Introduction to the Theory of Computation
Regular Languages

Arthur Nunes-Harwitt

1 DFA Formal Definition

Definition 1. A deterministic finite automaton (DFA), or deterministic finite-state machine, is a five-tuple \((Q, \Sigma, \delta, q_0, F)\), where

- \(Q\) is a finite set of states.
- \(\Sigma\) is a finite set of input symbols.
- \(\delta : Q \times \Sigma \to Q\) is the transition function. This function is the mathematical representation of the arcs in a transition diagram.
- \(q_0 \in Q\) is the start state.
- \(F \subseteq Q\) is the set of accepting states\(^1\).

Note that since \(\delta\) is a total function, this definition is saying that every state must have arcs coming out labeled with each input symbol.

2 Regular Languages

While the DFA definition formally characterizes the structure of the diagram, it says nothing about how to interpret the diagram. For example, how are we allowed to move along transition arcs? We need some more definitions to characterize the proper interpretation. Once we have established how to interpret DFAs, we can reformulate the notion of a computation by a DFA as a kind of language. We can then use the notion of language to compare the languages associated with DFAs to languages associated with other models of computation.

The following definition generalizes the transition function; its second input is a string rather than a symbol. It is this definition that explains how to move along the arcs.

Definition 2. Given a DFA \(M = (Q, \Sigma, \delta, q_0, F)\), \(\delta^* : Q \times \Sigma^* \to Q\) is defined as follows.

- \(\delta^*(q, \varepsilon) = q\) if \(q \in Q\), and
- \(\delta^*(q, x\sigma) = \delta(\delta^*(q, x), \sigma)\) if \(q \in Q, x \in \Sigma^*, \text{ and } \sigma \in \Sigma\).

The next definition gives meaning to the initial state \((q_0)\) and the set of final states \((F)\); it specifies the meaning of the entire DFA five-tuple.

\(^1\)Accepting states are also called final states.
Definition 3. Given a DFA $M = (Q, \Sigma, \delta, q_0, F)$ and a string $x \in \Sigma^*$, $x$ is accepted by $M$ if $\delta^*(q_0, x) \in F$.

We make use of this notion of acceptance to associate a language with any DFA.

Definition 4. Given a DFA $M = (Q, \Sigma, \delta, q_0, F)$, the language accepted by $M$, or the language recognized by $M$, is denoted by $L(M)$, where $L(M) = \{ x \in \Sigma^* \mid x$ is accepted by $M \}$.

Finally, we can circumscribe what any DFA could possibly do by defining the class of all languages associated with DFAs. We can use this class to compare the power of the DFA model of computation to other models.

Definition 5. A language $L$ is regular if there exists a DFA $M$ such that $L = L(M)$.

If another model of computation has the same power as the DFA model, then its associated class would be equal to the class of regular languages; but if it were more powerful than the DFA model, then the class of regular languages would be a proper subset of its associated class.

3 Non-Deterministic Finite Automata

3.1 The Idea of Non-Determinism

There are different notions of non-determinism. We allow multiple transitions with the same label. A non-deterministic machine “automagically” guesses which is the right transition to take. Thus a non-deterministic machine accepts a string if there is a path from the start state to a final state. Because of this power, it is often easier to come up with a non-deterministic machine that accepts a particular language.

3.2 The Formal Definitions

Definition 6. Given an alphabet $\Sigma$, the set $\Sigma_e = \Sigma \cup \{ \epsilon \}$.

Definition 7. A non-deterministic finite automaton (NFA) is a five-tuple $(Q, \Sigma, \delta, q_0, F)$, where

- $Q$ is a finite set of states.
- $\Sigma$ is a finite set of input symbols.
- $\delta : Q \times \Sigma_e \rightarrow \mathcal{P}(Q)$ is the transition function.
- $q_0 \in Q$ is the start state.
- $F \subseteq Q$ is the set of accepting states.

In contrast to DFA, an NFA may have missing arcs.

Definition 8. Given an NFA $N = (Q, \Sigma, \delta, q_0, F)$, the $\epsilon$-closure of a set of states $S \subseteq Q$, $E(S)$, is defined as follows.

- $q \in E(S)$ if $q \in S$, and
- $q' \in E(S)$ if $q \in E(S)$ and $q' \in \delta(q, \epsilon)$. 

2
Definition 9. Given an NFA \( N = (Q, \Sigma, \delta, q_0, F) \), \( \delta^* : Q \times \Sigma^* \rightarrow \mathcal{P}(Q) \) is defined as follows.

- \( \delta^*(q, \varepsilon) = E(\{q\}) \) if \( q \in Q \), and
- \( \delta^*(q, x\sigma) = E(\bigcup_{r \in \delta^*(q, x)} \delta(r, \sigma)) \) if \( q \in Q \), \( x \in \Sigma^* \), and \( \sigma \in \Sigma \).

Definition 10. Given an NFA \( N = (Q, \Sigma, \delta, q_0, F) \) and a string \( x \in \Sigma^* \), \( x \) is accepted by \( N \) if \( \delta^*(q_0, x) \cap F \neq \emptyset \).

Definition 11. Given an NFA \( N = (Q, \Sigma, \delta, q_0, F) \), the language accepted by \( N \), or the language recognized by \( N \), is denoted by \( L(N) \), where \( L(N) = \{ x \in \Sigma^* \mid x \text{ is accepted by } N \} \).

4 Equivalence of DFAs and NFAs

It is intuitively clear that NFAs can do anything that DFAs can do by observing that a DFA transition diagram can be understood as an NFA transition diagram. However, to formally prove that result requires a little more work because the five-tuple that is a DFA is not the same as the five-tuple that is an NFA. In particular, the DFA transition function maps to states, but the NFA transition function maps to sets of states. Bridging this gap is the job of the proof of the following theorem.

Theorem 1. Given a DFA \( M \), there exists an NFA \( N \) such that \( L(N) = L(M) \).

Proof

The DFA \( M \) has the form \( M = (Q, \Sigma, \delta, q_0, F) \).
Consider the following NFA \( N = (Q_N, \Sigma, \delta_N, q'_0, F_N) \).
Let \( Q_N = Q \).
The alphabet \( \Sigma \) has been given.
Let \( \delta_N(q, \sigma) = \{ \delta(q, \sigma) \} \), for any \( q \in Q \) and \( \sigma \in \Sigma \).
Let \( \delta_N(q, \varepsilon) = \emptyset \), for any \( q \in Q \).
Let \( q'_0 = q_0 \).
Let \( F_N = F \).
Observe that \( L(N) = L(M) \).

\( \square \)

It is less obvious that DFAs can do anything that NFAs can do. Nevertheless that is the case. One attempt to prove this result might involve implementing an NFA using the concept of search. Such an argument establishes that our magical guessing machines can actually be implemented. However, it does not establish the stronger result that an NFA can be implemented by a DFA.

Another implementation approach involves placing a pebble on every state that the NFA might be in. Implementing the transition function this way then becomes a matter of updating which states have pebbles on them. Now observe that a pebble configuration itself can be understood as a state. Mathematically, the constructed DFA states are taken from the power set of the NFA states. Hence the proof below is often referred to as the “power set construction.”
Theorem 2. Given an NFA $N$, there exists a DFA $M$ such that $L(M) = L(N)$.

Proof

The NFA $N$ has the form $N = (Q, \Sigma, \delta, q_0, F)$.
Consider the following DFA $M = (Q_M, \Sigma, \delta_M, q'_0, F_M)$.
Let $Q_M = \mathcal{P}(Q)$.
The alphabet $\Sigma$ has been given.
Let $\delta_M(q, \sigma) = E(\bigcup_{r \in q} \delta(r, \sigma))$, for any $q \in Q_M$ and $\sigma \in \Sigma$.

Let $q'_0 = E(\{q_0\})$.
Let $F_M = \{q \in Q_M \mid q \cap F \neq \emptyset\}$.
Now observe that $L(M) = L(N)$.

$\delta_M(q'_0, x) = \delta_M(q_0, x)$ follows by structural induction.

- Observe that $\delta_M(q'_0, \varepsilon) = q'_0 = E(\{q_0\}) = \delta_N(q_0, \varepsilon)$.
- Assume that $\delta_M(q'_0, y) = \delta_N(q_0, y)$ if $y$ is a substructure of $x$.

Suppose $x = y\sigma$.

$$\delta_M(q'_0, x) = \delta_M(q'_0, y\sigma)$$
$$= \delta_M(\delta_M(q'_0, y), \sigma)$$
$$= \delta_M(\delta_N(q_0, y), \sigma)$$
$$= E(\bigcup_{r \in \delta_N(q_0, y)} \delta(r, \sigma))$$
$$= \delta_N(q_0, y\sigma)$$
$$= \delta_N(q_0, x)$$

Thus $M$ accepts $x$ iff $\delta_M(q'_0, x) \in F_M$ iff $\delta_N(q_0, x) \in F_M$ iff $\delta_N(q_0, x) \cap F \neq \emptyset$ iff $N$ accepts $x$.

$\square$

Corollary 1. A language $L$ is regular if and only if there exists an NFA $N$ such that $L = L(N)$.

5 Closure Properties of Regular Languages

Lemma 1. If $L$ is a regular language, then there exists an NFA $N$ such that $N$ has exactly one accepting state and $L(N) = L$.

Proof

Since $L$ is regular, there exists an NFA $N = (\{q_0, \ldots, q_n\}, \Sigma, \delta, q_0, F)$ that recognizes $L$.
Consider the following example NFA $N' = (\{q_0, \ldots, q_n, q_{n+1}\}, \Sigma, \delta_N', q_0, \{q_{n+1}\})$.
The set of states has been given.
The alphabet $\Sigma$ has been given.
Let $\delta_N'(q_i, a) = \begin{cases} 
\delta(q_i, a) & \text{if } i \leq n \text{ and } a \neq \varepsilon \text{ or } q_i \notin F \\
\delta(q_i, a) \cup \{q_{n+1}\} & \text{if } i \leq n \text{ and } a = \varepsilon \text{ and } q_i \in F \\
\emptyset & \text{if } i = n + 1 
\end{cases}$
The initial state has been given.
The set of accepting states has been given.
Observe that $L(N') = L$. 
Theorem 3. If $L_1$ is a regular language and $L_2$ is a regular language, then $L_1 \cup L_2$ is a regular language.

Proof
Sketch:
New start state has empty transitions to the start states of $N_1$ and $N_2$.

Theorem 4. If $L_1$ is a regular language and $L_2$ is a regular language, then $L_1 \circ L_2$ is a regular language.

Proof
Sketch:
Connect single accepting state of $N_1$ to start state of $N_2$.

Theorem 5. If $L$ is a regular language, then $L^*$ is a regular language.

Proof
Sketch:
New initial and final states. New initial has empty transitions to the old initial and the new final states. Add an empty transition from old final to old initial. Add an empty transition from old final to new final.

Theorem 6. If $L_1$ is a regular language and $L_2$ is a regular language, then $L_1 \cap L_2$ is a regular language.

Proof
Since $L_1$ is regular, there exists a DFA $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ that recognizes $L_1$, and since $L_2$ is regular, there exists a DFA $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ that recognizes $L_2$.
Consider the following example DFA $M = (Q_M, \Sigma, \delta_M, q_0, F_M)$.
Let $Q_M = Q_1 \times Q_2$.
The alphabet $\Sigma$ has been given.
Let $\delta_M((q, q'), \sigma) = (\delta_1(q, \sigma), \delta_2(q', \sigma))$, for any $q \in Q_1, q' \in Q_2$, and $\sigma \in \Sigma$.
Let $q_0 = (q_1, q_2)$.
Let $F_M = \{(q, q') \in Q_1 \times Q_2 \mid q \in F_1$ and $q' \in F_2\}$.
Observe that $L(M) = L_1 \cap L_2$.

Theorem 7. If $L_1$ is a regular language and $L_2$ is a regular language, then $L_1 - L_2$ is a regular language.

Proof
Since $L_1$ is regular, there exists a DFA $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ that recognizes $L_1$, and since $L_2$ is regular, there exists a DFA $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ that recognizes $L_2$.
Consider the following example DFA $M = (Q_M, \Sigma, \delta_M, q_0, F_M)$.
Let $Q_M = Q_1 \times Q_2$.
The alphabet $\Sigma$ has been given.
Let $\delta_M((q, q'), \sigma) = (\delta_1(q, \sigma), \delta_2(q', \sigma))$, for any $q \in Q_1, q' \in Q_2$, and $\sigma \in \Sigma$.
Let $q_0 = (q_1, q_2)$.
Let $F_M = \{(q, q') \in Q_1 \times Q_2 \mid q \in F_1$ and $q' \notin F_2\}$.
Observe that $L(M) = L_1 - L_2$. 
Corollary 2. If $L$ is a regular language, then $\bar{L}$ is a regular language.

6 Formal Definition of Regular Expressions

Definition 12. Given an alphabet $\Sigma$, a regular expression $R$ (over $\Sigma$) is one of the following.

- $\emptyset$, or,
- $\varepsilon$, or,
- $\sigma$ if $\sigma \in \Sigma$, or,
- $(R_1 \cup R_2)$ if $R_1$ and $R_2$ are regular expressions, or,
- $(R_1 R_2)$ if $R_1$ and $R_2$ are regular expressions, or,
- $(R^*)$ if $R$ is a regular expression.

It is conventional among mathematicians to blur the distinction between concrete syntax and abstract syntax. Thus we allow regular expressions to be written without parentheses with the understanding that the union operator $\cup$ has lowest precedence and the Kleene star operator $^*$ has the highest precedence. It is also customary to extend the syntax of regular expressions and allow $(R^+)$ if $R$ is an extended regular expression, and to allow $(R^n)$ if $R$ is an extended regular expression and $n \in \mathbb{N}$.

We can now define the meaning of a regular expression.

Definition 13. Given a regular expression $R$ over $\Sigma$, the language defined by $R$, $L(R)$, is defined as follows.

- $L(\emptyset) = \emptyset$, and,
- $L(\varepsilon) = \{\varepsilon\}$, and,
- $L(\sigma) = \{\sigma\}$ if $\sigma \in \Sigma$, and,
- $L(R_1 \cup R_2) = L(R_1) \cup L(R_2)$, and,
- $L(R_1 R_2) = L(R_1) \circ L(R_2)$, and,
- $L((R)^*) = L(R)^*$.

The meaning of $R^+$ is determined by rewriting all occurrences as $(RR^*)$; similarly, $R^n$ is rewritten as $(R \cdots R)_n$.

7 Equivalence of Regular Languages and Regular Expressions

Lemma 2. The language $L = \emptyset$ is regular.

Lemma 3. The language $L = \{\varepsilon\}$ is regular.

Lemma 4. Given an alphabet $\Sigma$, the language $L = \{\sigma\}$, where $\sigma \in \Sigma$, is regular.
Theorem 8. Given an alphabet $\Sigma$, for any regular expression $R$ over $\Sigma$, $L(R)$ is regular.

Proof

By structural induction on $R$.

- Observe that, by lemma 2, if $R = \emptyset$ that $L(R)$ is regular.
- Observe that, by lemma 3, if $R = \epsilon$, that $L(R)$ is regular.
- Observe that, by lemma 4, if $R = \sigma$, where $\sigma \in \Sigma$, that $L(R)$ is regular.
- Assume that $L(R')$ is regular if $R'$ is a substructure of $R$.

- Suppose that $R = (R_1 \cup R_2)$.
  By assumption, $L(R_1)$ and $L(R_2)$ are regular. Hence by the closure theorem 3, $L(R) = L(R_1) \cup L(R_2)$ is regular.

- Suppose that $R = (R_1 R_2)$.
  By assumption, $L(R_1)$ and $L(R_2)$ are regular. Hence by the closure theorem 6, $L(R) = L(R_1) \circ L(R_2)$ is regular.

- Suppose that $R = (R_1^*)$.
  By assumption, $L(R_1)$ is regular. Hence by the closure theorem 7, $L(R) = L(R_1)^*$ is regular.

Corollary 3. For any language $L$, if there exists a regular expression $R$ such that $L(R) = L$, then $L$ is regular.

Proof

Let $L$ be an abstract example.
Suppose there exists a regular expresion $R$ such that $L(R) = L$.
$L(R)$ is regular follows from theorem 8. Hence $L$ is regular.

Definition 14. Given a DFA $M = (Q, \Sigma, \delta, q_0, F)$, the mapping from states to languages, $L : Q \times Q \rightarrow \mathcal{P}(\Sigma^*)$, is $L(p, q) = \{x \in \Sigma^* | \delta^*(p, x) = q\}$.

Definition 15. Given a DFA $M = (Q, \Sigma, \delta, q_0, F)$, where $Q = \{1, \ldots, m\}$, the parameterized mapping from states and natural numbers to languages, $L : Q \times Q \times N \rightarrow \mathcal{P}(\Sigma^*)$, is $L(p, q, n) = \{x \in \Sigma^* | \delta^*(p, x) = q$ and no state with number greater than $n$ was on the path from $p$ to $q\}$.

Lemma 5. Given an alphabet $\Sigma$, for any string $x \in \Sigma^*$, there exists a regular expression $R$ over $\Sigma$ such that $L(R) = \{x\}$.

Lemma 6. Given an alphabet $\Sigma$, for any language $L \subseteq \Sigma^*$, if $L$ is finite, then there exists a regular expression $R$ over $\Sigma$ such that $L(R) = L$. 

Theorem 9. Given a DFA $M = (Q, \Sigma, \delta, q_0, F)$ where $Q = \{1, \ldots, m\}$, for any natural number $n$ and any $p, q \in Q$, there exists a regular expression $R$ over $\Sigma$ such that $L(R) = L(p, q, n)$.

Proof

By mathematical induction.

- Observe that there exists a regular expression $R$ over $\Sigma$ such that $L(R) = L(p, q, 0)$.
  Note that $L(p, q, 0) = \{x \in \Sigma^* \mid \delta^*(p, x) = q\}$ such that there are no states on the path from $p$ to $q$. Thus we can characterize this language explicitly as follows.
  $L(p, q, 0) = \begin{cases} 
  \{a \in \Sigma \mid \delta(p, a) = q\} & \text{if } p \neq q \\
  \{a \in \Sigma \mid \delta(p, a) = p\} \cup \{\varepsilon\} & \text{if } p = q
  \end{cases}$

  Since it is finite, by lemma 6 there exists a regular expression $R$ over $\Sigma$ such that $L(R) = L(p, q, 0)$.

- Assume there exists a regular expression $R$ over $\Sigma$ such that $L(R) = L(p, q, k)$.
  Now we will show that $L(p, q, k+1)$ is expressible using a parameterized mapping involving only $k$. This is trivial when $k + 1 > m$. In particular, we will show for the non-trivial case that $L(p, q, k+1) = L(p, q, k) \cup L(p, k+1, k) \circ L(k+1, k+1, k)^* \circ L(k+1, q, k)$.

  - To show that $L(p, q, k) \cup L(p, k+1, k) \circ L(k+1, k+1, k)^* \circ L(k+1, q, k) \subseteq L(p, q, k+1)$, assume that $x \in L(p, q, k) \cup L(p, k+1, k) \circ L(k+1, k+1, k)^* \circ L(k+1, q, k)$.
    Then it must be that $x \in L(p, q, k+1)$, since both parts of the union characterize strings that go from state $p$ to state $q$ using no state larger than $k+1$.
  - To show that $L(p, q, k+1) \subseteq L(p, q, k) \cup L(p, k+1, k) \circ L(k+1, k+1, k)^* \circ L(k+1, q, k)$, assume that $x \in L(p, q, k+1)$.
    Then there are two possibilities:
      * The path from $p$ to $q$ did not involve state $k+1$. Therefore $x \in L(p, q, k)$.
      * The path from $p$ to $q$ does involve state $k+1$. If so, there must be a first occurrence of $k+1$ after state $p$ and a last occurrence. So we can break the path into three parts: from $p$ to $k+1$, from $k+1$ back to $k+1$, and from the last occurrence of $k+1$ to $q$. Therefore $x \in L(p, k+1, k) \circ L(k+1, k+1, k)^* \circ L(k+1, q, k)$.
    Hence $x \in L(p, q, k) \cup L(p, k+1, k) \circ L(k+1, k+1, k)^* \circ L(k+1, q, k)$.

By assumption the following regular expressions exist.

- $R_1$, where $L(R_1) = L(p, q, k)$.
- $R_2$, where $L(R_2) = L(p, k+1, k)$.
- $R_3$, where $L(R_3) = L(k+1, k+1, k)$.
- $R_4$, where $L(R_4) = L(k+1, q, k)$.

And so there is a regular expression $R$ over $\Sigma$, such that $L(R) = L(p, q, k+1)$, namely $R_1 \cup R_2 R_3^* R_4$.

$\square$

Corollary 4. Given a DFA $M = (Q, \Sigma, \delta, q_0, F)$, for any $p, q \in Q$, there exists a regular expression $R$ over $\Sigma$ such that $L(R) = L(p, q)$.

Proof

Assume, without loss of generality, that $Q = \{1, \ldots, m\}$. Since there is no state with number larger than $m$, $L(p, q, m) = L(p, q)$. Hence by theorem 9, there exists a regular expression $R$ over $\Sigma$ such that $L(R) = L(p, q)$.
**Corollary 5.** For any language $L$, if $L$ is regular, then there exists a regular expression $R$ over $\Sigma$ such that $L(R) = L$.

**Proof**

Suppose $L$ is regular. Then there exists a DFA $M = (Q, \Sigma, \delta, q_0, F)$ that recognizes $L$. Without loss of generality, suppose $F = \{q_{f_1}, \ldots, q_{f_k}\}$. Then $L = L(q_0, q_{f_1}) \cup \cdots \cup L(q_0, q_{f_k})$. By corollary 4, there exists a regular expression $R_i$ such that $L(R_i) = L(q_0, q_{f_i})$. Hence there is a regular expression $R$ over $\Sigma$ such that $L(R) = L$, namely $R_1 \cup \cdots \cup R_k$.

**Corollary 6.** For any language $L$, $L$ is regular if and only if there exists a regular expression $R$ over $\Sigma$ such that $L(R) = L$. 
