1 Regular expressions

Regular expressions provide another, elegant and simple way of describing regular languages. We denote regular expressions by small letters such as \( r, s, r_1 \), etc. As with regular languages, they are defined with respect to a specified alphabet \( \Sigma \). They are defined by the following rules.

We begin with the base cases.

1. \( \phi \) is a regular expression; it represents \( \emptyset \), the empty set.

2. \( \lambda \) is a regular expression; it represents the language \( \{ \lambda \} \), the language containing the empty string alone.

3. \( a \) is a regular expression; it represents the language \( \{ a \} \).

For the recursive rules, let regular expressions \( r \) and \( s \) represent languages \( R \) and \( S \). The rules follow.

1. \( r \cup s \) is a regular expression; it represents language \( R \cup S \).

2. \( r \circ s \) is a regular expression; it represents language \( R \circ S \). We write \( rs \) for short.

3. \( r^* \) is a regular expression; it represents language \( R^* \).

Parentheses are used to indicate the scope of an operator. It is also convenient to introduce the notation \( r^+ \); while not a standard regular expression, it is shorthand for \( rr^* \), which represents the language \( RR^* \). Also we use \( \Sigma \) as a shorthand for \( a_1 \cup a_2 \cup \cdots \cup a_k \), where \( \Sigma = \{ a_1, a_2, \cdots, a_k \} \). Finally, \( L(r) \) denotes the language represented by regular expression \( r \).

Examples.

1. \( a^*ba^* = \{ w \mid w \text{ contains exactly one } b \} \).

2. \( \Sigma^*bab\Sigma^* = \{ w \mid w \text{ contains } bab \text{ as a substring} \} \).
3. $(\Sigma \Sigma)^* = \{ w \mid w \text{ has even length} \}$.

4. $a\Sigma^*a \cup b\Sigma^*b \cup a \cup b = \{ w \mid w \text{ has the same first and last character} \}$.

5. $(a \cup \lambda)b^* = ab^* \cup b^*$.

6. $a^* \phi = \phi$.

Next we show that regular expressions represent exactly the regular languages.

**Lemma 1** Let $r$ be a regular expression. There is an NFA $N_r$ that recognizes the language represented by $r$: $L(N_r) = L(r)$.

**Proof:** The proof is by induction on the number of operators (union, concatenation, and star) in regular expression $r$.

The base case is for zero operators, which are also the base cases for specifying regular expressions. It is easy to give DFAs that recognize the languages specified in each of these base cases and this is left as an exercise for the reader.

For the inductive step, suppose that $r$ is given by one of the recursive definitions, $r_1 \cup r_2$, $r_1 \circ r_2$, or $r_1^*$. Since $r_1$, and $r_2$ if it occurs, contain fewer operators than $r$, we can assume by the inductive hypothesis that there are NFAs recognizing the languages represented by regular expressions $r_1$ and $r_2$. Then Lemmas 5–7 of *Finite Automata, Part 2* provide the NFAs recognizing the languages $r$.

We can conclude that there is an NFA recognizing $L(r)$.

To prove the converse, that every regular language can be represented by a regular expression takes more effort. To this end, we introduce yet another variant of NFAs, called GNFAs (for Generalized NFAs).

In a GNFA each edge is labeled by a regular expression $r$ rather than by one of $\lambda$ or a character $a \in \Sigma$. We can think of an edge labeled by regular expression $r$ being traversable on reading string $x$ only if $x$ is in the language represented by $r$. String $w$ is recognized by a GNFA $M$ if there is a path $P$ in $M$ from its start vertex to a final vertex such that $P$’s label, the concatenation of the labels on $P$’s edges, forms a regular expression that includes $w$ among the set of strings it represents.

In more detail, suppose $P$ consists of edges $e_1, e_2, \ldots, e_k$, with labels $r_1, r_2, \ldots, r_k$, respectively; then $P$ is a $w$-recognizing path if $w$ can be written as $w = w_1w_2\cdots w_k$ and each $w_i$ is in the language represented by $r_i$, for $1 \leq i \leq k$.

Clearly, every NFA is a GNFA, so it will be enough to show that any language recognized by a GNFA can also be represented by a regular expression.

We begin with two simple technical lemmas.

**Lemma 2** Suppose that GNFA $M$ has two vertices $p$ and $q$ with $h$ edges from $p$ to $q$, edges $e_1, e_2, \ldots, e_h$. Suppose further that these edges are labeled by regular expressions $r_1, r_2, \ldots, r_h$, respectively. Then replacing these $h$ edges by a new edge $e$ labeled $r_1 \cup r_2 \cup \cdots \cup r_h$, or $r$ for short, yields a GNFA $N$ recognizing the same language as $M$. 
1 REGULAR EXPRESSIONS

Figure 1: Edge replacement in Lemma 2.

Proof: The edge replacement is illustrated in Figure 1.

First we show \( L(M) \subseteq L(N) \). Suppose that \( M \) recognizes string \( w \). Let \( P \) be a \( w \)-recognizing path in \( M \). Replacing each instance of edge \( e_i \), for \( 1 \leq i \leq h \), by edge \( e \) yields a new path \( P' \) in \( N \), with \( P' \) being \( w \)-recognizing (for a substring \( w_i \) read on traversing edge \( e_i \), and hence represented by \( r_i \), is also represented by \( r \) and so can be read on traversing edge \( e \) also. This shows \( w \in L(N) \).

Next we show \( L(N) \subseteq L(M) \). Suppose that \( N \) recognizes string \( w \). Let \( P \) be a \( w \)-recognizing path in \( N \). Suppose that edge \( e \) is traversed \( j \) times in \( P \), and let \( w_1, w_2, \ldots, w_j \) be the substrings of \( w \) read on these \( j \) traversals. Each substring \( w_g \), \( 1 \leq g \leq j \), is represented by regular expression \( r \), and as \( r = r_1 \cup \cdots \cup r_h \), each \( w_g \) is represented by one of \( r_1, r_2, \ldots, r_h \); so let \( w_g \) be represented by \( r_{i_g} \), where \( 1 \leq i_g \leq h \). Replacing the corresponding instance of \( e \) in \( P \) by \( e_{i_g} \) yields a path \( P' \) in \( M \) which is also \( w \)-recognizing. This shows that \( w \in L(M) \).

Lemma 3 suppose GNFA \( M \) has three vertices \( u, v, q \), connected by edges labeled as shown in Figure 2 and further suppose that they are the only edges incident on \( q \). Then removing vertex \( q \) and creating edge \( (u, v) \) with label \( r_1 r_2^* r_3 \) yields a GNFA \( N \)

Figure 2: Edge replacement in Lemma 3.

recognizing the same language as \( M \).

Proof: First we show that \( L(M) \subseteq L(N) \). Suppose that \( w \in L(M) \), and let \( P \) be a \( w \)-recognizing path in \( M \). We build a \( w \)-recognizing path \( P' \) in \( N \). Consider a segment of the path from \( u \) to \( v \) going through \( q \) in \( P \). It consists of edge \( (u, q) \), followed by some \( k \geq 0 \) repetitions of edge \( (q, q) \), followed by edge \( (q, v) \). This subpath has label
But all strings represented by $r_1r_2^kr_3$ are represented by $r_1r_2^*r_3$, and so we can replace this subpath by the new edge $(u,v)$ in $N$. Thus $w$ is recognized by $N$.

Next, we show that $L(N) \subseteq L(M)$. Suppose that $w \in L(N)$, and let $P$ be a $w$-recognizing path in $N$. We build a $w$-recognizing path $P'$ in $M$. Consider an instance of edge $e$ with $e = (u,v)$ on $P$, if any. Suppose string $w_i$ is read on traversing $e$. Then $w_i$ is in the language represented by $r_1r_2^kr_3$ for some $k \geq 0$ (for this is what $r_1r_2^*r_3$ means: $r_1r_2^*r_3 = r_1r_3 \cup r_1r_2r_3 \cup r_1r_2^2r_3 \cup r_1r_2^3r_3 \cup \cdots$). So we can replace this instance of edge $e$ with the edge sequence $(u,q)$ followed by $k$ instances of edge $(q,q)$, followed by edge $(q,v)$. When all instances of $e$ in $P$ are replaced, the resulting path $P'$ in $M$ is still $w$-recognizing, so $w \in M$.

\[ \text{Lemma 4} \]

Let $M$ be a GNFA. Then there is a regular expression $r$ representing the language recognized by $M$: $L(r) = L(M)$.

**Proof:** We begin by modifying $M$ so that its start vertex has no in-edge (if need be by introducing a new start vertex) and so that it has a single final vertex with no out-edges (again, by adding a new vertex, if needed). Also, for each pair of vertices if there are several edges between them, they are replaced by a single edge by applying Lemma 2. Let $N_0$ be the resulting machine and suppose that it has $n+2$ vertices \{start, $q_1$, $q_2$, \ldots, $q_n$, final\}.

We construct a sequence $N_1$, $N_2$, \ldots, $N_n$ of GNFAs, where $N_i$ is $N_{i-1}$ with $q_i$ removed and otherwise modified so that $N_{i-1}$ and $N_i$ recognize the same language, for $1 \leq i \leq n$.

Thus $L(N_n) = L(M)$. But $L_n$ has two vertices, start and final, joined by a single edge labeled by a regular expression $r$, say. Clearly, the language recognized by $N_n$ is $L(r)$. So it remains to show how to construct $N_i$ given $N_{i-1}$.

**Step 1.** Let $q = q_i$ be the vertex being removed. For each pair of vertices $u, v \neq q$ such that there is a path $u, q, v$ we make a new copy $q_{u,v}$ of $q$ as shown in Figure 3, and then remove the vertex $q$ and the edges incident on it.

\[ \text{Figure 3: Vertex duplication in Step 1.} \]

Clearly, a path segment $(u, q$ repeated $k \geq 1$ times, $v$ in $N_{i-1})$ has the same label as the path segment $(u, q_{u,v}$ repeated $k$ times, $v)$, and consequently the labels on the
recognizing paths in $N_{i-1}$ and the machine following the Step 1 modification are the same.

**Step 2.** In turn, for each vertex $q_{u,v}$, apply Lemmas 3 and 2 to the subgraph formed by $u$, $q_{u,v}$, and $v$, for each pair of vertices $u$, $v$. The resulting GNFA is $N_i$. Clearly, $L(N_i) = L(N_{i-1})$. The effect of the application of these lemmas is illustrated in Figure 4.

![Figure 4: Change due to Step 2.](image)

Note that in fact we could have performed the change achieved by Step 2 without introducing the vertices $q_{u,v}$. They are here just to simplify the explanation. ■