CS 121 Lecture 11

Oct. 8, 2019

Regular Expressions and Finite Automata

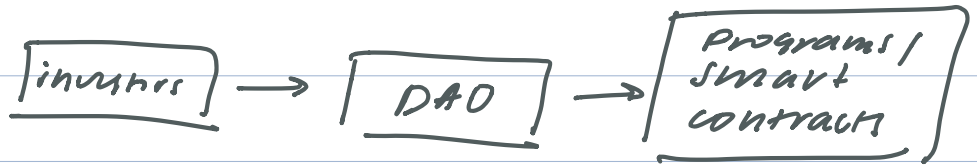
Resources

- Sipser, book exercises, prior midterms

Turing Completeness as a Bug

- Restricted programming languages: |TM| > |other prog|

- DAO (Distributed Autonomous Organization)



- ran into the HALTing problem, stuck in infinite loop

Restricted Models

- set of functions $f: \{0,1\}^* \to \{0,1\}^*$ s.t.
  - Description D_f of function f yields algo to compute f (algo always terminates)
  - Compute semantic properties of D_f:
    - Emptiness: $D_f \cong 0$
    - Fullness: $D_f \cong 1$
    - Equivalence: $D_f \cong D_g$

context-free grammars

regular expressions

Regular Expressions

alphabet $\Sigma = \{0,1\}$

Expression e corresponds to $\phi_e : \Sigma^* \to \{a, \}$ ← matching function

$e = (00|11)^*$ $\quad \Phi_e(x) = \begin{cases} 1 & x = 00\,11\,00\,00\,00 \\ 0 & \text{otherwise} \end{cases}$

Expression e		Function Φ_e
$\emptyset$	null	$\emptyset(\forall x\ \Phi_e(x) = 0)$
$u\,u$	empty	$\Phi_e(x) = 1$ iff $x = ""$
σ	number	$\Phi_e(x) = 1$ iff $x = \sigma$
$e \mid e'$	or	$\Phi_{e\mid e'}(x) = \Phi_e(x) \vee \Phi_{e'}(x)$
$e\ e'$	concat	$\Phi_{ee'}(x) = 1$ iff $\exists u,v \in \Sigma^*$ s.t. $x = uv \wedge \Phi_e(u) = 1 \wedge \Phi_{e'}(v) = 1$
e^*	for all lengths	$\Phi_{e^*}(x) = 1$ iff $\exists u_0 \dots u_{k-1} \in \Sigma^*$ s.t. $x = u_0 \dots u_{k-1}$ and $\forall i \in [k]$ $\Phi_e(u_i) = 1$

Ex: Let $\Sigma = \{0,1\}$. What is Φ_e for

$e = (1|2|3|4|5|6|7|8|9)(0|1|2|3|4|5|6|7|8|9)^*$

decimal representation of a integer

Ex: Let $\Sigma = \{0,1\}$. Find e s.t. $\Phi_e(x) = 1$ iff x contains 0110 as a substring

$e = (0|1)^*(0110)(0|1)^*$

Ex: Let $\Sigma = \{0,1\}$. Find e s.t. $\Phi_e(x) = 1$ iff $\sum x_i = 0 \bmod 3$

$e = ((0)^*(1)(0)^*(1)(0)^*(1)(0)^*)^* \mid 0^*$

$e = 0^*(10^*10^*10^*)^*0^*$

Theorem: Let e be a regular expression. Then the function $\Phi_e : \{0,1\}^* \to \{0,1\}$ is computable.

Moreover, $\exists$ algorithm computing $\Phi_e(x)$ for $x \in \{0,1\}^n$ in $O(n)$ time.

Moreover, $\exists$ such algorithm making single pass over x with $O(1)$ memory.

Deterministic Finite Automaton

Moreover, $F = \Phi_e$ for some e iff $\exists$ DFA computing F.

F is some function/language

$$L_F = \{x \mid F(x) = 1\}$$

Proof of Theorem:

MATCH(e, x):

input: e regexp, $x \in \{0,1\}^n$

operations:

if $e = ""$ or $e = 0$ or $e = 1$: return 1 iff $x = e$

if $e = e' \mid e''$:

return MATCH(e', x) $\vee$ MATCH(e'', x)

if $e = e'e''$:

return $\bigvee_{i=0}^{n}$ (MATCH(e', $x_0 \dots x_i$), $\wedge$ MATCH(e'', $x_{i+1} \dots x_{n-1}$))

if $e = (e')^*$:

return $\bigvee_{k=1}^{n} \exists\, 0 = i_0 < i_1 < \dots < i_k = n \wedge \bigwedge_{j=0}^{k-1}$ MATCH(e', $x_{i_j} \dots x_{i_{j+1}-1}$)

Claim. $MATCH(e,x) = \Phi_e(x)$ for every e, x

Pf. Induction on $|e|$

$|e| \leq 1$: obvious

If e of form $e'|e''$, $e'e''$ or $(e')^*$ follows from induction hypothesis since $|e'|, |e''| < |e|$. □

Proof of Moreover #1: $\exists$ $O(|x|)$ time alg $FMATCH(e,x)$ to compute $\Phi_e(x)$.

Lemma 1. If e is a regex, exist $e[0]$ and $e[1]$ (and compute them) s.t.

$$\Phi_{(e[0])}(x) = \Phi_e(x0) \qquad \Phi_{(e[1])}(x) = \Phi_e(x1)$$

Q: if e is a regex s.t. $\Phi_e(x) = 1$ iff $\sum_i x_i \bmod 3 = 0$ what is $\Phi_{e[1]}$?

$$\Phi_{e[1]}(x) = \begin{cases} 1 & \text{when } \sum_i x_i \bmod 3 = 2 \\ 0 & \text{o/w} \end{cases}$$

$e[1] = e(0^*10^*10^*)$

Proof of Lemma 1: (skip basic)

if $e = e'|e''$ then $e[1] = e'[1]|e''[1]$

if $e = e'e''$ then $e[0] = e'e''[0] \mid e'[0]$ if $\Phi_{e'}(\varepsilon) = 1$

if $e = (e')^*$ then $e[1] = (e')^* e'[1]$

Lemma 2. If e is a regex, $|e[0]|, |e[1]|, |e[0][0]| \ldots \leq c(|e|)$

FMATCH(τ, X):

Input: τ regexp, $x \in \{0,1\}^n$

Operations:

if $X = \text{"}\text{"}$: return $\Phi_\tau(\text{"}\text{"})$

return FMATCH($\tau[x_{n-1}], x_0, x_1 \dots x_{n-2}$)

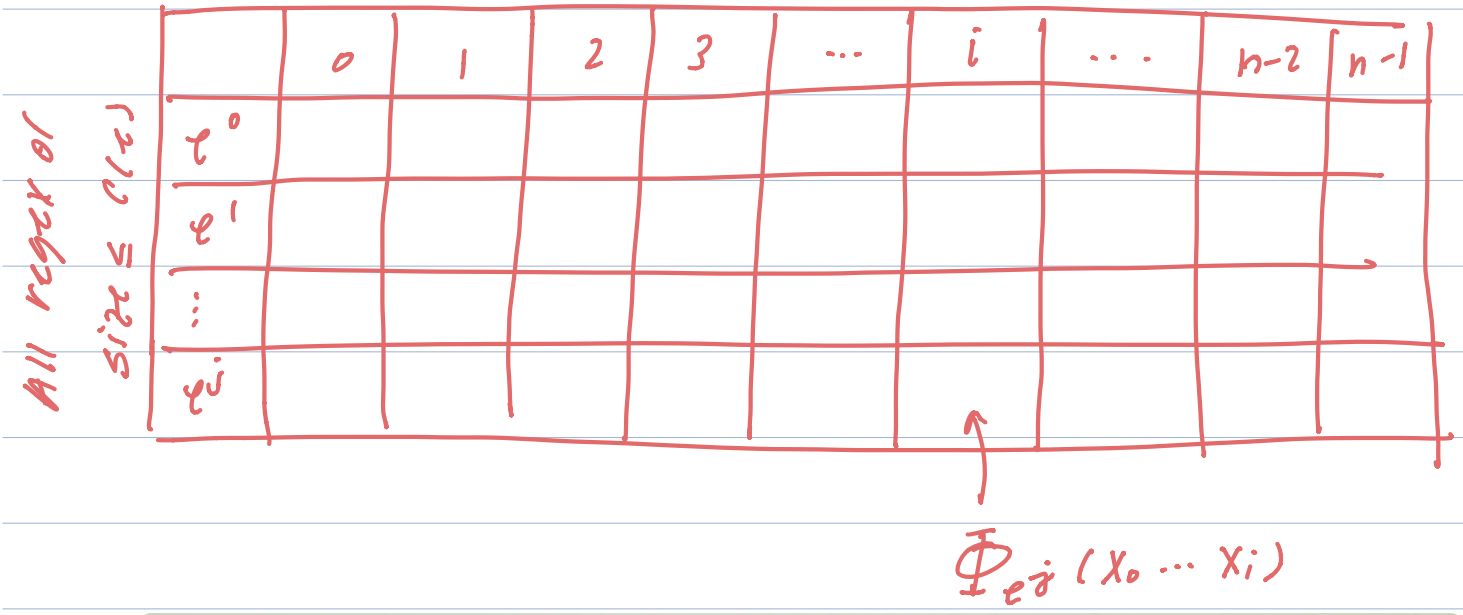
Let τ = orig expression

T_0 := max time to compute

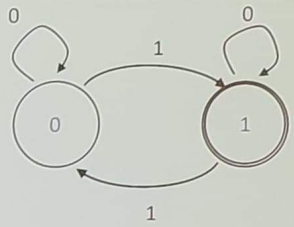
$\tau'[0]\ \tau'[1] \le C(|\tau|)$

$TIME(n) = T_0 + TIME(n-1)$

Using memoization to make it a single-pass $O(1)$ space algorithm: DFA



Deterministic Finite Automaton



Single pass constant memory algorithm.

- Constant number of states.
- Every step read x_i, change state based on whether $x_i = 0$ or $x_i = 1$
- When done output 0 or 1 based on state (accepting / rejecting).

Theorem: $F: \{0,1\}^n \to \{0,1\}$ regular if and only if F computable by a DFA.

Corollary: If $F: \{0,1\}^n \to \{0,1\}$ regular then $NOT(F)$ is also regular.

Corollary: If $F, G: \{0,1\}^n \to \{0,1\}$ regular then $NAND(F, G)$ is also regular.

Corollary: Let $EQ(e, f) = 1$ iff $\Phi_e = \Phi_f$. Then EQ is computable.

Pumping Lemma

Q: Let $e = (0000|111|0100)(020)^*(00111|11|00|11)$.
Prove that if $|x| > 100$ and $\Phi_e(x) = 1$ then x must contain the digit 2

Pumping Lemma: (Informal version). Let $F = \Phi_e$ for some e.
If $|w| > 2|e|$ and $\Phi_e(w) = 1$ then "we must use star" to match w.

Pumping Lemma: (formal version). Let $F = \Phi_e$ for some e and $n = 2|e|$.
If $|w| > n$ and $\Phi_e(w) = 1$ then $\exists x, y, z$ s.t. $w = xyz$, $|xy| \leq n$, $|y| \geq 1$ s.t.

$$\Phi_e\left(xy^k z\right) = 1$$

for every $k \in \mathbb{N}$