CIT 596 Recitation Notes - Spring 2014

Honglin Zhang

Jan 30, 2014

Book List

Introduction to the Theory of Computation, Third Edition [@sipser13].

Week 1 – Jan 24, 2014

1. A finite automaton is a quin-tuple $(Q, \Sigma, \delta, q_0, F)$ where

    - Q is a finite set called the states (why finite?)
    - Σ is a finite set called the alphabet (advantage of finite? Induction)
    - $\delta \times \Sigma \to Q$ transition function
    - $q_0 \in Q$ start state
    - $F \subseteq Q$ set of final states (aka accept states) (asymmetric to starte state)

2. A language is called a regular language if some finite automaton recognizes it.

3. The set of regular languages is closed under such operations

    - $A \cup B = \{x | x \in A \text{ or } x \in B\}$
    - $A \circ B = \{xy | x \in A \text{ and } y \in B\}$
    - $A^* = \{x_1, \ldots, x_k | k \geq 0 \text{ and } x_i \in A_i\}$

4. Briefly sketch the idea of DFA construction for union operation. Then illustrate by example the case for intersection. Also, point out the difference between union case and intersection case.

    - union case
        - $F = \{(r_1, r_2) | r_1 \in F_1 \text{ or } r_2 \in F_2\} = (F_1 \times Q_2) \times (Q_1 \times F2)$
    - intersection case

- $F = \{(r_1, r_2) | r_1 \in F_1 \text{ and } r_2 \in F_2\} = F_1 \times F_2$

5. [Exercise] @sipser13 [p. 83] exercise 1.4 a.

    - This is beautiful! What are the final states? What about the union case?

6. [Exercise] @sipser13 [p. 83] exercise 1.4 e.

7. The reverse question: if a set of languages is closed under such operations, is that the set of regular languages we defined? Come back to this after NFA and regular expression.

Week 2 – Jan 31, 2014

1. The difference between DFA and NFA

    - In NFA, a state may have zero, one or many existing arrows
    - In NFA, labels may contain members of the alphabet or ϵ
    - NFA may die (why? because δ allows the mapping to $\emptyset$)

2. Formal definition of NFA, quin-tuple $(Q, \Sigma, \delta, q_0, F)$

    - Q a finite set of the states
    - Σ a finite alphabet
    - $\delta : Q \times (\Sigma \cup \{\epsilon\}) \to 2^Q$ transition function
    - $q_0 \in Q$ is the start state
    - $F \subseteq Q$ is the set of accept states

3. Every NFA has an equivalent DFA. First sketch the basic idea. Then show by examples.

    - Observe the transition function defined for NFA. To make it look prettier, we lift $\delta : Q \times (\Sigma \cup \{\epsilon\}) \to 2^Q$ to $\hat{\delta} : 2^Q \times (\Sigma \cup \{\epsilon\}) \to 2^Q, \hat{\delta}(R, a) = \cup_{r \in R} \delta(r, a)$. We say $\hat{\delta}$ is induced by δ. Then rename the space 2^Q to Q'.
    - Some other chores, start state and final states
    - Handle epsilon-transition. Treat states connected by epsilon transitions "equivalently", but btw epsilon-transition is not an equivalent relation (Why? Reflexivity, symmetry, transition?). When dealing with some state, aggregate all states attached to this one by epsilon relation. Namely, replace $\delta(r, a)$ by $E(\delta(r, a))$.

4. [Exercise] @sipser13 [p. 86] exercise 1.16 (a)
5. NFA accepts it $\Leftrightarrow$ L is regular (Why?)

    - L is regular iff there is a DFA accepting it
    - NFA can be converted to a DFA

- Is that sufficient? No, we also need to know that DFA can be converted to an NFA (but this is trivial)
- Then there exists a DFA iff there exists an NFA
- BTW is iff an equivalence relation? No, not reflexive.
- A direct corollary of the previous one

6. The set of regular languages is closed under union operation. Show by example

    - $\forall L_1, L_2 \in \mathcal{L}, L_1 \cup L_2 \in \mathcal{L}$

7. [Exercise] @sipser13 [p. 85] exercise 1.9 (b)

    - The set of regular languages is closed under concatenation.
    - $\forall L_1, L_2 \in \mathcal{L}, L_1 L_2 \in \mathcal{L}$
    - Why should the example reject every string? Suppose not. Then let's assume it accepts string w. Then there must exist a partition $w = xy$ such that x is accepted by the first one, and y accepted by the second one. Contradiction!

Week 3 – Feb 7, 2014

1. How to derive epsilon closure?

    - Say programmatically. Step by step with BFS or DFS. (pseudo code?)
    - Look this up right now if you are not familiar with these concepts.

2. [Exercise] @sipser13 [p. 85] exercise 1.14
3. [Exercise] @sipser13 [p. 86] exercise 1.19
4. [Exercise] @sipser13 [p. 86] exercise 1.20
5. [Exercise] @sipser13 [p. 87] exercise 1.22
6. [Exercise] @sipser13 [p. 88] exercise 1.28
7. [Exercise] @sipser13 [p. 90] exercise 1.45
8. [Exercise] @sipser13 [p. 92] exercise 1.57
9. [Exercise] @sipser13 [p. 93] exercise 1.71