CSC236 fall 2018

languages: the last words

...plus some exam review tips...

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

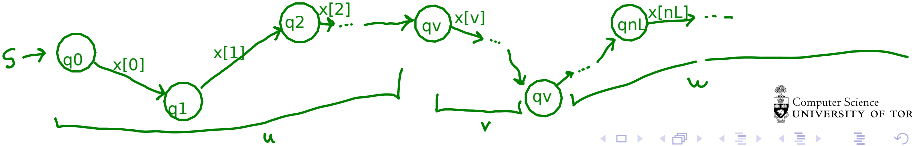
pumping lemma (see course notes, page 234)

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:

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

The magic number nL is the number of states in some DFSA that allegedly accepts L...

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 $L = \{1^n0^n | n \in \mathbb{N}\}$

Assume, for the sake of contradiction, that L is regular. Then there must be a machine M that accepts L.
M has |Q| = m > 0 states. Consider 1^m0^m. Then, but the pumping lemma, 1^m0^m = uvw, where
|uv| <= m and |v| > 0 and for all k \in \N, uv^kw \in L. But then uvvw \in L, and uvvw has m + |v| 1s followed
by m 0s ---><--- contradiction, elements of L have same number of 1s and 0s.

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

another approach...Myhill-Nerode

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

Assume, for the sake of contradiction, that L is regular. Then there is a machine M that accepts L, and M has some number of states, say |Q| = m. Consider the prefixes 1^0, 1^1, ..., 1^m. Since there are m+1 of these, at least two drive M to the same state, so there are 0 <= h < i <= m such that 1^h and 1^i drive M to the same state. But then 1^h0^h and 1^i0^h both drive the machine to the same state, and 1^h0^h should be accepted, whereas 1^i0^h should not (i not = h). ---><---

Since assuming that L is regular led 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
  $L' = \{x \in \{0,1\}^* \mid x \text{ has an equal number of 1s and 0s}\}$
  (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 $L = \{w \in \Sigma^* \mid |w| = p \wedge p \text{ is prime}\}$

Assume, for the sake of contradiction, that L is regular. So there is some machine M with m = |Q| states that accepts L. Let p be a prime that is no smaller than m. Then 1^p has length >= m and 1^p = uvw where |v|>0 and uv^kw \in L for all natural numbers k. Then uv^{1+p}w \in L. |uv^{1+p}w| = p + p|v| = (1 + |v|)p, a composite number --------------><---------------- contradiction, L consists of only strings of prime length.

By assuming L is regular we arrived at a contradiction, so that assumption is false!

a humble admission...

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

  that is, a snapshot of your computing power...

- do the arithmetic...

  My laptop has 66108489728 bits of ram and 843585945600 bits of disk storage, so it can be in 2^{66108489728 + 843585945600} different states. Big, but finite...
  (of course I didn't count GPUs, various registers and cashes, and other peripherals, but the result is the same...)

- however, we could dynamically add/access increasing stores of memory

  i.e., run over to Spadina and College and buy some more RAM as needed by a computation...

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 See p 252 course notes.
- each transition results in a state, (optional) push onto stack

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

a context-free grammar (set of production rules) that recognizes this language

S -> 1S0
S -> \varepsilon

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

  Some people claim this is a realistic model of current computers...

- (informally) turing machine: finite states, read/write an infinite tape of memory, tape moves are one position L-to-R

  ... but most treat this as the benchmark for what is computable...

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

review suggestions

- three hours, pencils, pens, erasers, caffeine, sugar

  put everything else underneath your desk during exam...
- I will announce some office hours during study period

  Dec 14 1--3, December 17 2--4:30, December 18 2--4...
- **review:** lecture slides, tutorial exercises and solutions, assignments and solutions

  These were my inspiration when designing questions... *not* previous exams...
- **invent** questions similar to those in the previous bullet point, vary and extend the questions

  Try to design questions that take 15--30 minutes to solve...
- **form:** study groups to challenge each other

  social skills, social skills... you need these!
- **ask:** me about things that are still unclear

  come to office hours, above
- **if you still have time:** look at previous exams for presentation andn length

  chances are very small that I'll repeat an old question...

notes