Poetry

The Pumping Lemma

Any regular language L has a magic number p
And any long-enough word in L has the following property:
Amongst its first p symbols is a segment you can find
Whose repetition or omission leaves x amongst its kind.

So if you find a language L which fails this acid test,
And some long word you pump becomes distinct from all the
rest,
By contradiction you have shown that language L is not
A regular guy, resiliant to the damage you have wrought.

But if, upon the other hand, x stays within its L,
Then either L is regular, or else you chose not well.
For w is xyz, and y cannot be null,
And y must come before p symbols have been read in full.

As mathematical postscript, an addendum to the wise:
The basic proof we outlined here does certainly generalize.
So there is a pumping lemma for all languages context-free,
Although we do not have the same for those that are r.e.

-- Martin Cohn

MA/CSSE 474 Theory of Computation

More on showing L non-regular

Algorithms and Decision Procedures for Regular Languages

Questions on Pumping Theorem?

- Or anything else from Chapter 8?

Reminder: Start early on HW 7, which has a "grace day" until Wednesday at noon.

Recap: Using the Pumping Theorem Effectively

- To choose *w*:
  - Choose a *w* that is in the part of *L* that makes it not regular.
  - Choose a *w* that is only barely in *L*.
  - Choose a *w* with as homogeneous as possible an initial region of length at least *k*.
- To choose *q*:
  - Try letting *q* be either 0 or 2.
  - If that doesn't work, analyze *L* to see if there is some other specific value that will work.

Q1

Where we are so far

- To show a language L to be non-regular:
  - Myhill-Nerode theorem
    - Number of equivalence classes for $\approx_L$ is infinite
  - Pumpiing Theorem
  - Closure properties, in conjunction with languages already shown to be non-regular.

Using the Closure Properties to prove a language non-regular

The two most useful properties are closure under:

- Intersection
- Complement

Using the Closure Properties

$L = \{w \in \{\mathtt{a}, \mathtt{b}\}^*: \#_{\mathtt{a}}(w) = \#_{\mathtt{b}}(w)\}$

If L were regular, then:

$L' = L \cap$ ________

would also be regular. But it isn't.

$L = \{\mathtt{a}^i\mathtt{b}^j: i, j \geq 0 \text{ and } i \neq j\}$

Try to use the Pumping Theorem.

What would you choose for w?

$L = \{a^i b^j : i, j \geq 0 \text{ and } i \neq j\}$

An easier way:

If L is regular then so is $\neg L$. Is it?

$L = \{a^i b^j : i, j \geq 0 \text{ and } i \neq j\}$

An easier way:

If L is regular then so is $\neg L$. Is it?

$\neg L = A^nB^n \cup \{\text{out of order}\}$

If $\neg L$ is regular, then so is $L' = \neg L \cap \mathtt{a}^*\mathtt{b}^*$

$= \_\_\_\_\_\_\_\_\_\_\_\_$

$L = \{a^i b^j c^k$: $i, j, k \geq 0$ and (if $i = 1$ then $j = k$)$\}$

We will show that every string in L of length at least 1 is pumpable.

Does that imply that L is regular? We shall see!

Rewrite the final condition as: $(i \neq 1)$ or $(j = k)$

$L = \{a^i b^j c^k$: $i, j, k \geq 0$ and ($i \neq 1$ or $j = k$)$\}$

Every string in L of length at least 1 is pumpable:

•If $i = 0$ then: if $j \neq 0$, let y be `b`; otherwise, let y be `c`. Pump in or out. Then i will still be 0 and thus not equal to 1, so the resulting string is in L.

•If $i = 1$ then: let y be `a`. Pump in or out. Then i will no longer equal 1, so the resulting string is in L.

•If $i = 2$ then: let y be `aa`. Pump in or out. Then i cannot equal 1, so the resulting string is in L.

•If $i > 2$ then: let y be `a`. Pump out once or in any number of times. Then i cannot equal 1, so the resulting string is in L.

$L = \{a^i b^j c^k: i, j, k \geq 0 \text{ and } (i \neq 1 \text{ or } j = k)\}$

But the closure theorems help. If L is regular, then so is:

$L' = L \cap ab^*c^*$.

$L' = \{ab^k c^k : j, k \geq 0\}$

Can easily use Pumping Theorem to show that L' is not regular

$L = \{a^i b^j c^k: i, j, k \geq 0 \text{ and } (i \neq 1 \text{ or } j = k)\}$ An Alternative

If L is regular, then so is L^R:

$L^R = \{c^k b^j a^i : i, j, k \geq 0 \text{ and } (i \neq 1 \text{ or } j = k)\}$

Use Pumping to show that L' is not regular:

Is English Regular?

Is English finite?

In the event that the Purchaser defaults in the payment of any installment of purchase price, taxes, insurance, interest, or the annual charge described elsewhere herein, or shall default in the performance of any other obligations set forth in this Contract, the Seller may: at his option: (a) Declare immediately due and payable the entire unpaid balance of purchase price, with accrued interest, taxes, and annual charge, and demand full payment thereof, and enforce conveyance of the land by termination of the contract or according to the terms hereof, in which case the Purchaser shall also be liable to the Seller for reasonable attorney's fees for services rendered by any attorney on behalf of the Seller, or (b) sell said land and premises or any part thereof at public auction, in such manner, at such time and place, upon such terms and conditions, and upon such public notice as the Seller may deem best for the interest of all concerned, consisting of advertisement in a newspaper of general circulation in the county or city in which the security property is located at least once a week for Three (3) successive weeks or for such period as applicable law may require and, in case of default of any purchaser, to re-sell with such postponement of sale or resale and upon such public notice thereof as the Seller may determine, and upon compliance by the Purchaser with the terms of sale, and upon judicial approval as may be required by law, convey said land and premises in fee simple to and at the cost of the Purchaser, who shall not be liable to see to the application of the purchase money; and from the proceeds of the sale: First to pay all proper costs and charges, including but not limited to court costs, advertising expenses, auctioneer's allowance, the expenses, if any required to correct any irregularity in the title, premium for Seller's bond, auditor's fee, attorney's fee, and all other expenses of sale occurred in and about the protection and execution of this contract, and all moneys advanced for taxes, assessments, insurance, and with interest thereon as provided herein, and all taxes due upon said land and premises at time of sale, and to retain as compensation a commission of five percent (5%) on the amount of said sale or sales; SECOND, to pay the whole amount then remaining unpaid of the principal of said contract, and interest thereon to date of payment, whether the same shall be due or not, it being understood and agreed that upon such sale before maturity of the contract the balance thereof shall be immediately due and payable; THIRD, to pay liens of record against the security property according to their priority of lien and to the extent that funds remaining in the hands of the Seller are available; and LAST, to pay the remainder of said proceeds, if any, to the vendor, his heirs, personal representatives, successors or assigns upon the delivery and surrender to the vendee of possession of the land and premises, less costs and excess of obtaining possession.

Is English Regular?

- The rat ran.
- The rat that the cat saw ran.
- The rat that the cat that the dog chased saw ran.

Let:
A = {cat, rat, dog, bird, bug, pony}
V = {ran, saw, chased, flew, sang, frolicked}.

Let L = English $\cap$ {The A (that the A)* V^* V}.

L = {The A (that the A)^n V^n V, $n \geq 0$}.

Let w = The cat (that the rat)^k saw^k ran.

Functions from one Language to Another

Let *firstchars*(L) =
{w : $\exists y \in L$
($y = cx$,
$c \in \Sigma_L$,
$x \in \Sigma_L^*$, and
$w \in c^*$)}

Are the regular languages closed under *firstchars*?

L	*firstchars*(L)
$\varnothing$	
a*b*	
ca*cb*	

Defining Functions from one Language to Another

Let $chop(L)$ =
$\{w : \exists x \in L$
$(x = x_1 c x_2,$
$x_1 \in \Sigma_L^*,$
$x_2 \in \Sigma_L^*,$
$c \in \Sigma_L,$
$|x_1| = |x_2|$, and
$w = x_1 x_2)\}$

Recap: Give an English description of the relationship between chop(L) and L

Are the regular languages closed under *chop*?

L	*chop*(*L*)
∅	
a*b*	
a*db*	

Decision Procedures

A decision procedure is an algorithm whose result is a Boolean value. It must:

- Halt
- Be correct

Important decision procedures exist for regular languages:

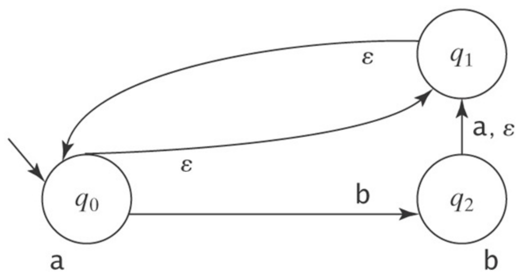
- Given an FSM *M* and a string *s*, does *M* accept *s*?

- Given a regular expression α and a string *w*, does α generate *w*?

Membership

We can answer the membership question by running an FSM.

But we must be careful if it's an NDFSM:



Membership

decideFSM(*M*: FSM, *w*: string) =
  If *ndfsmsimulate*(*M*, *w*) accepts then return *True*
                    else return *False*.

Recall that ndfsmsimulate takes epsilon-closure at every stage, so there is no danger of getting into an infinite loop.

decideregex(α: regular expression, *w*: string) =
  From α, use *regextofsm* to construct an FSM *M*
    such that $L(\alpha) = L(M)$.
  Return *decideFSM*(*M*, *w*).

Emptiness and Finiteness

- Given an FSM M, is $L(M)$ empty?
- Given an FSM M, is $L(M) = \Sigma_M*$?
- Given an FSM M, is $L(M)$ finite?
- Given an FSM M, is $L(M)$ infinite?
- Given two FSMs M_1 and M_2, are they equivalent?

Emptiness

- Given an FSM M, is $L(M)$ empty?
  - The graph analysis approach:
    1. Mark all states that are reachable via some path from the start state of M.
    2. If at least one marked state is an accepting state, return *False*. Else return *True*.
  - The simulation approach:
    1. Let M'= *ndfsmtodfsm*(M).
    2. For each string w in $\Sigma*$ such that $|w| < |K_{M'}|$ do:
       Run *decideFSM*(M', w).
    3. If M' accepts at least one such string, return *False*. Else return *True*.

Totality

- Given an FSM M, is $L(M) = \Sigma_M{}^*$?

Finiteness

- Given an FSM M, is $L(M)$ finite?
  - The graph analysis approach:
  - The simulation approach

Equivalence

- Given two FSMs M_1 and M_2, are they equivalent? In other words, is $L(M_1) = L(M_2)$? We can describe two different algorithms for answering this question.

Equivalence

- Given two FSMs M_1 and M_2, are they equivalent? In other words, is $L(M_1) = L(M_2)$?

equalFSMs$_1$(M_1: FSM, M_2: FSM) =

1. $M_1{'}$= *buildFSMcanonicalform*(M_1).
2. $M_2{'}$= *buildFSMcanonicalform*(M_2).
3. If $M_1{'}$ and $M_2{'}$ are equal, return *True*, else return *False*.

Equivalence

- Given two FSMs M_1 and M_2, are they equivalent? In other words, is $L(M_1) = L(M_2)$?

Observe that M_1 and M_2 are equivalent iff:

$$(L(M_1) - L(M_2)) \cup (L(M_2) - L(M_1)) = \varnothing.$$

equalFSMs$_2$(M_1: FSM, M_2: FSM) =

1. Construct M_A to accept $L(M_1) - L(M_2)$.
2. Construct M_B to accept $L(M_2) - L(M_1)$.
3. Construct M_C to accept $L(M_A) \cup L(M_B)$.
4. Return *emptyFSM*(M_C).