CSC236 fall 2018

languages: the last words

Danny Heap
heap@cs.toronto.edu / BA4270 (behind elevators)
http://www.teach.cs.toronto.edu/~heap/236/F18/
416-978-5899

Using Introduction to the Theory of Computation, Chapter 7

Outline

non-regular languages

need... more... power

notes

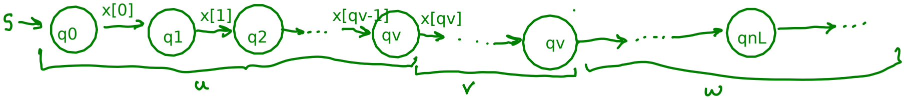
pumping lemma (see course notes, page 234)

If $L \subseteq \Sigma^*$ is a regular language, then there is some $n_L \in \mathbb{N}$ (n_L depends on L) such that if $x \in L$ and $|x| \geq n_L$ then:

- $\exists u, v, w \in \Sigma^*, x = uvw$
- $|v| > 0$
- $|uv| \leq n_L$
- $\forall k \in \mathbb{N}, uv^k w \in L$

if uvw accepted, so are uw, uvvw, uvvvvw, uvvvvvvvw

idea: if machine $M(L)$ has $|Q| = n_L$, $x \in L \wedge |x| \geq n_L$, denote $q_i = \delta^*(q_0, x[:i])$, so x "visits" $q_0, q_1, \ldots, q_L$ with the first $n_L + 1$ prefixes of x... so there is at least one state that x "visits" twice (pigeonhole principle)



consequences of regularity

How about $L = \{1^n 0^n | n \in \mathbb{N}\}$

members of L: "", 10, 1100, 111000, 11110000

Proof, by contradiction, that L is not regular.
Assume, for sake of contradiction, that L is regular, so there is some FSM M that accepts L. Then the number of states in M is some positive integer m.

consider the string 1^m0^m. By pumping lemma, 1^m0^m = uvw such that |v|>0, |uv| <= m and for all k \in \N, uv^km \in L. But then uv^0w is in L, but uv^0w has m-|v| 1s and m 0s.

Contradiction.

another approach...Myhill-Nerode

Consider how many different states $1^k \in \text{Prefix}(L)$ end up in...for various k

Proof by contradiction. Assume L is regular, so it's accepted by M with m = |Q|. Then the prefixes 1^0, 1^1, 1^2, ..., 1^m are sent to just m different states, so (pigeonhole principle) there must exist 0 <= h < i <=m such that \delta^*(q0, 1^h) = \delta^*(q0, 1^i). But then we know that \delta^*(q0, 1^h0^h) is an accepting state, but \delta^*(q0, 1^i0^h) is thus an accepting state. contradiction: i \not = h, and so 1^i0^h is not in L.

Since assuming that L is regular leads to a contradiction, that assumption is false.

"real life" consequences...

- the proof of irregularity of $L = \{1^n 0^n | n \in \mathbb{N}\}$ suggests a proof of irregularity of $L' = \{x \in \{0,1\}^* \mid x$ has an equal number of 1s and 0s$\}$ (explain... consider $L' \cap L(1 * 0*)$)
- a similar argument implies the irregularity of $L'' = \{x \in \Sigma^* \mid$ x has an equal number of $\langle div \rangle$ as of $\langle / div \rangle$ substrings$\}$, where $\Sigma = \{a, ..., z, \langle, \rangle, /\}$... so html cannot be checked by a DFSA!
- what about $L''' = \{(w, w) \mid w \in \{0,1\}^*\}$? What does this say about whether an FSA can check whether a pair of strings is equal?

How about $L = \{w \in \Sigma^* \mid |w| = p \wedge p \text{ is prime}\}$

Prove it not regular by contradiction.
For the sake of contradiction, assume L is regular. So, there must be some machine M with |Q| = m \in \N^+ states that accepts it.

Since there are infinitely many primes, there always a prime number (several, actually) larger than m.
Let p be a prime with p > m, so 1^p \in L, by definition.

Also 1^p = uvw, where |uv| <= m, and |v| > 0 and uv^kw \in L, for any natural number k.
so |uvw| = p, a prime, but also |uw| must be prime, also |uvvvvvw| must be prime.
But |uv^{1+p}w| = p + p|v| = (1+|v|)p ---><---- Contradiction!, that string has composite length!

a humble admission...

- at any point in time my computer, and yours, are DFSAs

- do the arithmetic... figure out the number of states in my machine... crashed!

- however, we could dynamically add/access increasing stores of memory also, most "practical" problems fit in our little DFSAs

PDA

- DFSA plus an infinite stack with finite set of stack symbols. Each transition depends on the state, (optionally) the input symbol, (optionally) a pop from stack
- each transition results in a state, (optional) push onto stack

design a PDA that accepts $L = \{1^n 0^n \mid n \in \mathbb{N}\}$.

see vassos's note page 252

CFG for the same thing

S -> 1S0
S -> ""

yet more power

- (informally) linear bounded automata: finite states, read/write a tape of memory proportional to input size, tape moves are one position L-to-R
- (informally) turing machine: finite states, read/write an infinite tape of memory, tape moves are one position L-to-R

Each machine has a corresponding **grammar** (e.g. FSAs$\leftrightarrow$regexes (right-linear grammar))

review suggestions

- three hours, pencils, pens, erasers, caffeine, sugar
- I will announce some office hours during study period
- **review**: lecture slides, tutorial exercises and solutions, assignments and solutions
- **invent** questions similar to those in the previous bullet point, vary and extend the questions
- **form**: study groups to challenge each other
- **ask**: me about things that are still unclear
- **if you still have time**: look at previous exams for presentation andn length

notes