CS 378 – (Introduction to) Theory of Computation

Professor Eric Aaron

Lecture – T R 2:30pm

Lecture Meeting Location: It’s complicated…

Business

• Full PS3 out today, due Apr. 1 (see assignment sheet for details)

• Read Sipser, pages 63-69, 77-82

• Additional reading to be posted: A few pages from…
  – Introduction to Automata Theory, Languages, and Computation by Hopcroft, Motwani, and Ullman
  – I’ll refer to this text as HMU in lecture notes sometimes
  – These will be available on the course Lecture Notes page under today’s date
    • Recall that a Colby IP address is needed to access this material
Our Goal:

• Theorem:
  The languages represented by regular expressions are exactly the regular languages(!)
  – Theorem 1: For every regular expression r, there exists a corresponding FA
    • (an NFA, actually, but that’s not important)
  – Theorem 2: For every DFA M, there exists a corresponding regular expression

Theorem 2

• For all DFA M, there exists a corresponding regular expression
• Pf: build it step by step (dynamic programming)
  – Use construct $L_{ij}^k$ to stand for the language of strings that drive M from state i to state j while going through no state higher than k
    • Be sure to start numbering states with 1, not 0!
  – Recursive definition:
    • What are all the $L_{ij}^0$ when k = 0?
    • When k ≠ 0, recursion based on case analysis:
      Strings either go through state k, or they don’t…
    • Inductively build the set of accepted strings of M as a reg. exp.
Theorem 2, pt. 2

- For DFA $M = (Q, \Sigma, \delta, q_0, F)$, a regular expression:
  - Pf: build it step by step (*dynamic programming*)
    - Use construct $L_{ij}^k$ to stand for the language of strings that drive $M$ from state $i$ to state $j$ while going through no state higher than $k$
    - Be sure to start numbering states with 1, not 0! ($|Q| = n$)
    - Recursive definition:
      - When $k = 0$:
        - When $i \neq j$: $L_{ij}^0 = \{ \sigma | \delta(q_i, \sigma) = q_j \}$
        - When $i = j$: $L_{ij}^0 = \{ \epsilon \} \cup \{ \sigma | \delta(q_i, \sigma) = q_j \}$
      - When $k > 0$:
        - Then, $L(M) = \bigcup_{i \in F} L_{1i}$

See HMU, pg. 93-97
State Minimization
(“Is that the best you can do?”)

• Next stop: State Minimization

(but first…)

DISTINGUISHABILITY STRIKES BACK

• Generalized Distinguishability Lemma
  – Let L be a language (not nec’ly regular). If there exists a pairwise distinguishable set S w.r.t. L s.t. |S| = k then…

(Note: S isn’t infinite!! It’s not the same as the previous Distinguishability Lemma!!)
Generalized Distinguishability Lemma

- Generalized Distinguishability Lemma
  - Let $L$ be a language (not necessarily regular). If there exists a pairwise distinguishable set $S$ w.r.t. $L$ s.t. $|S| = k$

  then any DFA that recognizes $L$ must have at least $k$ states.

- Pf: Pigeonhole! Imagine a machine recognizing $L$ that has fewer than $k$ states. Then...

So Minimize This

- An example: A machine (DFA) $M$ that recognizes $L = \{w \mid w \in \{0,1\}^* \text{ s.t. third digit from right is a 1} \}$

- One way, $M$ has 8 states – can we do better?
  - (pairwise-distinguishable sets, anyone?)
  - How many strings do we need to test to prove that our set is pairwise distinguishable?
How to Minimize a DFA (or, THE RETURN OF DISTINGUISHABILITY, with special Dynamic Programming)

• Defn: (in)distinguishable states
  – Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA.
  – Two reachable states $q, r$ of $M$ are said to be indistinguishable states if for all $w \in \Sigma^*$,
    • $w$ drives both $q$ and $r$ to non-final states, OR
    • $w$ dries both $q$ and $r$ to final states.
    • (Otherwise, $q$ and $r$ are distinguishable states.)

Note: These aren’t distinguishable words, they’re distinguishable states! Those concepts are related to each other, but they aren’t the same—please be sure to … well, to distinguish them!

Minimize This, Claim 1

• Claim 1: If $q, r$ are indistinguishable states in $M$, we can remove one of them.
  – Either one! Arbitrarily! Doesn’t matter! (Why?)

  – (But how do we determine that two states are indistinguishable? Can it be done in finite time? Or do we need to test all words in $\Sigma^*$?)
Minimize This, Claim 2

- Claim 2: If $M = (Q, \Sigma, \delta, q_0, F)$ and $|Q| = n$, then to test whether two states are distinguishable, it suffices to test all strings of length $n(n-1)/2$.
  - Pf idea: First, convince ourselves we need only see strings of size $\leq n^2$ by a pigeonhole argument; then notice that we really only need size $\leq \binom{n}{2}$
    - But how?

- And, even then… a running time of $|\Sigma|^{|n(n-1)/2|}$?!!!

Shrink This… Runtime

- Make a more efficient procedure!
- Observation
  - If $q, r$ are distinguishable states, they are either
    1. Distinguished by $w$ s.t. $|w| = 0$, or
    2. Distinguished by $w$ s.t. $|w| \geq 1$. In this case, there exists some $q_k, q_l$ in $Q$ that are distinguished by some string $x$, where $|x| = |w| - 1!$
Implement That: An Efficient, Dynamic Programming Algorithm

- To “optimize” a DFA $M = (Q, \Sigma, \delta, q_0, F)$
  - First, remove all unreachable states
  - (We say a state $q_i$ is unreachable when there is no path from $q_0$ to
    $q_i$ in $M$—i.e., when there is no word that drives $M$ from $q_0$ to
    $q_i$.)

It’s easy to overlook this step, and some presentations of the procedure may not
mention it, but please be sure to remove all easily-found unreachable states (i.e., states
for which there is no path from $q_0$ to them) when state-minimizing DFAs for CS378!

See HMU,
pg. 155-165

Implement That: An Efficient, Dynamic Programming Algorithm

- To “optimize” a DFA $M = (Q, \Sigma, \delta, q_0, F)$
  - First, remove all unreachable states
  - Then build a table to identify distinguishable pairs of states
    - $(n \choose 2)$ cells in the table, one for each pair of (reachable) states
  - To be efficient, use dynamic programming
    - First: is each pair distinguished by $\epsilon$
    - Then iterate: after k’th iteration, assume we’ve identified all pairs
distinguished by strings of length k or less
      - How to find pairs distinguished by strings of length k+1?
      - Observation: if $q$, $r$ distinguished by string of length k+1, there must be
        states $q'$, $r'$ distinguishable by string of length k…
    - Mark table with # of iteration on which a pair is distinguished

HMU marks the table with x’s not numbers—please use numbers for your work in CS378!

- Whatever’s unmarked: indistinguishable pairs (see claim 1)

(What’s the complexity of this?)

See HMU,
pg. 155-165