Theory of Computation
REGULAR OPERATIONS

Md. Jakaria

Lecturer
Department of Computer Science & Engineering
Military Institute of Science and Technology

July 23, 2019

Regular Operation

- In arithmetic, the basic objects are numbers and the tools are operations such as + and x.
- In TOC, the objects are languages and tool includes operations specifically designed for them.
- We define three operations on languages, called the regular operations.

Regular Operation

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**: $A^* = \{x_1 x_2 \ldots x_k |\ k \geq 0 \text{ and each } x_i \in A\}$.

Regular Operation

- The union operation simply takes all the strings in both A and B and lumps them together into one language.
- The concatenation operation attaches a string from A in front of a string from B in all possible ways to get the strings in the new language.
- The star operation is a unary operation instead of a binary operation. It works by attaching any number of strings in A together to get a string in the new language.

Regular Operation

EXAMPLE 1.24

Let the alphabet Σ be the standard 26 letters $\{\mathtt{a}, \mathtt{b}, \ldots, \mathtt{z}\}$. If $A = \{\mathtt{good}, \mathtt{bad}\}$ and $B = \{\mathtt{boy}, \mathtt{girl}\}$, then

$$A \cup B = \{\mathtt{good}, \mathtt{bad}, \mathtt{boy}, \mathtt{girl}\},$$

$$A \circ B = \{\mathtt{goodboy}, \mathtt{goodgirl}, \mathtt{badboy}, \mathtt{badgirl}\}, \text{ and}$$

$$A^* = \{\varepsilon, \text{good, bad, goodgood, goodbad, badgood, badbad, goodgoodgood, goodgoodbad, goodbadgood, goodbadbad}, \ldots\}.$$

■

Regular Operation

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 $A_1 \cup A_2$.

Regular Operation

PROOF IDEA We have regular languages A_1 and A_2 and want to show that $A_1 \cup A_2$ also is regular. Because A_1 and A_2 are regular, we know that some finite automaton M_1 recognizes A_1 and some finite automaton M_2 recognizes A_2. To prove that $A_1 \cup A_2$ is regular, we demonstrate a finite automaton, call it M, that recognizes $A_1 \cup A_2$.

Regular Operation

This is a proof by construction. We construct M from M_1 and M_2. Machine M must accept its input exactly when either M_1 or M_2 would accept it in order to recognize the union language. It works by *simulating* both M_1 and M_2 and accepting if either of the simulations accept.

Regular Operation

PROOF

Let M_1 recognize A_1, where $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$, and
M_2 recognize A_2, where $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$.

Construct M to recognize $A_1 \cup A_2$, where $M = (Q, \Sigma, \delta, q_0, F)$.

Regular Operation

1. $Q = \{(r_1, r_2) | \; r_1 \in Q_1 \text{ and } r_2 \in Q_2\}$.
   This set is the ***Cartesian product*** of sets Q_1 and Q_2 and is written $Q_1 \times Q_2$. It is the set of all pairs of states, the first from Q_1 and the second from Q_2.

Regular Operation

1. $Q = \{(r_1, r_2) | \, r_1 \in Q_1 \text{ and } r_2 \in Q_2\}$.
   This set is the ***Cartesian product*** of sets Q_1 and Q_2 and is written $Q_1 \times Q_2$. It is the set of all pairs of states, the first from Q_1 and the second from Q_2.

2. Σ, the alphabet, is the same as in M_1 and M_2. In this theorem and in all subsequent similar theorems, we assume for simplicity that both M_1 and M_2 have the same input alphabet Σ. The theorem remains true if they have different alphabets, Σ_1 and Σ_2. We would then modify the proof to let $\Sigma = \Sigma_1 \cup \Sigma_2$.

Regular Operation

3. δ, the transition function, is defined as follows. For each $(r_1, r_2) \in Q$ and each $a \in \Sigma$, let

$$\delta\big((r_1, r_2), a\big) = \big(\delta_1(r_1, a), \delta_2(r_2, a)\big).$$

Hence δ gets a state of M (which actually is a pair of states from M_1 and M_2), together with an input symbol, and returns M's next state.

Regular Operation

3. δ, the transition function, is defined as follows. For each $(r_1, r_2) \in Q$ and each $a \in \Sigma$, let

$$\delta\big((r_1, r_2), a\big) = \big(\delta_1(r_1, a), \delta_2(r_2, a)\big).$$

Hence δ gets a state of M (which actually is a pair of states from M_1 and M_2), together with an input symbol, and returns M's next state.

4. q_0 is the pair (q_1, q_2).

Regular Operation

5. F is the set of pairs in which either member is an accept state of M_1 or M_2. We can write it as

$$F = \{(r_1, r_2) |\ r_1 \in F_1 \text{ or } r_2 \in F_2\}.$$

This expression is the same as $F = (F_1 \times Q_2) \cup (Q_1 \times F_2)$. (Note that it is *not* the same as $F = F_1 \times F_2$. What would that give us instead?[3])

Regular Operation

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 then so is $A_1 \circ A_2$.

This is the place where non-determinism starts!