Assignment 1

Adam Young

January 13, 2022

1 Summarize selected parts of chapters 0 and 1

1.1 THEOREM 0.20

For any two sets A and B, $\overline{A \cup B} = \overline{A} \cap \overline{B}$

1.2 THEOREM 0.21

For every graph G, the sum of the degrees of all the nodes in G is an even number.

1.3 THEOREM 0.22

For each number n greater than 2, there exists a 3-regular graph with n nodes.

1.4 THEOREM 0.24

$\sqrt{2}$ is irrational.

1.5 THEOREM 0.25

For each $t \geq 0$,

$$P_0 = PM^0 - Y\left(\frac{M^k - 1}{M - 1}\right)$$

1.6 DEFINITION 1.5

A **finite automation** is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is a finite set called **states**,
2. Σ is a finite set called the **alphabet**,
3. $\delta : Q \times \Sigma \to Q$ is the **transition function**,

4. $q_0 \in Q$ is the **start state**, and
5. $F \subseteq Q$ is the **set of accept states**.

1.7 DEFINITION 1.16

A language is called a **regular language** if some finite automation recognizes it.

1.8 DEFINITION 1.23

Let A and B be languages. We define the regular operations ***union***, ***concatenation***, and ***star*** as follows:

- **Union**: $A \cup B = \{x | x \in A \text{ or } x \in B\}$.
- **Concatenation**: $A \circ B = \{xy | x \in A \text{ and } y \in B\}$.
- **Star**: $A^* = \{x_1 x_2 ... x_k | k \geq 0 \text{ and each } x_i \in A\}$.

1.9 THEOREM 1.25

The class of regular languages is closed under the union operation.

In other words, if A_1 and A_2 are regular languages, so is $A_1 \cup A_2$

1.10 THEOREM 1.26

The class of regular languages is closed under the concatenation operation.

In other words, if A_1 and A_2 are regular languages, so is $A_1 \circ A_2$

1.11 DEFINITION 1.37

A **nondeterministic finite automation** is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is a finite set called states,
2. Σ is a finite set called the alphabet,
3. $\delta : Q \times \Sigma_\varepsilon \to P(Q)$ is the transition function,
4. $q_0 \in Q$ is the start state, and
5. $F \subseteq Q$ is the set of accept states.

1.12 THEOREM 1.39

Every nondeterministic finite automation has an equivalent deterministic finite automation.

1.13 COROLLARY 1.40

A language is regular if and only if some nondeterministic finite automation recognizes it.

One direction of the "if and only if" condition states that a language is regular if some NFA recognizes it. Theorem 1.39 shows that any NFA can be converted into an equivalent DFA. Consequently, if an NFA recognizes some language , so does some DFA, and hence the language is regular. The other direction of the "if and only if" condition states that a language is regular only if some NFA recognizes it. That is, if a language is regular only if some nFA recognizes it. That is, if a language is regular, some NFA must be recognizing it. Obviously, this condition is true because a regular language has a DFA recognizing it and an DFA is also an NFA.

1.14 THEOREM 1.45

The class of regular languages is closed under the union operation.

1.15 THEOREM 1.47

The class of regular languages is closed under the concatenation operation.

1.16 THEOREM 1.49

The class of regular languages is closed under the star operation.

1.17 DEFINITION 1.52

Say that R is a ***regular expression*** if R is

1. a for some a in the alphabet Σ,
2. ε,
3. $\emptyset$,
4. $(R_1 \cup R_2)$, where R_1 and R_2 are regular expressions,
5. $(R_1 \circ R_2)$, where R_1 and R_2 are regular expressions, or
6. $(R_1^* \cup R_2)$, where R_1 is a regular expression.

In items 1 and 2, the regular expressions a and ε represent the languages $\{a\}$ and $\{\varepsilon\}$, respectively. In item 3, the regular expression $\emptyset$ represents the empty language. In items 4, 5, and 6, the expressions represent the languages obtained by taking the union or concatenation of the languages R_1 and R_2 , or the star of the language R_1, respectively.

1.18 THEOREM 1.54

A language is regular if and only if some regular expression describes it.

This theorem has two directions. We state and prove each direction as a separate lemma.

1.19 LEMMA 1.55

If a language is described by a regular expression, this it is regular.

1.20 LEMMA 1.60

If a language is regular, then it is described by a regular expression.

1.21 DEFINITION 1.64

A **generalized nondeterministic finite automation** is a 5-tuple $(Q, \Sigma, \delta, q_{start}, q_{accept})$, where

1. Q is the finite set of states,
2. Σ is the input alphabet,
3. $\delta : (Q - q_{accept}) \times (Q - q_{start}) \to \mathcal{R})$ is the transition function,
4. q_{start} is the start state, and
5. q_{accept} is the set of accept states.

1.22 CLAIM 1.65

For any GNFA G, CONVERT(G) is equivalent to G.

We prove this claim by induction on k, the number of states of the GNFA.

Basis: Prove the claim true for k = 2 states. If G has only two states, it can have only a single arrow, which goes from the start state to the accept state. The regular expression label on this arrow describes all the strings that allow G to get to the accept state. Hence this expression is equivalent to G.

Induction step: Assume that the claim is true for k-1 states and use this assumption to prove that the claim is true for k states. First we show that G and G' recognize the same language. Suppose that G accepts an input w. Then in an accepting branch of the computation, G enters a sequence of states:

$$qstart, q1, q2, q3, ..., qaccept.$$

If none of them is the removed state q_{rip}, clearly G' also accepts w. The reasonis that each of the new regular expressions labeling the arrows of G0 contains the old regular expression as part of a union. If qrip does appear, removing each run of consecutive qrip states forms an accepting computation for G0 . The states qi and qj bracketing a run have a new regular expression on the arrow between them that describes all strings taking qi to qj via qrip on G. So G0 accepts w. Conversely, suppose that G0 accepts an input w. As each arrow between any two states qi and qj in G0 describes the collection of strings taking qi to qj in G, either directly or via qrip, G must also accept w. Thus G and G0 are equivalent. The induction hypothesis states that when the algorithm calls itself recursively on input G0 , the result is a regular expression that is equivalent to G0 because G0 has k-1 states. Hence this regular expression also is equivalent to G, and the algorithm is proved correct. This concludes the proof of Claim 1.65, Lemma 1.60, and Theorem 1.54.

1.23 THEOREM 1.70

2 Sipser Exercises

2.1 Question 0.2

a. $S = \{1, 10, 100\}$

b. $S = \{x | x \in Z \text{ and } x > 5\}$

c. $S = \{x | x \in N \text{ and } x < 5\}$

d. $S = \{aba\}$

e. $S = \{\varepsilon\}$

f. $S = \emptyset$

2.2 Question 0.5

Set C is set with c elements how many elements in power set of C?

Answer: $|P(C)| = 2^c$

Work:

- $|C| = 1, C = \{1\}, P(C) = (\{\}, \{1\}), |P(C)| = 2 = 2^1$
- $|C| = 2, C = \{1, 2\}, P(C) = \{\{\}, \{1\}, \{2\}, \{1, 2\}\} = 4 = 2^2$
- $|C| = 3, C = \{1, 2, 3\}, P(C) = \{\{\}, \{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}\} = 8 = 2^3$
- We can see that as $|C|$ grows by 1, then $|P(C)|$ grows 2 to the power of the number of elements.
- When $|C| = c, C = \{1, 2, 3, ..., c\}$, then $|P(C)| = 2^c$

2.3 Question 0.6

a. $f(2) = 7$

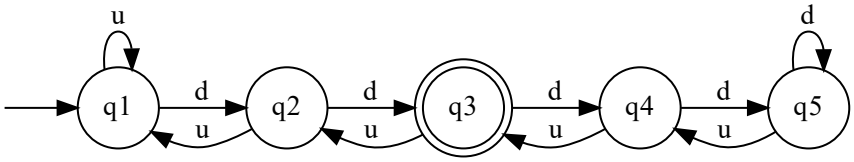
b. domain $f = \{1, 2, 3, 4, 5\}$, range $f = \{6, 7\}$

c. $g(2, 10) = 6$

d. domain $G = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$, range $G = \{6, 7, 8, 9, 10\}$

e. $g(4, f(4)) = g(4, 7) = 8$

2.4 1.3



3 Working with sets

Submitted via https://baylor.kattis.com/courses/CSI5310/22s.