Lecture 4: Minimizing 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

New Homework

- Now available on line
  - Turn in single pdf file to gradeScope
- New pairs in Piazza by tonight

Equivalence relations

- $\approx$ is equivalence relation 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], ... from P 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

Proof

Lemma: $\forall u,v \in \Sigma^*$, $([\varepsilon], uv) \vdash_M^* ([u], v)$.

Use lemma to finish proof.

By lemma, $([\varepsilon], w) \vdash_M^* ([w], \varepsilon)$ because $w = w\varepsilon$

Thus $w \in L(M)$ iff $[w] \in A$ iff $w \in L$.

Therefore $L(M) = L$.

Can't be smaller machine. Unique as well.

$x \approx_M y \Rightarrow x \approx y$

Lemma: $\forall u,v \in \Sigma^*$, $([\varepsilon], uv) \vdash_M^* ([u], v)$

Proof by Induction on length of u:
Clearly true for $u = \varepsilon$

S'pose $([\varepsilon], uv) \vdash_M^* ([u], v)$ for $|u| = k$.
Prove k+1: S'pose $u = yc$ where $|y| = k$, $c \varepsilon \Sigma$.

Then $([\varepsilon], ycv) \vdash_M^* ([y], cv)$ by induction
But $([y], cv) \vdash_M ([yc], v)$ by def of δ
So $([\varepsilon], ycv) \vdash_M^* ([yc], v)$. ✔

Myhill-Nerode Theorem

- A language is regular if and only if it is the union of some of the equivalence classes of a right regular equivalence relation with finitely many equivalence classes.

Regular Expressions

- Language of regular expressions over Σ:
  - The symbols ε, $\underline{\emptyset}$, and $\underline{a}$ for $a \in \Sigma$ are regular expressions.
    - Use underlines to distinguish the regular expression symbols from their other uses.
  - If α is a regular expression, so is (α^*).
  - If α and β are regular expressions, so are $(\alpha\ \beta)$ and $(\alpha \cup \beta)$.
    - Text also uses (α^+), we'll define $(\alpha^+) = (\alpha\ (\alpha^*))$
    - Often drop ( ) when clear how to reconstruct

Neither are $\underline{a}$, $\underline{\emptyset}$, …!

Examples

- ($\underline{a} \cup \underline{\varepsilon}$) -- means optional a
  - also written $\underline{a}$?
- Regular expressions used in Bash:
  - http://tldp.org/LDP/Bash-Beginners-Guide/html/sect_04_01.html
  - [0-9]+(\.[0-9]*)? | \.[0-9]+
    - decimal numbers

RegExp in Unix

Operator	**Effect**
.	Matches any single character.
?	The preceding item is optional and will be matched, at most, once.
*	The preceding item will be matched zero or more times.
+	The preceding item will be matched one or more times.
{N}	The preceding item is matched exactly N times.
{N,}	The preceding item is matched N or more times.
{N,M}	The preceding item is matched at least N times, but not more than M times.
-	represents the range if it's not first or last in a list or the ending point of a range in a list.
^	Matches the empty string at the beginning of a line; also characters not in the range of a list
$	Matches the empty string at the end of a line.
\b	Matches the empty string at the edge of a word.
\B	Matches the empty string provided it's not at the edge of a word.
\<	Match the empty string at the beginning of word.

Cheating on Crossword Puzzles!

- grep '\<c...h\>' /usr/share/dict/words
  - returns all 5-letter words in dictionary that start with c and end with h.
  - \< matches empty string at beginning of line
  - \> matches empty string at end of line

Interpreting Regular Expressions

- For each regular expression we define language it denotes:
  - $L(\underline{\varepsilon}) = \{\varepsilon\}$, $L(\underline{\emptyset}) = \emptyset$, and $L(\underline{a}) = \{a\}$ for all $a \in \Sigma$
  - $L(\alpha^*) = (L(\alpha))^* = \{w_1 \ldots w_n \mid n \geq 0, w_i \in L(\alpha)\}$
  - $L(\alpha\beta) = L(\alpha)L(\beta)$ and $L(\alpha \cup \beta) = L(\alpha) \cup L(\beta)$.
- If $\Sigma = \{a,b\}$, what is $L(\underline{aba})$?
  What is $L((\underline{a} \cup \underline{b})^*\underline{aab}(\underline{a} \cup \underline{b})^*)$?
  Is $L((\underline{a} \cup \underline{b})^*) = L(\underline{a}^* \cup \underline{b}^*)$?

Regular Expressions = Regular Languages

- If α is a regular expression then there is a DFM M s.t. $L(\alpha) = L(M)$
  - Construct NDFM by induction on regular expressions
    - Base case of single symbols easy
    - union, concatenation, * simple
- Other direction trickier. Add new start & final. Tear apart NDFM by removing states and writing labels as regular expressions.

Creating RegExp from DFM

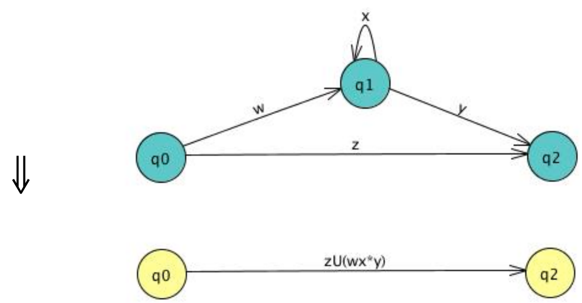
- Get rid of unreachable states
- Add new start state s' w/ε-move to original s
- Add new accepting state f' w/ε-move from original accepting states.
  - Make originals non-accepting
- Now one start and one accepting with no transitions to start or from accepting.
  - If original had that then no need to change.

Creating RegExp from DFM

- Change transitions so
  - exactly one transition from all $q \neq f'$ to every state but s'
  - exactly one transition into all $q \neq s'$ from all states but f'
    - No transition from new final, none to new start, but all others!
- How?
  - If no transitions, add one with $\emptyset$ label
  - If more than one, group using reg exp. involving $\cup$

Remove intermediate states

- Remove states one at a time until only s', f'
  - When remove states, change labels to reg exp.



Done when down to s', f'