal # of 01s and 10s} (PS4)

$\{1^k y \mid y \text{ in } \{0,1\}^* \text{ and } y \text{ contains at least } k \text{ 1s, for } k \geq 1\}$ (PS7)

Pumping Down: $L_3 = \{0^i1^j \mid i > j\}$

1. *You:* give the demon the language L_3
2. *Demon:* gives you p
3. *You:* give the demon what string $s \in L_3$ with $|s| \geq p$?

4. *Demon:* divides s into xyz such that $|y| > 0$ and $|xy| \leq p$

5. *You:* give the demon an i such that $xy^iz \notin L_3$

Moral: Sometimes i needs to be 0. This is called "pumping down".

$L_4 = \{ww \mid w \in \{0,1\}^*\}$

1. *You:* give the demon the language L_4
2. *Demon:* gives you p
3. *You:* give the demon what string $s \in L_4$ with $|s| \geq p$?

4. *Demon:* divides s into xyz such that $|y| > 0$ and $|xy| \leq p$

5. *You:* give the demon an i such that $xy^iz \notin L_4$.

Moral: Again, choosing s carefully can save you lots of work!

$L_5 = \{1^{n^2} \mid n \geq 0\}$

1. *You:* give the demon the language L_5
2. *Demon:* gives you p
3. *You:* give the demon what string $s \in L_5$ with $|s| \geq p$?

4. *Demon:* divides s into xyz such that $|y| > 0$ and $|xy| \leq p$

5. *You:* give the demon an i such that $xy^iz \notin L_5$.

Moral: Arithmetic details matter!

Pumpable Languages

Pumpability

A language L is **pumpable** iff there is a number p (the pumping length) such that any string $s \in L$ with length $\geq p$ can be expressed as xyz, where:

1. $|y| > 0$
2. $|xy| \leq p$
3. $xy^iz \in L$ for each $i \in Nat$.

The pumping lemma says:

L is regular $\Rightarrow$ L is pumpable

Careful: the converse is **not** true!

L is pumpable $\not\Rightarrow$ L is regular (Sipser 1.54, PS7 Prob3)