Decision Properties of Regular Languages

The Pumping Lemma

Apology

- Office Hours Yesterday
- Last week's homework

Announcement

- RIT Career Fair
  - Thursday, Oct 3rd 1pm – 7pm
  - Friday, Oct 4th 9am – 5pm (interviews)
- Clark Gym
- www.rit.edu/co-op/careers

Announcement

- CS Student Meeting
- Wednesday, Oct 2nd
- 5-6pm
- 08-1250
- Followed by free pizza in Building 10 (outside of ICLs)

Homework

- Homework #3 Due Today
- Homework #4
  - 5.16d,e
  - 5.20b,e (Use the Pumping Lemma)
    - $n_0(x)$ = number of 0's in x
    - $n_1(x)$ = number of 1's in x
  - Given the RE (a*(a + b + abb)*)*
    - Find a FA with minimal number of states that accepts the language described.
    - Go through entire process: NDFA-Λ -> NDFA -> FA -> MFA

Before We Start

- Any questions?

Decision Properties

- Given regular languages, specified in any one of the four means, can we develop algorithms that answer the following questions:
  1. Is the language empty?
  2. Is the language finite?
  3. Is a given string in the language?
  4. Given 2 languages, are there strings that are in both?
  5. Is the language a subset of another regular language?
  6. Is the language the same as another regular language?

Non-regular languages

- And then there's the question:
  - Is there a language L that is not regular?

Additional tools

- Minimal finite automata
  - Create a finite automata with the minimum number of possible states
- Pumping Lemma
  - Defines necessary properties for strings in a regular language
  - Can be used to show that languages are not regular
  - Will consider next week

The Pumping Lemma

- The pumping lemma formalizes the idea that if a string from a RL is long enough, eventually at least one state on its FA will be have to be repeated on the path that accepts the string.
  - Implies that there is a Kleene star in there somewhere!
- Continually looping on this state will produce an infinite number of strings in the language

The Pumping Lemma

- Statement of the pumping lemma
  - Let L be a regular language.
  - Then there exists a constant n (which varies for different languages), such that for every string $x \in L$ with $|x| \geq n$, x can be expressed as x = uvw such that:
    1. $|v| > 0$
    2. $|uv| \leq n$
    3. For all $k \geq 0$, the string uv^kw is also in L.

The Pumping Lemma

- What this means
  - For a long enough string x in L:
    - We can express x as the concatenation of three smaller strings
    - The middle string can be "pumped" (repeated) any number of times (including 0 = deleting) and the resulting string will be in L.

Pumping Lemma

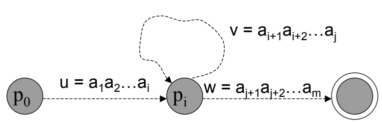
- Proof of the pumping lemma
  - Since L is regular, there is a FA $M=(Q,\Sigma,q_0,A,\delta)$ that accepts L.
    - Assume M has n states.
  - Consider a string x with $|x| = m \geq n$.
    - Express $x = a_1a_2a_3 \ldots a_m$ where each $a_i \in \Sigma$.
  - Define p_i to be the state M is in after reading i characters:
    - $p_i = \delta^* (q_0, a_1a_2\ldots a_i)$
    - $p_0 = q_0$

Pumping Lemma

- Proof of the pumping lemma
  - Since $|x| \geq n$, and we only have n states, one state on it's path must visited more than once.
    - There exists integers i and j, $0 \leq i < j \leq n$ such that $p_i = p_j$
    - Then x = uvw
      - $u = a_1a_2\ldots a_i$
      - $v = a_{i+1}a_{i+2}\ldots a_j$
      - $w = a_{j+1}a_{j+2}\ldots a_m$

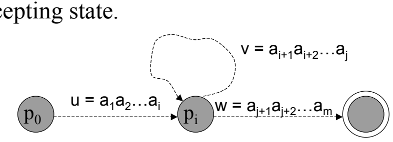
Pumping Lemma

- Proof of pumping lemma
  - Then x = uvw
    - $u = a_1a_2\ldots a_i$
    - $v = a_{i+1}a_{i+2}\ldots a_j$
    - $w = a_{j+1}a_{j+2}\ldots a_m$



Pumping Lemma

- Proof of pumping lemma
  - You can loop (pump) on the v loop 0 or more times and there will still be a path to the accepting state.



Pumping Lemma

- So what good is the pumping lemma?
- Let's revisit the question:
  - Given a specification of a regular language, is that language finite?

Pumping Lemma

- Test for finiteness
  - First stab
    - The pumping lemma tells us that if there is a string x with length greater than the number of states accepted by an FA, M, then L(M) is infinite.
    - Let's test all strings of length >= number of states.
    - Will give us a "yes", L(M) is finite but
    - For L(M) infinite, the algorithm will never stop.

Pumping Lemma

- Test for infiniteness
  - Given an FA M, L(M) is infinite if there is a string x accepted by M such that
    - $n \le |x| \le 2n$
    - n = number of states in M
  - Let's prove using the Pumping Lemma

Pumping Lemma

- Using the pumping lemma to prove things
  - Always involves proof by contradiction
    - If X then Y
    - Assume Y is false
    - Arrive at a contradiction to a known true fact (including X)
    - Must conclude that our assumption is false thus Y is true.

Pumping Lemma

- Test for infiniteness
  - Given an FA M, L(M) is infinite if there is a string x accepted by M such that
    - $n \le |x| \le 2n$
    - n = number of states in M
  - Assume that no such x exists.

Pumping Lemma

- Test for infiniteness
  - Since L(M) is infinite, there is a string that have length of at least 2n. Let z be the smallest such string.
  - By the pumping lemma
    - $z = uvw$
    - $|v| > 0$
    - $|uv| \le n$

Pumping Lemma

- Test for infiniteness
  - Also by the pumping lemma
    - $uv^0w = uw \in L(M)$
  - Since $|z| = |uvw| \ge 2n$ and $|v| \ge 0$ then
    - $|uw| > 2n$
  - Since $|z| = |uvw| \ge 2n$ and $|v| \le n$ then
    - $|uw| \ge n$
  - So either
    - $n \le |uw| < 2n$ This contradicts our orig. assumption
    - $2n > |uw| > |z|$ This contradicts z being smallest

Pumping Lemma

- Test for infiniteness
  - We came to a contradiction
  - Thus our original assumption that there is no x such that $n \le |x| \le 2n$ is false.
  - Thus we proved that there <u>is</u> such an x.

Pumping Lemma

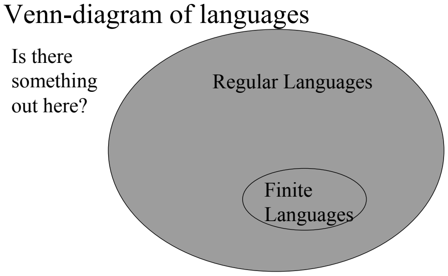
- Test for infiniteness
  - How does this help us?
  - Algorithm for testing if L(M) is infinite.
    - Systematically generate all strings of length between n and 2n where n is the number of states of M
    - Test each string generated
    - If at least 1 is accepted, then L(M) is infinite
    - Otherwise L(M) is finite.

Pumping Lemma

- Questions

Non-regular languages

- Venn-diagram of languages



Pumping lemma

- The real strength of the pumping lemma is proving that languages are not regular
  - Proof by contradiction
    - Assume that the language to be tested is regular
    - Use the pumping lemma to come to a contradiction
    - Original assumption about the language being regular is false
- You cannot prove a language to be regular using the Pumping Lemma!!!!

Pumping lemma

- The Pumping Lemma game
  - To show that a language L is not regular
    - Assume L is regular
    - Choose an "appropriate" string x in L
    - Express x = uvw following rules of pumping lemma
    - Show that uv^kw is not in L, for some k
    - The above contradicts the Pumping Lemma
    - Our assumption that L is regular is wrong
    - L must not be regular

Pumping lemma

- Example:
  - $L = \{x \in \{0,1\}^* \mid 0^i1^i, i \geq 0\}$
  - Ex: 000111, 0011, Λ, 00001111
  - Let's play!
    - Assume that L is regular.
      - Then there is an FA, M that accepts L.
      - Let n be the number of states in M

Pumping lemma

- Example:
  - $L = \{x \in \{0,1\}^* \mid 0^i 1^i, i \geq 0\}$
  - Let's play
    - Choose an appropriate string $x \in L$
      - Let $x = 0^n 1^n$
    - Apply Pumping Lemma to x
      - $x = uvw$
      - $|uv| \leq n$
      - $|v| > 0$

Pumping lemma

- $x = uvw = 0^n 1^n$
  - 00 … 0 11 … 1
  - Since $|uv| \leq n$, uv must consists entirely of 0s and, as such, v must also consist entirely of 0s.
    - $v = 0^j$ for some $j \leq n$

Pumping lemma

- $x = uvw = 0^n 1^n = 0^i 0^j 0^k 1^n$ $i + j + k = n$
  - Let's pump!
  - By the Pumping Lemma
    - uv^2w is also in L
    - $uv^2w = 0^i 0^{2j} 0^k 1^n$
    - Certainly $i + 2j + k \neq n$
    - uv^2w has more 0's that 1's
    - Thus $uv^2w \notin L$ CONTRADICTION!

Pumping lemma

- We arrived at a contradiction,
  - Thus our original assumption that L is regular must be incorrect
  - Thus L is not regular.
- Note that we need to find only 1 string x that fails in order for the proof by contradiction to work.
  - The key is finding the x that won't work
- Questions?

Pumping Lemma

- Let's try another example:
  - $L = \{x \in \{0,1\}^* \mid 0^i x, |x| \leq i\}$
  - Ex: 0001, 0010, Λ, 0000101
  - Let's play!
    - Assume that L is regular.
      - Then there is an FA, M that accepts L.
      - Let n be the number of states in M

Pumping Lemma

- Another Example:
  - $L = \{x \in \{0,1\}^* \mid 0^i x, |x| \leq i\}$
  - Let's play
    - Choose an appropriate string $x \in L$
      - Let $x = 0^n 1^n$
    - Apply Pumping Lemma to x
      - $x = uvw$
      - $|uv| \leq n$
      - $|v| > 0$

Pumping Lemma

- $x = uvw = 0^n1^n$
  - 00 … 0 11 … 1
  - Since $|uv| \leq n$, uv must consists entirely of 0s and, as such, v must also consist entirely of 0s.
    - $v = 0^j$ for some $j \leq n$

Pumping Lemma

- $x = uvw = 0^n1^n = 0^i0^j0^k1^n$ $i + j+k= n$
  - Let's (un)pump!
  - By the Pumping Lemma
    - uv^0w is also in L
    - $uv^0w = uw = 0^i0^k1^n$
    - Certainly $n > i + k$
    - The length of the prefix of 0s is less than the suffix x
    - Thus $uv^0w \notin L$ CONTRADICTION!

Pumping Lemma

- We arrived at a contradiction,
  - Thus our original assumption that L is regular must be incorrect
  - Thus L is not regular.
- Note that we need to find only 1 string x that fails in order for the proof by contradiction to work.
  - We can show x not to work by pumping 0 times
- Questions?

Non-regular languages

- Informal notion of what regular languages can't express:
  - Counting and comparing
  - Any operation that implies the use of a stack
    - Pal
    - xx^r

Pumping Lemma

- Summary
  - The pumping lemma formalizes the idea that if a string is longer enough, eventually at least one state on the FA will be have to be repeated on the path that accepts the string.
  - Continually looping on this state will produce an infinite number of strings in the language
  - Used to show that languages are not regular

Looking ahead

- At this point, we have found languages that are not regular
  - So we need to look at a larger class of languages.
  - Start after the break.