Finite-state Automata and Regular Languages

1 Formal Definition

A Deterministic Finite-State Automaton is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where

1. Q is a finite set of states
2. Σ is a finite alphabet
3. $\delta : Q \times \Sigma \rightarrow$ is the transition function
4. $q_0 \in Q$ is the start state, and
5. $F \subseteq Q$ is the set of accept states

Let $M = (Q, \Sigma, \delta, q_0, F)$ to be a DFA and let $w = w_1 w_2 ... w_n$ be a word over Σ. M accepts w if there is a sequence of states $r_0, r_1, r_2, ..., r_n$ satisfying the following conditions

1. $r_0 = q_0$
2. $\delta(r_i, w_{i+1}) = r_{i+1}$ for every i, $0 \leqslant i \leqslant n-1$
3. $r_n \in F$

2 Regular Languages

The DFA M recognises the language L if L = $\{w|M \text{ accepts } w\}$

Definition: Regular language

A language is called a regular language if some DFA recognises it

3 Regular Operations

Boolean (set-theoretic):

Union	$A \cup B = \{x \mid x \in A \text{ or } x \in B\}$
Intersection	$A \cap B = \{x \mid x \in A \text{ and } x \in B\}$
Difference	$A \backslash B = \{x \mid x \in A \text{ or } x \notin B\}$
Complement	$\overline{A} = \Sigma^* \backslash A$

Language theory specific

Concatenation	$A \circ B = \{xy \mid x \in A \text{ and } y \in B\}$
Star	$A^* = \{x_1 x_2 \dots x_k \mid k \geq 0 \text{ and } x_i \in A \text{ for every } i, 1 \leq i \leq k\}$

4 Regular expression

A Regular Expression (RE) R defines a regular language L(R). We shall eventually prove that $RE \equiv DFA$ (i.e. REs define exactly class of the regular languages)

The definition is inductive (recursive), i.e. there are initial RE, and new REs can be obtained from old ones by means of Regular Operations

Definition: R is a Regular expression over the alphabet Σ if R is

1. a for some $a \in \Sigma$
2. ϵ
3. $\varnothing$
4. $(R_1 \cup R_2)$, where R_1 and R_2 are REs

5. $(R_1 \circ R_2)$, where R_1 and R_2 are REs, or
6. (R_1^*), where R_1 is an RE

Note that by convention the concatenation symbol may be omitted, i.e. R_1R_2 means $R_1 \circ R_2$. Parentheses may also be omitted, bearing in mind the precedence order

5 Combining Automata

Given L_1 recognised by $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ and L_2 recognised by $M_1 = (Q_2, \Sigma, \delta_2, q_2, F_2)$; want to combine M_1 and M_2 into a new automaton M that would recognise $L_1 \cup L_2$

Naive idea: Simulate first M_1 on the input and then simulate M_2 on the same input; accept if either M_1 or M_2 or both accept. This does not work as after M_1 has run on the input, the input is exhausted and there is no way to "rewind" it in order to run M_2

The solution is to run M_1 and M_2 on the input in parallel