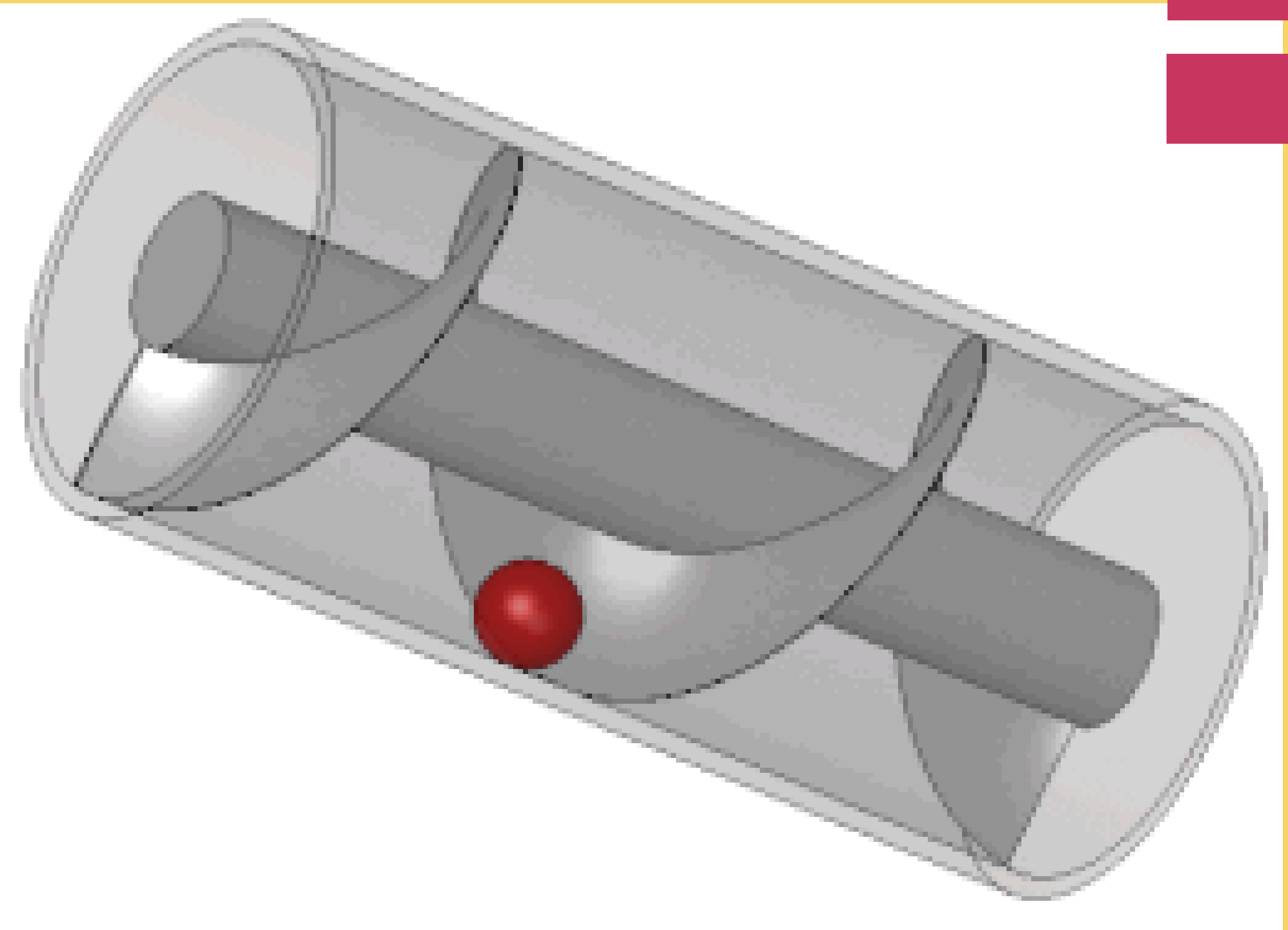
HW 2 due this Saturday, Feb 7 (10:00pm)

Quiz 3 due next Monday, Feb 9



https://en.wikipedia.org/wiki/Archimedes%27_screw

Class 6: Limitations of Regular Expressn

University of Virginia
CS3120: DMT2
https://weikailin.github.io/cs3120-toc
Wei-Kai Lin

Plan

Regular Language vs DFA

Limitations

Pumping Lemma

Today: Chapter 6 in the TCS book

https://introtcs.org/public/lec_05_infinite.html#regexpsec

Recap:

Theorem:

Reg-Fun = DFA-Comp

Definition 6.6 (Regular expression)

A *regular expression* e over an alphabet Σ is a string over $\Sigma \cup \{(,),|,*,\emptyset,""\}$ that has one of the following forms:

1. $e = \sigma$ where $\sigma \in \Sigma$
2. $e = (e'|e'')$ where e', e'' are regular expressions.
3. $e = (e')(e'')$ where e', e'' are regular expressions. (We often drop the parentheses when there is no danger of confusion and so write this as $e'\ e''$.)
4. $e = (e')^*$ where e' is a regular expression.

Finally we also allow the following "edge cases": $e = \emptyset$ and $e = ""$. These are the regular expressions corresponding to accepting no strings, and accepting only the empty string respectively.

Definition 6.2 (Deterministic Finite Automaton)

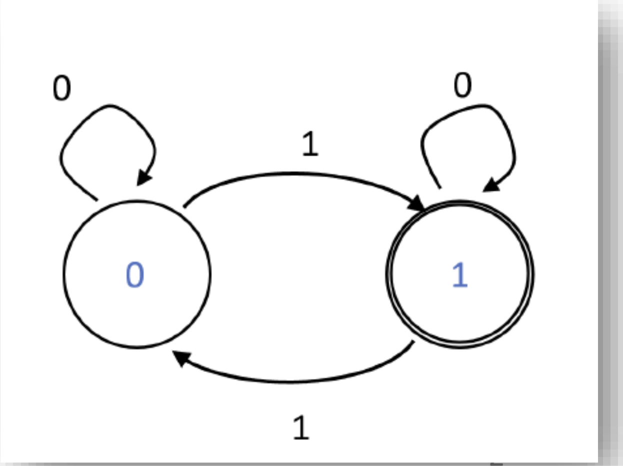
A deterministic finite automaton (DFA) with C states over $\{0,1\}$ is a pair $(T, \mathcal{S})$ with $T : [C] \times \{0,1\} \to [C]$ and $\mathcal{S} \subseteq [C]$. The finite function T is known as the *transition function* of the DFA. The set $\mathcal{S}$ is known as the set of *accepting states*.

Let $F : \{0,1\}^* \to \{0,1\}$ be a Boolean function with the infinite domain $\{0,1\}^*$. We say that $(T, \mathcal{S})$ *computes* a function $F : \{0,1\}^* \to \{0,1\}$ if for every $n \in \mathbb{N}$ and $x \in \{0,1\}^n$, if we define $s_0 = 0$ and $s_{i+1} = T(s_i, x_i)$ for every $i \in [n]$, then

$$s_n \in \mathcal{S} \Leftrightarrow F(x) = 1$$

$1101(0|1)^*1000$

$0^*1(0^*10^*10^*)^*$



Recap: Reg-Fun $=$ DFA-Comp

- $F \in$ Reg-Fun iff exists e s.t. $\Phi_e(x) = F(x)$ iff
  $F \in$ DFA-Comp iff exists DFA M s.t. $M(x) = F(x)$

- Every reg. exp. e, matching $\Phi_e(x)$ is computable in time $O(|x|)$ for all $x \in \{0,1\}^*$
- Write DFA gives regular expression, and vice versa

More Implication of Reg-Fun $=$ DFA-Comp

Complement of regular expression?

No substring

- For any reg exp e, is there a “negate” reg exp e’?
- I.e., to find e' such that for all string x,

$$\Phi_{e'}(x) = NOT(\Phi_e(x))$$

- Highly asked question!

Stack Overflow
https://stackoverflow.com › questions › how-to-negate-...

How to negate specific word in regex? [duplicate]

A great way to do this is to use negative lookahead: ^(?!.***bar**).*$ The negative lookahead construct is the pair of parentheses, with the opening parenthesis ...

How to **negate** the whole **regex**? - Stack Overflow	6 answers	Apr 14, 2010
Regular expression to match a line that doesn't ...	34 answers	Jan 2, 2009
how to **negate** any **regular expression** in Java ...	3 answers	Dec 22, 2011
Regular Expressions and **negating** a whole character ...	9 answers	Jun 10, 2009

More results from stackoverflow.com

12 Answers

1044

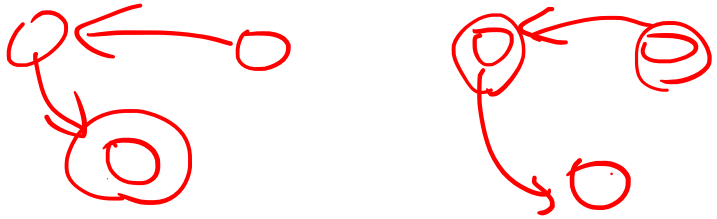
A great way to do this is to use negative lookahead:

```
^(?!.*bar).*$
```

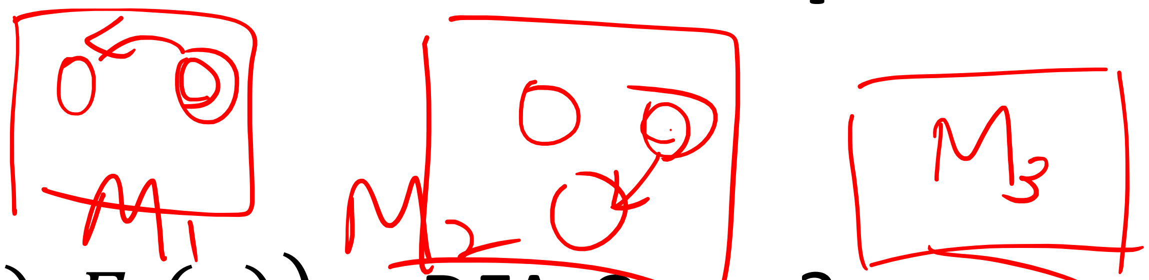
Reg-Fun = DFA-Comp

Complement of regular expression

- Suppose F is regular ($F = \Phi_e$ for some e)
- Does $\bar{F} = NOT(F(x))$ also have a regular expression?
- Yes!
- Suppose M is a DFA for F
- Let $\bar{M}$ be DFA that switches the accept/reject states of M
- $\bar{M}$ computes $\bar{F}$ $\Rightarrow \bar{e}$ s.t. $\Phi_{\bar{e}}(x) = NOT(\Phi_e(x))$
- Then $\bar{F} \in$ DFA-Comp = Reg-Langs



OR of DFA-comp functions are DFA-comp?



- Suppose $F_1, F_2 \in$ DFA-Comp.
- Does it hold $F(x) = OR\big(F_1(x), F_2(x)\big) \in$ DFA-Comp?
- Yes!
- Suppose e_b is a reg expression for F_b
- Let $e_3 = (e_1)|(e_2)$
- F is Φ_{e_3} $\Rightarrow$ M_3
- Therefore $F \in$ Reg-Fun = DFA-Comp

Reg-Fun = DFA-Comp

AND of DFA-comp functions are DFA-comp?

M_1 M_2 M_3

- Suppose $F_1, F_2 \in$ DFA-Comp.
- Does it hold $F(x) = AND\big(F_1(x), F_2(x)\big) \in$ DFA-Comp?
- Yes!
- Suppose M_b is a DFA for F_b for $b = 1{,}2$
- Want to construct M s.t. $M(x) = F(x)$ using M_1, M_2
- ?
-
-

How about more complicated transformations?

- Suppose functions $f_1, f_2, \ldots, f_k$ are all regular functions.
- Let $T: \{0,1\}^k \to \{0,1\}$ be an arbitrary Boolean function. for some $k \in \mathbb{N}$
- Is the function $f(x) = T\big(f_1(x), f_2(x), \ldots, f_k(x)\big)$ also regular?
  DFA/exp
- Yes!

- Proof sketch: $\text{AND}(a, b) \equiv \text{NOT}(\text{OR}(\text{NOT}(a), \text{NOT}(b)))$
  - Union of two languages is the same as OR of their Boolean functions
  - Complement of a language is the same as NOT of its Boolean function
  - {OR,NOT} is universal set of gates.

How to construct DFA / Reg Exp?

Write a DFA or a regular expression that matches all binary strings has no substring XXX. (HW 2)

In general, how to get DFA from any regular expression (and vice versa)?

This is sufficient to prove “Reg-Fun = DFA-Comp”
Stay tuned.

Limits of finite state computation

There is at least one non-regular function

- Proof:

There is at least one non-regular function

$0 \; 1 \; (\;) \; | \; * \; \phi \; ""$

- Proof:
- Reg-Fun is a countable set (why?) $|e| = k$ for some $k \in \mathbb{N}$
- Set of all Boolean functions (Bool-Fun) is uncountable (why?) proven mod 0 $\{0,1\}^* \longrightarrow \{0,1\}$

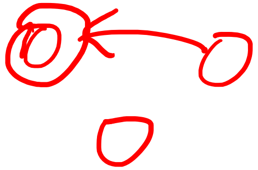
$|\text{Bool Fun}| > |\text{Reg Fun}|$

- If Bool-Fun ⊆ Reg-Fun, then Bool-Fun would have been countable

Any “natural” languages that is not regular?

Boolean Functions

- By Reg-Fun = DFA-Comp,
  Any regular function must be computable by a DFA (Thm Reg = DFA)
- Any DFA has constant “memory”, ie, num of states



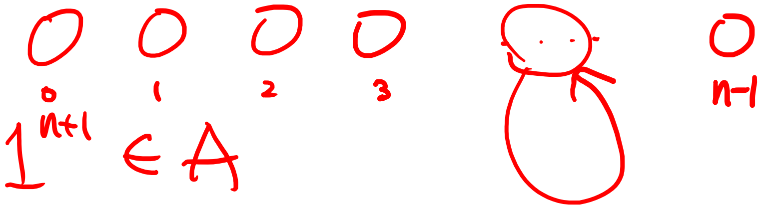
- Find a function needs more than const mem

((((...))))

Example A = $\{0^k 1^k : k \in N\}$

Theorem: A is not regular

Proof: by contra, $\exists$ DFA M w/ n states



Consider $\underbrace{0^{n+1}}\, 1^{n+1} \in A$

first 0s must visit a state twice

$\Rightarrow \exists\ m \geq 1$ $\quad 0^{n+1-m}\, 1^{n+1}$ accepted

Example A = $\{\mathbf{0}^{k}\mathbf{1}^{k}\text{:}\ \boldsymbol{k} \in$ N $\}$

Theorem: A is not regular

Proof:

Give it 00 … 0 … as inputs till a state q is repeated.

Let $x = 0^{i}, y = 0^{j}$ such that $j > 0$ and
both x and $x \mid\mid y$ on q.

Consider $0^{i}1^{i}$ and $0^{i+j}1^{i}$.

Either both will be accepted,

Or both will be rejected.

Pumping Lemma

Pumping Lemma

Let e be a regular expression over the alphabet of bits, $\{0,1\}$.

There is some number n_0 such that
for every $w \in \{0,1\}^*$ with $|w| > n_0$ and $\Phi_e(w) = 1$,
we can write $w = xyz$ for strings $x, y, z \in \{0,1\}^*$
satisfying the following conditions:

- $|y| \geq 1$
- $|xy| \leq n_0$
- $\Phi_e(x\, y^k\, z) = 1$ for every natural number k

Proof

Proof

Reg-Fun = DFA-Comp ==> exists DFA $M(w) = \Phi_e(w)$

Let n_0 be the number of states in M

Any t-bit string must visit $t + 1$ states.

For any $|w| > n_0$ that $M(w) = 1, w$ must visited the same state s twice in the first n_0 symbols.

Let i_0, i_1 be first and second time visited s.

$i_1 - i_0 \geq 1.\ i_1 \leq n_0.$

Let $x = w[:\ i_0], y = w[i_0 :\ i_1]$ and z the remaining.

Repeating y or not give the same output

Ex: $L = \{01110\}$ is regular

n_0: 5

w st $|w| > n_0$: No such w that $\Phi_e(w) = 1$

Theorem 6.21 (Pumping Lemma)

Let e be a regular expression over some alphabet Σ. Then there is some number n_0 such that for every $w \in \Sigma^*$ with $|w| > n_0$ and $\Phi_e(w) = 1$, we can write $w = xyz$ for strings $x, y, z \in \Sigma^*$ satisfying the following conditions:

1. $|y| \geq 1$.
2. $|xy| \leq n_0$.
3. $\Phi_e(xy^k z) = 1$ *for every* $k \in \mathbb{N}$.

$\{ 1, 11, 111, \ldots \}$

Ex: all-1 strings is regular

Yes, 1*

n_0: 5

w st $|w| > n_0$:

$= 111111, \; 1111111, \; \ldots$

Theorem 6.21 (Pumping Lemma)

If

...r expression over some alphabet Σ. $\boxed{\text{Then}}$ there is some number n_0 such that for ... ith $|w| > n_0$ and $\Phi_e(w) = 1$, we can write $w = xyz$ for strings $x, y, z \in \Sigma^*$ ... lowing conditions:

$\exists$ $\forall$ $\forall$

1. $|y| \geq 1$.
2. $|xy| \leq n_0$.
3. $\Phi_e(xy^k z) = 1$ for every $k \in \mathbb{N}$. $\forall$

Not helpful
We already showed a reg. exp.

Ex: Palindroms is not regular

Lemma: e is regular expression, then
There is some number n_0 such that
for every $w \in \{0,1\}^*$ with $|w| > n_0$ and $\Phi_e(w) = 1$,
we can write $w = xyz$ for strings $x, y, z \in \{0,1\}^*$
satisfying the following conditions:

- $|y| \geq 1$
- $|xy| \leq n_0$
- $\Phi_e(x\, y^k\, z) = 1$ for every natural number k

A string x is palindrome iff x is identical to reverse(x).

Proof.

For any n_0, let $w =$

Where $w = xyz$ for $x =$ $y =$ $z =$