Lecture 3: 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

Homework

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

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

- 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
  - Always includes 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 $w = \varepsilon$
  - (⇒) 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 $P = eps(q)$. ✔
  - (⇐) 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 $(\textit{eps}(q), w) \vdash_{M'}^* (P,\varepsilon)$ & $p \in P$

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

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

Other direction left as exercise!

Converting to Deterministic

- Algorithm exactly as have defined in proof.
- See algorithm in text.

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