



The Regular Operations

Teodor Rus

`rus@cs.uiowa.edu`

The University of Iowa, Department of Computer Science

Introduction

- Once automata and computation have been defined, their properties need be studied.
- This requires appropriate tools and technique which we initiate here.
- We also need tools and techniques for studying non-regular languages, i.e., languages which are beyond the capability of finite automata.

Contrast

- In Arithmetic, the basic objects are numbers and the tools are operations for number manipulation, such as $+$, \times .
- In Theory of Computation the objects are languages and the tools for language manipulation are specifically designed.
- The three common operations on languages, called *regular operations* are:
union (\cup), concatenation (\circ), and star ($*$).

Definition 1.10

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

- **Union:** $A \cup B = \{x \mid x \in A \vee x \in B\}$
- **Concatenation:** $A \circ B = \{xy \mid x \in A \wedge y \in B\}$
- **Star:**
 $A^* = \{x_1x_2 \dots x_k \mid k \geq 0 \wedge x_i \in A, 1 \leq i \leq k\}$

Question

- $A \cup B$, $A \circ B$ and A^* are languages.
- Consequently $A \cup B$, $A \circ B$ and A^* are sets of strings over some alphabets.
- What are their alphabets?

Answer: if Σ_A and Σ_B are the alphabet of $A \cup B$, $A \circ B$, A^* is $\Sigma_A \cup \Sigma_B$.

Reason: Any language over Σ_A or Σ_B is certainly a language over $\Sigma_A \cup \Sigma_B$. Hence, we may assume $\Sigma_A = \Sigma_B$.

Observations

- Union is the usual operation on sets.
- Concatenation is a little trickier; it attaches a string from A in front of a string from B in all possible ways to get strings from $A \circ B$.
- Star operation is different; it applies to one language, i.e., it is unary rather than binary. Star works by attaching any number of strings in A together to get strings in A^* .

Fact

Because “any number” includes 0 , $\epsilon \in A^*$,
no matter what A is.

Example 1.11

Let $\Sigma = \{a, b, \dots, z\}$. If $A = \{good, bad\}$ and $B = \{boy, girl\}$, then:

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

$$A \circ B = \{goodboy, goodgirl, badboy, badgirl\};$$

$$A^* = \{\epsilon, goob, bad, goodgood, goodbad, badgood, badbad, goodgoodgood, goodgoodbad, goodbadgood, goodbadbad, \dots\}.$$

Closed set

If A is a set and $f : A \times A \rightarrow A$ is total, i.e., $\forall a, b \in A [f(a, b) \in A]$ then we say that A is closed under f .

Example: consider $\mathcal{N} = \{1, 2, 3, \dots\}$ and $*, / : \mathcal{N} \times \mathcal{N} \rightarrow \mathcal{N}$ where $*$ is number multiplication and $/$ is number division.

- Since multiplication of natural numbers is total (i.e., over all defined) \mathcal{N} is closed under $*$;
- Since the division of two natural numbers is not always a natural number, for example $1/2 \notin \mathcal{N}$, \mathcal{N} is not closed under division.

Closure of regular languages

We will show that the collection of regular languages is closed under the three regular operations.

This property provides useful tools for manipulating regular languages and for understanding the power of finite automata.

Theorem 1.24

The class of regular languages is closed under union operation. That is, if A_1 and A_2 are regular then $A_1 \cup A_2$ is regular.

Proof idea

- Because A_1 and A_2 are regular there are automata M_1 and M_2 recognizing A_1 and A_2 , respectively.
- To prove that $A_1 \cup A_2$ is regular we construct a finite automaton M from M_1 and M_2 that recognizes $A_1 \cup A_2$.
- The machine M works by simulating M_1 and M_2 .
- **Simulation:** pretend that you are M . As you read the input symbols you simulate both M_1 and M_2 , simultaneously.
- To keep track of both simulations you need to remember the state that each machine would be in if it had read up to this point in the input.

More on proof idea

- To remember the states of M_1 and M_2 you need to remember a pair of states;
- If M_1 has k_1 states and M_2 has k_2 states then you need $k_1 \times k_2$ states to simulate simultaneously M_1 and M_2 ;
- Transitions of M goes from pair to pair, updating the state for both M_1 and M_2 ;
- The start state of M is the pair of start states of M_1 and M_2 ; the accept states of M is the set of pairs containing an accept state of M_1 or M_2 .

Proof

By construction. Let M_1 recognize A_1 where $M_1 = (Q_1, \Sigma, \delta_1, q_0^1, F_1)$, and M_2 recognize A_2 where $M_2 = (Q_2, \Sigma, \delta_2, q_0^2, F_2)$.

Construct M to recognize $A_1 \cup A_2$,

$M = (Q, \Sigma, \delta, q_0, F)$, where:

Construction of M

1. $Q = \{(r_1, r_2) \mid r_1 \in Q_1, r_2 \in Q_2\}$, i.e., $Q = Q_1 \times Q_2$;
2. Σ is assumed the same for M_1 and M_2 . If M_1 and M_2 have different alphabets, Σ_1 and Σ_2 then $\Sigma = \Sigma_1 \cup \Sigma_2$;
3. For each $(r_1, r_2) \in Q$ and $a \in \Sigma$,
 $\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a))$;
4. $q_0 = (q_0^1, q_0^2)$;
5. $F = \{(r_1, r_2) \mid r_1 \in F_1 \vee r_2 \in F_2\}$, i.e., $F = (F_1 \times Q_2) \cup (Q_1 \times F_2)$.
Note, this is not the same as $F_1 \times F_2$.

Corollary

Class of regular languages is closed under intersection.

Proof: For two regular languages A and B , recognized by the automata M_A and M_B the automaton that recognizes the language $A \cap B$ is constructed in the same way as the automaton that recognizes the language $A \cup B$ with the final states defined by $F = \{(r_1, r_2) \mid (r_1, r_2) \in F_1 \times F_2\}$.

Observations

- This construction is fairly simple and thus its correctness is evident from the strategy described by proof idea.
- More complicated constructions require additional discussion to prove correctness.
- A formal correctness proof for a construction of this type usually proceeds by induction. We will illustrate it further.

Theorem 1.26

The class of regular languages is closed under concatenation operation. In other words, if A_1 and A_2 are regular languages then so is $A_1 \circ A_2$.

Proof idea: As before, we can start with finite automata M_1 and M_2 recognizing A_1 and A_2 and construct the automaton M to recognize $A_1 \circ A_2$.

Ideas for construction of M

- Instead of constructing M to accept its input if either M_1 or M_2 accept, M must accept if its input can be broken into two pieces where M_1 accepts first piece and M_2 accepts second piece;
- The problem is that M does not know where to break its inputs;
- To solve this problem we need to introduce a new technique: **the nondeterminism**.