Problem Sheet 3

Rowan Saunders

January 17, 2022

1 Regular Languages and Regular Expressions

1.1 Regular Operations

The Union operation is defined as follows:

> For two languages $L_1, L_2 \subseteq \Sigma^*$, the **union** of these languages is denoted $L_1 \cup L_2$, and is defined analogously to sets. i.e. $L_1 \cup L_2$ contains all strings that are either members of L_1 or members L_2.

The Concatenation operation is defined as follows:

> For two languages $L_1, L_2 \subseteq \Sigma^*$, their **concatenation**, which is denoted $L_1 \circ L_2$ (alternatively L_1L_2, is the language given by $L_1 \circ L_2 = \{uv|u \in L_1, v \in L_2\}$

An Concatenation example: Let the following both be languages defined over the alphabet $\{a, b\}$

$$L_1 = \{a^n|n = 1, 2, 3...\},$$
$$L_2 = \{b^m|m = 1, 2, 3...\}$$

Then:

$$L_1 \circ L_2 = \{a^n b^m|n = 1, 2, ..., m = 1, 2, ...\}$$

So

$$aab \in L_1 \circ L_2$$
$$bab \notin L_1 \circ L_2$$

Therefore, If $L = \{a^n b^m\}$, then $L^2 = L \circ L = \{a^n b^m a^l b^k\}$

The Kleene Star, or simply star operation is a unary operation, which has been encountered before as Σ^*, which is the set of all strings in a given alphabet Σ.

> Given a language L, the **Kleene star** is defined as:
>
> $$L^* = \{w_1 w_2 ... w_n|w_i \in L, n = 0, 1, ...\}$$
>
> i.e.
>
> $$L^* = \{\varepsilon\} \cup L \cup L^2 \cup L^3 \cup ...$$

A Kleene star example:

$$L = \{a^n b^m\}, L^* = \{a^{n_1} b^{m_1} a^{n_2} b^{m_2} ... a^{n_k} b^{n_k}\}$$

If $L = \{a, b\}$ over $\Sigma = \{a, b\}$, then $L^* = \Sigma^*$

These three operations are called regular operations, because if you apply them to regular languages, the results are regular too.

This represents the following Theorem:

The class of regular languages is closed under the operations union, concatenation, and star. i.e. if L_1 and L_2 are regular languages, then the following languages are also regular:

a) $L_1 \cup L_2$
b) $L_1 \circ L_2$
c) $L_1^* \quad (\text{and} L_2^*)$

The method for proving this theorem is to construct automata that accept these languages (L_1, L_2, a, b, c)

1.2 Proof that Union Operation is Regular

Therefore, the proof that the Union operation is regular is as follows

> Assume we have regular languages L_1 and L_2. As they are regular, there must be finite automata N_1 and N_2 which accept these languages respectively. We wish to show that $L_1 \cup L_2$ is regular, and we do this by creating an NFA that accepts this language.

The following NFAs can be constructed for N_1 (*Figure 1*), N_2 (*Figure 2*) and $N_1 \cup N_2$ (*Figure 3*). It makes no difference as to whether we use NFAs or DFAs, however NFAs are often simpler to draw and document:

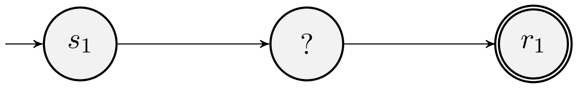

Figure 1: N_1

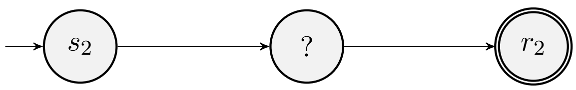

Figure 2: N_2

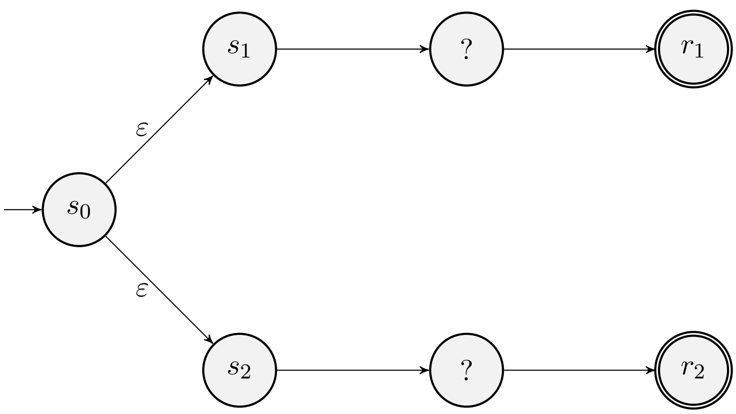

Figure 3: $N_1 \cup N_2$

Let $N_1 = (Q_1, \Sigma, \delta_1, s_1, F_1)$ and $N_2 = (Q_2, \Sigma, \delta_2, s_2, F_2)$ be non-deterministic finite automata which recognise L_1 and L_2 respectively.

We Construct $N = (Q, \Sigma, \delta, s_0, F)$ which recognises $L_1 \cup L_2$, where:

$$Q = Q_1 \cup Q_2 \cup \{s_0\}$$

Σ is the alphabet

$$\delta(q, a) = \begin{cases} \delta_1(q, a) & q \in Q_1 \\ \delta_2(q, a) & q \in Q_2 \\ \{s_1, s_2\} & q = s_0 \text{ and } a = \varepsilon \\ \emptyset & \text{otherwise} \end{cases}$$

s_0 is the initial state

$$F = F_1 \cup F_2$$

This completes the proof for closure of regular languages under the union operation.

1.3 Proof Concatenation Operator is regular

The proof that the Concatenation operation is regular is as follows

> Assume we have regular languages L_1 and L_2. As they are regular, there must be finite automata N_1 and N_2 which accept these languages respectively. We wish to show that $L_1 \circ L_2$ is regular, and we do this by creating an NFA that accepts this language.

The following NFAs can be constructed for N_1 (*Figure 4*), N_2 (*Figure 5*) and $N_1 \cup N_2$ (*Figure 6*):

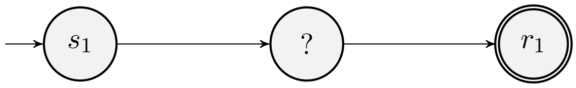

Figure 4: N_1

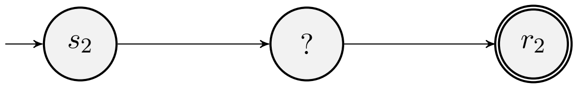

Figure 5: N_2

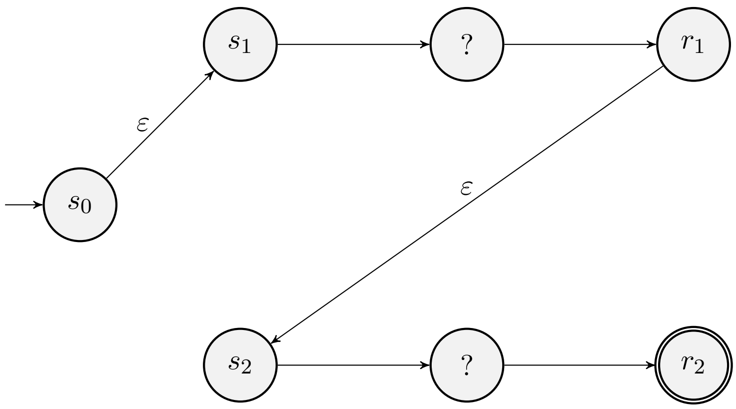

Figure 6: $N_1 \cup N_2$

Let $N_1 = (Q_1, \Sigma, \delta_1, s_1, F_1)$ and $N_2 = (Q_2, \Sigma, \delta_2, s_2, F_2)$ be non-deterministic finite automata which recognise L_1 and L_2 respectively.

We Construct $N = (Q, \Sigma, \delta, s_0, F)$ which recognises $L_1 \circ L_2$, where:

> $Q = Q_1 \cup Q_2 \cup \{s_0\}$
>
> Σ is the alphabet

$$\delta(q,a) = \begin{cases} \delta_1(q,a) & q \in Q_1 \\ \delta_2(q,a) & q \in Q_2 \\ \{s_1\} & q = s_0 \text{ and } a = \varepsilon \\ \{s_2\} & q = r_1 \text{ and } a = \varepsilon \\ \emptyset & \text{otherwise} \end{cases}$$

s_0 is the initial state

$F = F_2$

This completes the proof for closure of regular languages under the concatenate operation.

1.4 Proof Kleene Star Operator is regular

The proof that the Kleene Star operation is regular is as follows

> Assume we have a regular language L. As this is regular, there must be finite automata N which accepts this languages We wish to show that L^* is regular, and we do this by creating an NFA that accepts this language.

The following NFAs can be constructed for N (*Figure 7*) and N^* (*Figure 8*):

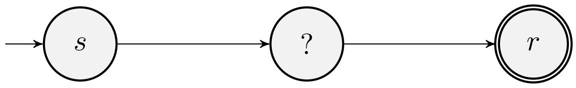

Figure 7: N

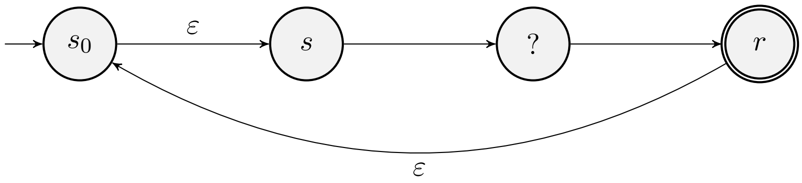

Figure 8: N^*

Let $N = (Q, \Sigma, \delta, s, F)$ be non-deterministic finite automata which recognises L.

We Construct $N_{\text{star}} = (Q_{\text{star}}, \Sigma, \delta_{\text{star}}, s_0, F_{\text{star}})$ which recognises L^*, where:

> $Q_{\text{star}} = Q \cup \{s_0\}$
>
> Σ is the alphabet
>
> $$\delta_{\text{star}}(q, a) = \begin{cases} \delta(q, a) & q \in Q \\ \{s_0\} & q = r \text{ and } a = \varepsilon \\ \{s\} & q = s_0 \text{ and } a = \varepsilon \\ \emptyset & \text{otherwise} \end{cases}$$
>
> s is the initial state
>
> $F = \{r\}$

This completes the proof for closure of regular languages under the Kleene Star operation.

2 Regular Expressions

We define a regular expression recursively. This is also called an inductive definition.

Given an alphabet Σ a **regular expression** is a string in the alphabet $\Sigma \cup \{(,), \varepsilon, \emptyset, \cup, *\}$, which meets the following rules:

1. $\emptyset, \varepsilon$ and $a \in \Sigma$ are regular expressions (called **atomic regular expressions**)
2. if α and β are regular expressions, then the following expressions are regular expressions: $(\alpha \cup \beta), (\alpha\beta), \alpha^*$

Note that $\alpha\beta$ is the standard string concatenation.

An example regular expression:

In the alphabet $\Sigma = \{a, b\}$, $(a \cup b)a^*$ is a regular expression. Because:

a and b are regular expressions by rule 1

So $(a \cup b)$ and a^* are regular expressions by rule 2

So $(a \cup b)a^*$ is a regular expressions by rule 2 again

$L(\alpha)$ is the language represented by regular expression α

The language of a regular expression is defined by:

1. $L(\emptyset) = \emptyset$
2. $L(\varepsilon) = \varepsilon$
3. $L(a) = a$ for every $a \in \Sigma$
4. $L(\alpha \cup \beta) = L(\alpha) \cup L(\beta)$
5. $L(\alpha\beta) = L(\alpha) \circ L(\beta)$
6. $L(\alpha^*) = (L(\alpha))^*$

For example, the language of $(a \cup b)a^*$:

$L((a \cup b)a^*)$

$L((a \cup b)) \circ L(a^*)$ by rule 5

$(L(a) \cup L(b)) \circ (L(a))^*$ by rule 4 and rule 6

$(\{a\} \cup \{b\}) \circ \{a\}^*$ by rule 3

$\{a, b\} \circ \{a^n | n \geq 0\}$ by regular operation definitions

We can write this as a single set in multiple ways. Here is one example:

$$(a \cup b)a^* = \{xa^n | x \in \{a, b\}, n \geq 0\}$$

We often drop redundant parenthesis, i.e the following left hand examples are written as the right hand side:

$$(a \cup (b \cup c)) = a \cup b \cup c$$
$$(a(bc)) = abc$$

There is no loss of meaning, as Union and Concatenation are associative.

Additionally, operations are always applied in a set order. The order of precedence is:

1. Star
2. Concatenation
3. Union

Therefore:

$$ab \cup c \text{ means } (ab) \cup c$$
$$ab^* \text{ means } a(b^*)$$

We can use parentheses to change the order, so we could write:

$$a(b \cup c)$$

$$(ab)^*$$

2.1 Regular Expressions to Finite Automata

We must now show that Regular Expressions are equivilent to DFA and NFAs, we can do this with the following Theorem:

(a) Any regular expression has an equivalent finite automaton
(b) Any finite automaton has an equivalent regular expression

This Theorem has the following proof:

First, construct NFAs for the atomic regular expressions $\emptyset, \varepsilon$, and $a \in \Sigma$.

These are as follows

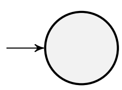

Figure 9: NFA that accepts the empty set $\emptyset$

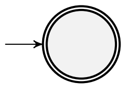

Figure 10: NFA that accepts the empty string ε

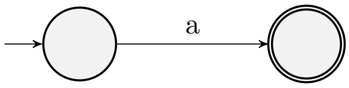

Figure 11: NFA that accepts a single letter in the alphabet $a \in \Sigma$

Now, we can recursively use the earlier proofs of regular operations to build NFA which accept $\alpha \cup \beta$, $\alpha\beta$ and α^*.

2.2 Converting Regular Expression into NFA

We create a NFA from a regular expression by building the finite automata step by step using the proofs described earlier for the regular operations.

For example for the regular expression $(a \cup b)a^*$:

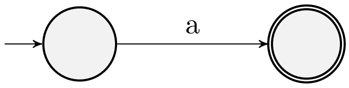

Figure 12: NFA to accept $a \in \Sigma$

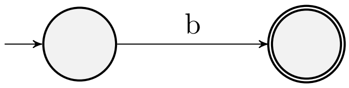

Figure 13: NFA to accept $b \in \Sigma$

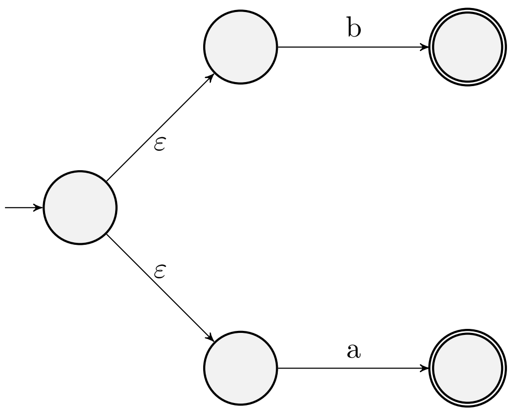

Figure 14: NFA to accept $a \cup b$

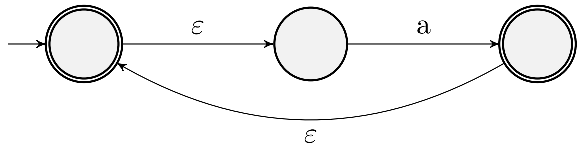

Figure 15: NFA to accept a^*

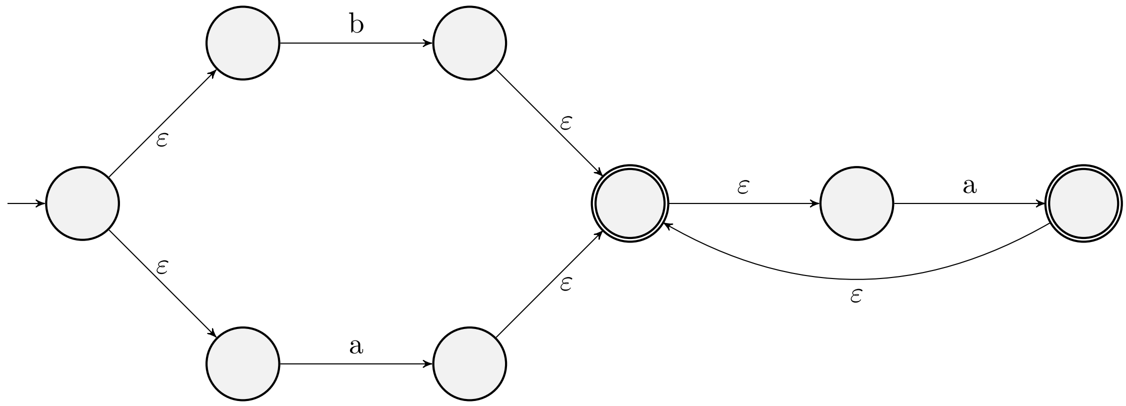

Figure 16: NFA to accept $(a \cup b) \circ a^*$

2.3 Converting Finite Automata into Regular Expressions