CS 3313 Foundations of Computing:

Properties of Regular Languages: Non-regular Languages

http://gw-cs3313.github.io

1

Properties of Regular Languages

- Closure Properties: what happens when we "combine" two regular languages or perform set operations on them ?
- Decision Problems: can we provide procedures to determine properties of a language ?
- Next….How do we determine if a language does not belong to that class of languages ?
  - Ex: How do we show that a language (problem?) cannot be accepted by a DFA ?

2

Closure Properties

- Regular languages are closed under:
  - Union
  - Complement
  - Intersection
  - Concatenation
  - Star closure
  - Reversal
  - Set difference
  - homomorphism

3

Decision Properties

- A property is decidable if there is an algorithm that can determine if the property holds over the language
- Regular languages, properties that are decidable:
  - Membership (Is w accepted by M)
  - Emptiness (if L(M) empty)
  - Set containment (L1 is contained in L2)
  - Set difference
  - …..

4

Finite and Infinite Languages

- Theorem: If a language is a finite set then it is a regular languages
  - $L = \{ w_1, w_2, \ldots, w_n \mid \text{for some fixed } n \}$
- Proof:
- Question: Can we test if a regular language is finite or infinite ?

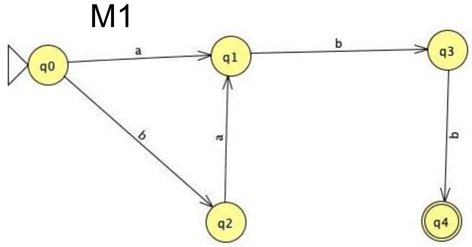
5

5

Transition graphs for two DFAs

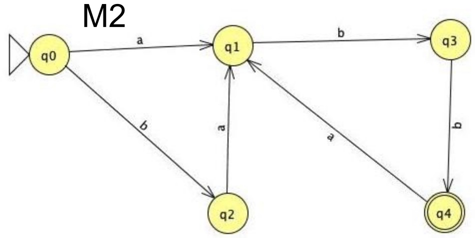
Is L(M1) finite ?

M1



Is L(M2) finite ?

M2



6

The Infiniteness Problem

- Theorem: Testing if L(M) is infinite is a decidable problem.
- Key idea: Homework 1
  - If there is a walk of length n or greater (from start to a final state) then there is a cycle in the graph
    - We can repeat the cycle any number of times

7

7

The Infiniteness Problem

- Theorem: Testing if L(M) is infinite is a decidable problem.
- Algorithm: compute all paths between all pairs of vertices (length n), and check if there is a cycle in the graph
  - Input: Transition graph for DFA M
  - Output: Yes if L(M) is infinite, No if L(M) is finite
  - Check if graph has a cycle (from start state to some final state)!
    - If cycle then L(M) is infinite else L(M) is finite

8

8

So what kinds of languages are not regular and how do we prove they are not ?

- Proof for testing infiniteness of L(M) reveals some properties that can be used to prove that a language is not regular.

- Given any language L, it is either regular or it is not.
  - To prove L is regular, we have to provide a DFA/NFA or Regular expression that accepts L.
  - To prove L is not regular, we need to provide a formal proof using some properties of all regular languages
    - Simply saying "I spent a lot of time and could not find a DFA" is NOT a proof.

9

Why is it useful to ask if a language is Regular

- Example: Can we check if there are syntax errors in a C program by using a DFA ?
  - Syntax checking is first step in a compiler's translation process
  - Program must satisfy the rules (specified as a grammar) of the C programming language (or any programming language)

10

Power of abstraction

- If a DFA can do syntax checking, then a DFA can check if there are an equal number of left and right braces ({ and } are used to specify a code block in C)
  - Choose $L = \{ w \mid w \text{ is a string over } a,b \text{ and } w \text{ has equal number of } a\text{'s and } b\text{'s}\}$
    - Using a to denote { and b to denote } (recall homomorphism which will let you substitute symbols)
- Now apply closure properties: we know that a*b* is regular and regular languages are closed under intersection
  - Therefore **L1 = L ∩ a*b* = {a^i b^i | i >1}** must be a regular language
    - Equal number of a's and b's
- So, is L1 a regular language ?

11

"power" of DFAs: A little intuition

- So what can DFAs (i.e., finite state machines) "compute" ?
- What can they store and where ?
  - State
  - Do they have an "external" memory to store a value ?
- Example: A DFA for $L = \{ a^j b^j \}$? (equal number of a's and b's)

12

"power" or limits of DFAs….

- Key takeaway: If we have a solution that requires external storage of arbitrary size (value of a counter, storing an entire string, etc.) then DFAs don't have that capability

- Ex: $L = \{ a^j b^j \}$ or $L = \{ w\, w \mid w \text{ in } \{0,1\}^* \}$
- But is this argument a formal proof ?
- How do you prove a language is not regular ?

13

How do we prove a language is not regular….constructing the formal framework

- Note: we are only interested in infinite language (since all finite languages are regular)
- If a language is regular then it is accepted by some DFA with n states
  - We don't know what n is …just that it exists
  - DFA represented by a graph… has n vertices (for the n states)
- When is a string w accepted by a DFA ? In terms of transition graph of the DFA

14

Walk/Paths in a graph – recall HW1 proof

- Graph has n vertices
- Walk/Path in a graph can be represented as a sequence of vertices: $v_1v_2v_3\ldots v_k$ where each (v_i, v_j) is an edge
- Recall proof from HW1: Suppose we have a path of length n, how many vertices on the path ?

15

DFA Transition Graph

- The transition function of a DFA can be represented as a (directed graph).
- DFA has a finite number of states: n
- If DFA accepts an infinite language, then it *must* accept a string of length $>= n$
  - Else it is a finite set !
- Suppose there is a string of length $m >= n$ accepted by the DFA
  - Vertex sequence in the path = ?
  - $p_1\, p_2\, p_3 \ldots p_i \ldots p_j \ldots p_n\, p_{n+1} \ldots p_m\, p_{m+1}$

16

Cycles in the path ?

- DFA M= (Q, **Σ**, δ, q_0. F) and |Q| = n (there are n states)
- M accepts string w of length $m >= n$, $\delta(q_0,w) \epsilon F =>$

  There is a walk of length (m+1) > n with vertex sequence of (m+1) vertices

$p_1\, p_2\, p_3 \ldots p_i \ldots p_j \ldots p_n\, p_{n+1} \ldots p_m\, p_{m+1}$ with $p_i \epsilon Q$, $p_1 = q_0$, $p_{m+1} \epsilon F$

We have n unique vertices, therefore from pigeon hole principle:

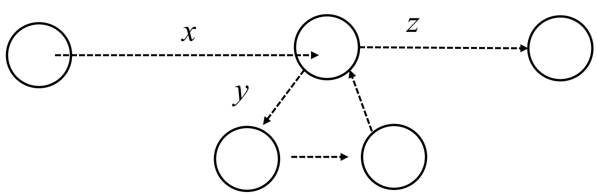
there must be two states p_i and p_j from first n+1 in { $p_1\, p_2\, p_3 \ldots p_i \ldots p_j \ldots p_n\, p_{n+1}$ } *such that* $p_i = p_j$ *(they are the same)*

The walk in DFA for input w is therefore:

$p_1\, p_2\, p_3 \ldots p_i \ldots p_{j-1}\, p_i\, p_{j+1} \ldots p_n\, p_{n+1} \ldots p_m\, p_{m+1}$

17

Paths in the graph….



$w = xyz$, $|y| <= 1$, $|xy| <= n$

$\delta(q_0, x) = p_i$ $\delta(p_i, y) = p_i$ $\delta(p_i, z) = p_{m+1} \epsilon F$

$$\delta(q_0, x\, y^i\, z) = p_{m+1} \epsilon F$$

therefore, if $w \epsilon L$ *then* $x\, y^i\, z \epsilon L$ *for all* $i >= 0$

18

The Pumping Lemma for Regular Languages

For every regular language L

There is an integer n, such that

For every string w in L of length $\geq n$

We can write $w = xyz$ such that:

1. $|xy| \leq n$.
2. $|y| > 0$.
3. For all $i \geq 0$, xy^iz is in L.

Number of states of DFA for L

Labels along first cycle on path labeled w

19

19

Example: $L = \{ a^i b^i \mid i >= 0 \}$

- L is not regular. Prove by contradiction
- Assume L is regular….(and apply pumping lemma)

20

Pumping Lemma as Adversarial Game

- 1: Player 1 (me) picks the language to be proved nonregular

  $L = \{ ww \mid w \in \{a,b\}^* \}$

- 2. Player 2 picks n, but does'nt reveal to player 1 what n is; player 1 must now devise a play for all possible n's

  important: you cannot assume a value for n !

- 3. Player 1 picks w, which may depend on n and which must be of length at least n

  pick $w = a^n b^n a^n b^n$ *(note: we express string in terms of* n *)*

21

Pumping Lemma as Adversarial Game

- 4: Player 2 divides w into x,y,z obeying the constraints that are stipulated in the lemma: y is not empty and $|xy| <= n$.
  - Again, Player 2 does not tell Player 1 what xyz are; just that they obey the constraints

  $w = xyz$ $\quad$ y *is not* λ *(since* $|y| >= 1$*)* $\quad |xy| <= n$

  since $|xy| <= n$*, string* xy *consists entirely of* a*'s*

  let $|x| = m_1$ *and* $|y| = m_2$ *and* $m_2 >= 1$

  $x = a^{m1}$ $\quad y = a^{m2}$ $\quad z = a^{n-m1-m2} b^n a^n b^n$

- 5. Player 1 "wins" by picking k, which may be a function of $n,x,y,$ and z such that xy^kz is not in L.

  pick $i=0$ *and consider* $xy^0z = a^{m1} a^{n-m1-m2} b^n a^n b^n = a^{n-m2}b^na^nb^n$

  since $m_2 >= 1$*,* $n - m_2 < n$

  therefore $a^{n-m2}b^na^nb^n$ *is not in* L*…Contradiction.*

22

Power of abstraction & Combining theorems

- If a DFA can do syntax checking, then a DFA can check if there are an equal number of left and right braces ({ and } are used to specify a code block in C)
  - Choose L = { w | w is a string over a,b and w has equal number of a's and b's}
    - Using a to denote { and b to denote } (recall homomorphism which will let you substitute symbols)
- Now apply closure properties: we know that a*b* is regular and regular languages are closed under intersection
  - Therefore **L1 = L ∩ a*b* = {$a^i b^i$ | i >1}** must be a regular language
    - Equal number of a's and b's
- So, is L1 a regular language ?

23

Example: L = { w | w does not have an equal number of a's and b's}

- Two approaches:
  1. Directly apply pumping lemma to get a contradiction by picking some string *w*
  2. Use previous results and closure properties of regular languages

24

Exercise:

- *Is the following language regular, prove or disprove:*

  $L = \{0^i 1^j 2^i 3^j \quad | \; i,j > 0 \}$

- Can you come up with a short proof using previous proofs/results and properties of regular languages ?