CS 378 – (Introduction to) Theory of Computation

Professor Eric Aaron

Lecture – T R 2:30pm

Lecture Meeting Location: It’s complicated…

Business

• PS2-lookahead out
  – More exercises will be added
  – Full assignment and deadline TBA soon, but submission deadline definitely not before 3/11

• Read Sipser, Ch. 0, 1.1, 1.2

• Small Assignment assigned today, due by beginning of next class meeting (as usual)
Formal definition of an NFA

• A NFA is a 5-tuple \((Q, \Sigma, \delta, q_0, F)\), where:
  – \(Q\) is a finite set of states
  – \(\Sigma\) is a finite alphabet
  – \(\delta\) is a transition function from \(Q \times \Sigma \epsilon\) to \(P(Q)\)
  – \(q_0\) is a designated start state \((q_0 \in Q)\)
  – \(F\) is a designated set of accept states \((F \subseteq Q)\)

• Formal definition of accepting a word
  – A lot like for DFAs (cf. Sipser pg. 54 and Sipser pg. 40)
  – Difference: in NFAs, states in sequence are elements of \(\delta\)-function result, not identical to it

Equivalence of NFAs and DFAs (“Sometimes the magic works…”)

• SURPRISE! Nondeterminism, in a significant sense, adds NOTHING to the class of DFAs

• **Theorem**: Every nondeterministic finite automaton has an equivalent deterministic finite automaton
  – Proof: A *power-set* construction
    • Given NFA \(N\) recognizing language \(A\), construct DFA \(M\) recognizing \(A\)
The NFA -> DFA construction

• Let NFA $N = (Q, \Sigma, \delta, q_0, F)$
  – Assume $N$ has no $\epsilon$-moves
  – DFA $M = (P(Q), \Sigma, \delta', \{q_0\}, F')$
    • where $\delta'$ maps sets of $Q$ to sets of $Q$ on input $a$
      – induced from $\delta$ in the expected way…
      – i.e., for $R \in P(Q)$ and $a \in \Sigma$, $\delta'(R,a) = \{q:Q \mid \exists r:R. q \in \delta(r,a)\}$
    • and $F'$ is any subset of $P(Q)$ containing a state in $F$

  Why is this $\{q_0\}$ instead of $q_0$?

This definition of $\delta'$ is notation-heavy, but the ideas are more straightforward than the notation might suggest! Please spend time with it until it makes total sense to you!

• Think of it as a power-set construction
  – The DFA works on sets of the NFA’s states

The NFA -> DFA construction, pt. 2

• Now consider the case where $N$ has $\epsilon$-moves
• Let NFA $N = (Q, \Sigma, \delta, q_0, F)$
  – Old DFA $M = (P(Q), \Sigma, \delta', \{q_0\}, F')$ not quite right
    • Need to consider epsilon closure for $\delta'$
  – For $R$ subset of $Q$, let $E(R) = \{ q \mid q$ can be reached from $R$ by traveling along $0$ or more $\epsilon$-arrows$\}$
  – Equivalent DFA: $(P(Q), \Sigma, \delta'', E(\{q_0\}), F')$
    • where $\delta''$ maps (sets of $Q$) to $E$(sets of $Q$) on input $a$
      – induced from $\delta$ in the expected way
      – i.e., for $R \in P(Q)$ and $a \in \Sigma$, $\delta''(R,a) = \{q:Q \mid \exists r:R. q \in E(\delta(r,a))\}$
    • and $F'$ is any subset of $P(Q)$ containing a state in $F$

The epsilon-closure in $\delta''$ is the only difference from $\delta'$ on the previous slide!

• Still a power-set construction
  – The DFA works on sets of the NFA’s states, extended to include epsilon-moves
Small Assignment, 03/04/21

• Exercise 1.16(b) from Sipser, page 86.

You do not need to explain your answer, just give the state-transition diagram for your DFA. (I’ll be happy to go over the exercise in class, if you’d like!)

The exercise in Sipser says to use the construction given in Theorem 1.39—that is the same construction as the one presented in lecture notes today.

Lemmas

• Lemma 1: If $L$ is accepted by some DFA, then it is recognized by some NFA

• Lemma 2: If $L$ is accepted by some NFA, then it is recognized by some DFA

(one of these is trivial!)
NFAs and regular languages

• *Corollary*: A language is regular iff some NFA recognizes it