MA/CSSE 474

Theory of Computation

Pumping Theorem Examples

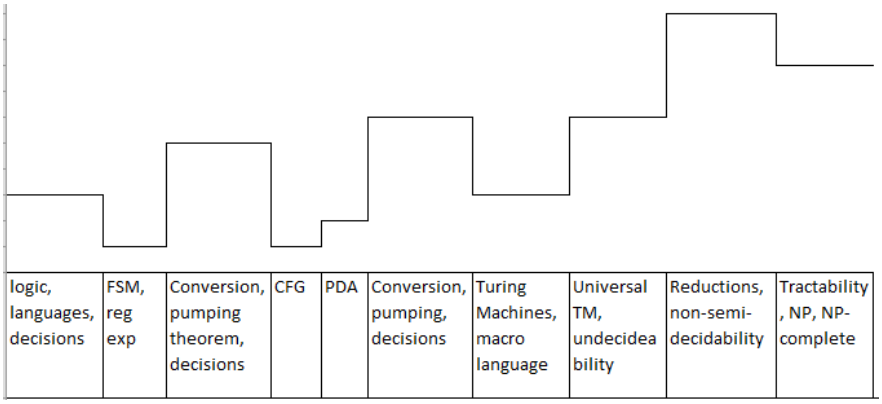
Decision Problems

The Any Key has a way of confusing the other keys.

Your Questions?

- Previous class days' material
- Reading Assignments
- HW 7 or 8 problems
- Anything else

474 Difficulty Timeline (imho, ymmv)



Using The Pumping Theorem to show that L is not Regular:

We use the contrapositive of the theorem:
If some long enough string in L is not "pumpable", then L is not regular.

What we need to show in order to show L non-regular:

$(\forall k \geq 1$
$(\exists$ a string $w \in L$
$(|w| \geq k$ *and*
$(\forall\ x, y, z\ ((w = xyz \wedge\ |xy| \leq k \wedge y \neq \varepsilon) \rightarrow$
$\exists\ q \geq 0\ (xy^q z \notin L))))))$

$\rightarrow$ L is not regular .

Before our next class meeting:
Be sure that you are convinced that this really is the contrapositive of the pumping theorem.

A way to think of it: adversary argument (following J.E. Hopcroft and J.D.Ullman)

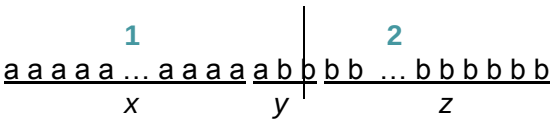
1. Choose the language L you want to prove non-regular.
2. The "adversary" picks k, the constant mentioned in the theorem.
3. We must be prepared for any positive integer to be picked, but once it is chosen, the adversary cannot change it.
4. We select a string $w \in L$ (whose length is at least k) that cannot be pumped".
5. The adversary breaks w into w=xyz, subject to constraints $|xy| \leq k$ *and* $y \neq \varepsilon$.
6. Our choice of w must take into account that any such x and y can be chosen.
7. All we must do is produce a single number $q \geq 0$ such that $xy^qz \notin L$.

Note carefully what we get to choose and what we do not get to choose.

Example: $\{a^n b^n : n \geq 0\}$ is not Regular

k is the number from the Pumping Theorem.
We don't get to choose it.

Choose *w* to be $a^{\lceil k/2 \rceil} b^{\lceil k/2 \rceil}$ ("long enough").



Adversary chooses *x*, *y*, *z* with the required properties:
$|xy| \leq k$,
$y \neq \varepsilon$,
We must show $\exists\, q \geq 0$ ($xy^qz \notin L$).

Three cases to consider:

- *y* entirely in region 1:
- *y* partly in region 1, partly in 2:
- *y* entirely in region 2:

For each case, we must find at least one value of q that takes xy^qz outside the language L.

The most common q values to use are q=0 and q=2.

A Complete Proof (read later)

We prove that $L = \{a^n b^n : n \geq 0\}$ is not regular

If L were regular, then there would exist some k such that any string w where $|w| \geq k$ must satisfy the conditions of the theorem. Let $w = a^{\lceil k/2 \rceil} b^{\lceil k/2 \rceil}$. Since $|w| \geq k$, w must satisfy the conditions of the pumping theorem. So, for some x, y, and z, $w = xyz$, $|xy| \leq k$, $y \neq \varepsilon$, and $\forall q \geq 0$, $xy^q z$ is in L. We show that no such x, y, and z exist. There are 3 cases for where y could occur: We divide w into two regions:

aaaaa…..aaaaaa| bbbbb…..bbbbbb
1 | 2

So y is in one of the following :

- (1): $y = a^p$ for some p. Since $y \neq \varepsilon$, p must be greater than 0. Let $q = 2$. The resulting string is $a^{k+p}b^k$. But this string is not in L, since it has more a's than b's.
- (2): $y = b^p$ for some p. Since $y \neq \varepsilon$, p must be greater than 0. Let $q = 2$. The resulting string is $a^k b^{k+p}$. But this string is not in L, since it has more b's than a's.
- (1, 2): $y = a^p b^r$ for some non-zero p and r. Let $q = 2$. The resulting string will have interleaved a's and b's, and so is not in L.

There exists one long string in L for which no pumpable x, y, z exist. So L is not regular.

What You Should Write (read these details later)

We prove that $L = \{a^n b^n : n \geq 0\}$ is not regular

Let $w = a^{\lceil k/2 \rceil} b^{\lceil k/2 \rceil}$. (If not completely obvious, as in this case, show that w is in fact in L.)

aaaaa…..aaaaaa| bbbbb…..bbbbbb
1 | 2

There *are three possibilities for y*:

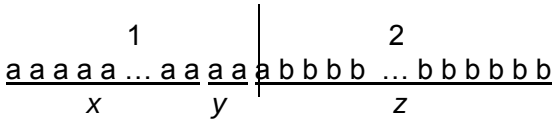
- (1): $y = a^p$ for some p. Since $y \neq \varepsilon$, p must be greater than 0. Let $q = 2$. The resulting string is $a^{k+p}b^k$. But this string is not in L, since it has more a's than b's. .
- (2): $y = b^p$ for some p. Since $y \neq \varepsilon$, p must be greater than 0. Let $q = 2$. The resulting string is $a^k b^{k+p}$. But this string is not in L, since it has more b's than a's.
- (1, 2): $y = a^p b^r$ for some non-zero p and r. Let $q = 2$. The resulting string will have interleaved a's and b's, and so is not in L.

Thus L is not regular.

A better choice for w

Second try. A choice of w that makes it easier:

Choose w to be $a^k b^k$
(We get to choose any w whose length is *at least* k).



We show that there is no x, y, z with the required properties:
$|xy| \leq k$,
$y \neq \varepsilon$,
$\forall\ q \geq 0$ ($xy^q z$ is in L).

Since $|xy| \leq k$, y must be in region 1. So $y = a^p$ for some $p \geq 1$.
Let q = 2, producing:

$$a^{k+p} b^k$$

which $\notin L$, since it has more a's than b's.

We only have to find **one** q that takes us outside of L.

Recap: Using the Pumping Theorem

If L is regular, then every "long" string in L is pumpable.

To show that L is not regular, we find one string that isn't.

To use the Pumping Theorem to show that a language L is not regular, we must:

1. Choose a string w where $|w| \geq k$. Since we do not know what k is, we must describe w in terms of k.
2. Divide the possibilities for y into a set of equivalence classes that can be considered together.
3. For each such class of possible y values where $|xy| \leq k$ and $y \neq \varepsilon$:
   Choose a value for q such that $xy^q z$ is not in L.

Some practice examples (do them with one or two other students)

- Bal = {$w \in$ {), (}* : the parens are balanced}
- PalEven = {ww^R : $w \in$ {a, b}*}
- {$w \in$ {a, b}* : $\#_a(w) = \#_b(w)$} Hint: Use closure
- {aba^nb^n : $n \geq 0$} Hint: Use closure

Decision Procedures

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

- Eventuallly halt, no matter what its input
- Be correct

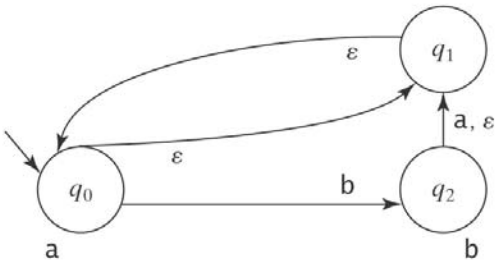
Important decision procedures exist for regular languages:

- 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*(α) = *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 Σ^* 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 $L(M) = \Sigma_M^*$?