Lecture 2: Finite State Machines

CSCI 101
Spring, 2019
Kim Bruce
TA's:Gerard Bentley, Sarp Misoglu, Seana Huang, Danny Rosen, Alice Tan

Course web page: http://www.cs.pomona.edu/classes/cs101

Mentor Office Hours

- Start nSunday outside of my office:
  - SMTW: 8 to 10 p.m., 112 Edmunds

Homework

- Now available on line
  - Second problem has lots of parts
  - Turn in single file
- Can use JFLAP to create automata
  - See tutorial on-line - *you must read it!*
  - Save as gif and then open and save as pdf (e.g., using Preview on Mac)
  - \includegraphics{myfile.pdf} to insert in LaTeX file.

Homework Grading

- Uses gradescope
  - Log in at https://www.gradescope.com/courses/36442
  - Turn in pdf written using LaTeX
  - Each problem must use specified number of pages or grader won't find it!
    - *We will have several mentors grading at once and it will serve each only the pages for the program being graded!*
  - *Sample Hmwk 0 that gives points for trivial questions submitted properly!*

Deterministic Finite State Machine

- A FSM (or DFSM) is a quintuple (K, Σ, δ, s, A)
  - K is a finite set of states
  - Σ is a finite input alphabet
  - s ∈ K is the start state
  - A ⊆ K is set of accepting (or final) states
  - δ: K × Σ → K is transition function
- Simple model of real computer
  - finite memory

Example

Review: Computations

- Single step of M uses δ to process next character:
  - $(q_1,cw) \vdash_M (q_2,w)$ iff $\delta(q_1,c) = q_2$
- $\vdash_M{}^*$ is reflexive, transitive closure
  - $(q_1,u) \vdash_M{}^* (q_2,w)$ means get from first to second in 0 or more steps

Defining Language

- M *accepts* string w iff
  there is q ∈ A s.t. $(s,w) \vdash_M{}^* (q,\varepsilon)$
- M *rejects* string w iff
  there is q ∉ A s.t. $(s,w) \vdash_M{}^* (q,\varepsilon)$
- *L*(M) = { w ∈ Σ* | M accepts w }
- L is regular if it is *L*(M) for some finite state machine M

Proof by induction

- Simple: To prove for all n ≥ k, H(n), where H(n) is a proposition that may be true or false

typically k is 0 or 1

  - Prove base case: H(k)
  - Prove induction case: if, for some n ≥ k, H(k) holds then prove H(k+1)
- Course-of-values: To prove for all n ≥ k, H(n)
  - Let n ≥ k. Suppose that for all m < n, H(m) holds, then prove H(n)

Example

- Prove $2^n > n^2$ for all $n \geq 5$.
- Base case ($n = 5$) Show $2^5 > 5^2$
- Induction case: Suppose for some $k \geq 5$, $2^k > k^2$
  - Show $2^{k+1} > (k+1)^2$.
  - *We'll assume $k^2 > 2k + 1$ for $k \geq 5$, but could prove it by induction!*

Proving FSM is correct

- For each state of FSM, specify invariant.
- By induction on number of steps in computation, prove for all n, after n steps, if the computation is in state q, then invariant for q holds.
- Make sure invariants of final states imply correctness.

Example

- Draw FSM for w contains at least 2 b's.

Set Up Proof

- Invariants:
  - q_0: No b's have been read in so far
  - q_1: Exactly one b has been read in so far
  - q_2: At least 2 b's have been read in so far
- Base case: $n = 0$:
  - If in state q_0 after 0 steps then no b's have been read in
  - If in state q_1 after 0 steps then one b has been read in
  - If in state q_1 after 0 steps then at least 2 b's have been read in

Can't happen!!

Induction step

- Induction hypothesis H(n):
  - After n steps, if the computation is in state q, then invariant for q holds.
- Induction: Show if H(n) holds for some $n \geq 0$, then H(n+1) holds.
  - *Could go from H(n-1) to H(n) if $n \geq 1$ instead*

Closure Properties

- Regular languages are closed under:
  - Complementation $\Sigma^* - L$ (change final set)
  - Intersection $L_1 \cap L_2$ (product machine)
  - Union $L_1 \cup L_2$ (deMorgan laws or variant of product)
- How about concatenation?
  - $L_1 \| L_2$
    - Can't just put transitions from final of L1 to start of L2!
  - Also want L*

Intersection

- If $M_1 = (K_1, \Sigma, \delta_1, s_1, A_1)$, $M_2 = (K_2, \Sigma, \delta_2, s_2, A_2)$ then let $M = (K_1 \times K_2, \Sigma, \delta, \langle s_1, s_2 \rangle, A_1 \times A_2)$
  - where $\delta(\langle q_1, q_1' \rangle, a) = \langle q_2, q_2' \rangle$
    if $\delta_1(q_1, a) = q_2$ & $\delta_2(q_1', a) = q_2'$
- Then $L(M) = L(M_1) \cap L(M_2)$

Nondeterministic Finite State Machine

- An NDFSM is a quintuple $(K, \Sigma, \Delta, s, A)$
  - K is a finite set of states
  - Σ is a finite input alphabet
  - $s \in K$ is the start state
  - $A \subseteq K$ is set of accepting (or final) states
  - $\Delta \subseteq K \times (\Sigma \cup \{\varepsilon\}) \times K$ is a finite transition relation
- Can have multiple or no transitions
- ε-moves as well

Example

NDSM Computations

- Let $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$
- Single step of M uses Δ to process next character (or nothing):
  - $(q_1,cw) \vdash_M (q_2,w)$ iff $((q_1,c), q_2) \in \Delta$, for $c \in \Sigma$
  - $(q_1,w) \vdash_M (q_2,w)$ iff $((q_1,\varepsilon), q_2) \in \Delta$ (ε-move)
- Initial and accepting configurations and computations defined as before.

NDSM Computations

- NDSM accepts a word w if at least one of its computations accepts
  - Always guesses right path if there is one!
- Why NDSM's?
  - Easier to design!
  - But how to implement?

NFSM ≈ DFSM

- Each DFSM is clearly NFSM
  - Just make result of transition into relation
- Other direction uses sets of states
- Define $eps(q) = \{ q' \in K \mid (q,\varepsilon) \vdash^* (q',\varepsilon) \}$
  - All states reachable via ε-moves from q

NFSM ⇒ DFSM

- Let $M = (K, \Sigma, \Delta, s, A_N)$ be an NFSM.
- Construct DFSM $M' = (K', \Sigma, \delta_D, eps(s), A_D)$ where
  - $K' = P(K)$
  - $\delta_D(Q,c) = \cup\{eps(p) \mid \exists q \in Q.\ (q, c, p) \in \Delta\}$ for $Q \in P(K)$
  - $A_D = \{R \subseteq K \mid R \cap A_N \neq \emptyset\}$, i.e., R has a final state
- Show $L(M) = L(M')$

Example

Proof

- Lemma: Let $w \in \Sigma^*$, $p, q \in K$, $P \in K'$. Then
  $(q,w) \vdash_M^* (p,\varepsilon)$ iff $(eps(q), w) \vdash_{M'}^* (P,\varepsilon)$ & $p \in P$
- Assume lemma. Then
  $w \in L(M)$ iff $(s,w) \vdash_M^* (p,\varepsilon)$ for $p \in A_N$
  iff $(eps(s),w) \vdash_{M'}^* (P,\varepsilon)$ for $p \in P$, $p \in A_N$
  iff $(eps(s),w) \vdash_{M'}^* (P,\varepsilon)$ where $P \in A_D$
  iff $w \in L(M')$
- Now prove lemma by induction on $|w|$

iff by Lemma

by def of A_D

Base cases

- Show
  $(q,w) \vdash_M^* (p,\varepsilon)$ iff $(eps(q), w) \vdash_{M'}^* (P,\varepsilon)$ & $p \in P$
  - By induction on length of w.
- Let $|w| = 0$. Thus $w = \varepsilon$
  - ($\Rightarrow$) Suppose $(q, \varepsilon) \vdash_M^* (p,\varepsilon)$. Then $p \in eps(q)$.
    - Thus $(eps(q), \varepsilon) \vdash_{M'}^* (eps(q),\varepsilon)$ & $p \in eps(q)$. So let $P = eps(q)$. ✔
  - ($\Leftarrow$) Suppose $(eps(q), \varepsilon) \vdash_{M'}^* (P,\varepsilon)$ & $p \in P$.
    - Then P must be $eps(q)$, and by def of *eps*,
      $p \in P$ implies $(q, \varepsilon) \vdash_M^* (p,\varepsilon)$ ✔

Induction case

Show $(q,w) \vdash_M^* (p,\varepsilon)$ iff $(eps(q), w) \vdash_{M'}^* (P,\varepsilon)$ & $p \in P$

- Suppose true for v s.t. $|v| = n$. Let $w = za$ for z s.t. $|z| = n$

($\Rightarrow$) Suppose $(q,za) \vdash_M^* (p,\varepsilon)$ where
$(q,za) \vdash_M^* (p',a)$ & $(p',a) \vdash_M (p'',\varepsilon)$ & $(p'', \varepsilon) \vdash_M (p,\varepsilon)$

Therefore $(q,z) \vdash_M^* (p', \varepsilon)$ & $p \in eps(p'')$
By induction $\exists P$ s.t. $(eps(q), z) \vdash_{M'}^* (P',\varepsilon)$ & $p' \in P'$ &
thus $(eps(q), za) \vdash_{M'}^* (P', a)$ & $p' \in P'$
By def of M', $(P', a) \vdash_{M'} (P, \varepsilon)$ for
$P = \cup\{eps(r) \mid \exists q \in Q.\ ((q, a), r) \in \Delta\}$
By above, $((p',a), p'') \in \Delta$ & $p \in eps(p'')$. Therefore $p \in P$

Thus $(eps(q), za) \vdash_{M'}^* (P,\varepsilon)$ for $p \in P$. ✔

Closure Revisited

- Union:
  - Make sets of states disjoint, add new start w/ε moves to starts of original. Final states union of original finals
- Concatenation:
  - From each final state of first, add ε move to start of second. Final states are only those of second.

Exercise

- If L is regular, show that L* is regular.

Minimizing FSM

- Useful for implementing in hardware
- Given regular L, is there a minimal FSM accepting it?
- Is it unique?
- Can we construct it?

Equivalence relations

- $\approx$ is equivalence class iff reflexive, symmetric, transitive.
- $\approx$ is right regular iff $x \approx y \Rightarrow xa \approx ya$ for all $a \in \Sigma$
- Ex. Let M be FSM over Σ. Then define $x \approx_M y$ iff $\delta_M(s,x) = \delta_M(s,y)$. $\approx$ is right-regular
- Equivalence class: $[w] = \{w' \in \Sigma^* \mid w \approx w'\}$
  - In example, equiv class is all w going to same state q.
- Then L(M) is union of equivalence classes.

Minimizing FSM

- Def: x, y are *indistinguishable* wrt L, $x \approx_L y$ iff for all $z \in \Sigma^*$, either both $xz, yz \in L$ or neither is
  - Ex: if $L = \{w \in \Sigma^* \mid w$ does not contain aab as a substring$\}$ then a and baba indistinguishable, but a and ab not.
- $\approx_L$ is right regular equivalence relation
- States of new minimal machine will be equivalence classes: $[w] = \{v \in \Sigma^* \mid v \approx_L w\}$

Example equiv classes

Observations

- No equiv class contains both $u \in L$ and $v \notin L$.
- If strings go to dead state, then all in same class
- More than one equiv class can contain elts of L
- If M is DFSM & q is state, then all strings going to state q are in same equiv. class
- If $L = \mathcal{L}(M)$ then
  # equiv classes of L ≤ # states of M
  - Thus, if L is regular, then # equiv classes of L is finite.

Non-Regular Languages

- Some language have ∞ # of equiv classes
  - $P = \{ww^R \mid w \in \{a,b\}^*\}$
  - [b], [ab], [aab], [aaab], ... all distinct
  - Thus P not regular.

Construct Minimal DFSM

Theorem: Let L be a regular language over some alphabet Σ. Then there is a DFSM M that accepts L and that has precisely n states where n is the number of equivalence classes of L. Any other FSM that accepts L must either have more states than M or it must be equivalent to M except for state names.

But how do we find equivalence classes?
See homework!

Proof

Let $M = (K, \Sigma, \delta, s, A)$, where:

- K consists of the n equivalence classes of L.
- $s = [\varepsilon]$, the equivalence class of ε under L.
- $A = \{[x] : x \in L\}$. *Well-defined!*
- $\delta([x], a) = [xa]$. *Well-defined because right regular.*

Show $L = \mathcal{L}(M)$ and unique minimal

Example