CS 4510 Automata and Complexity January 18th 2023

Lecture 3: Regular Expressions

Lecturer: Abrahim Ladha *Scribe(s): Yitong Li*

1 Regular Expressions

In the terminal, we usually enter "`ls *.pdf`" to search for every single file ending with ".pdf". Such language is called regular expressions.

Definition 1.1. We say that R is a **regular expression**, or regex, if R is one of the following:

(a) $\emptyset$ - empty set

(b) ε - empty string

(c) $a \quad \forall a \in \Sigma$

(d) R_i^*, R_iR_j or $R_i \cup R_j$ where R_i, R_j are regular expressions.

$\diamondsuit$

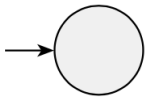
Our first three, our base cases. These are actually shorthand for the sets $\emptyset, \{\varepsilon\}, \{a\}$. Here are some examples of regular expressions:

1. $a^* = \{a^i \mid i \in \mathbb{N}\} = \{\varepsilon, a, aa, aaa, \dots\}$
2. $a^*ba^* = \{a^iba^j \mid i, j \in \mathbb{N}\}$ = all strings with a single b.
3. $\Sigma^*aab\Sigma^* = \{$all strings with aab as a substring$\}$
4. $(a \cup b)^*aab(a \cup b)^*$
5. $(\Sigma\Sigma)^* = ((a \cup b)(a \cup b))^* = ((aa \cup ab \cup ba \cup bb))^*$ = all strings of even length
6. $L_1L_2 = \{xy \in \Sigma^* \mid x \in L_1, y \in L_2\}$
7. $a^*\emptyset = \emptyset$
8. $\emptyset^* = \{\varepsilon\}$

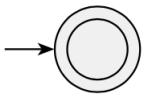
2 $\mathscr{L}(REX) \subseteq \mathscr{L}(NFA)$

To prove that $\mathscr{L}(REX) \subseteq \mathscr{L}(NFA)$, we want to show that for each regular expression, there exists an equivalent NFA. Given that regular expressions are recursively defined, it is natural to choose to proceed by induction. Let R be a regular expression. We start with the following base cases:

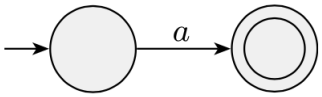
1. $R = \emptyset$.



2. $R = \varepsilon$.

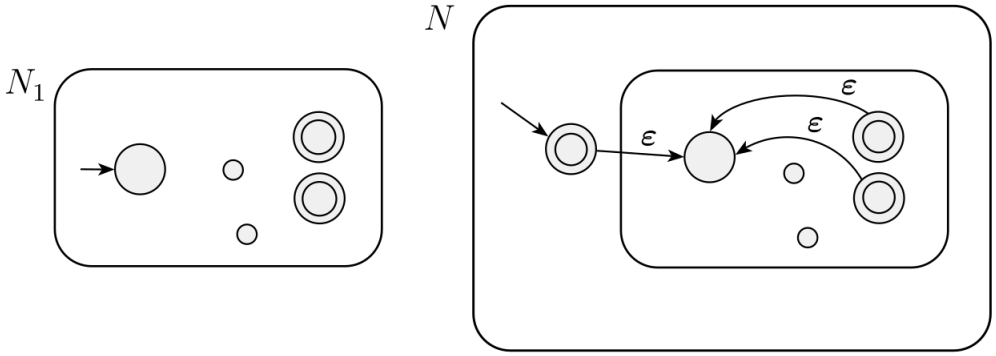


3. $R = a \in \Sigma$.



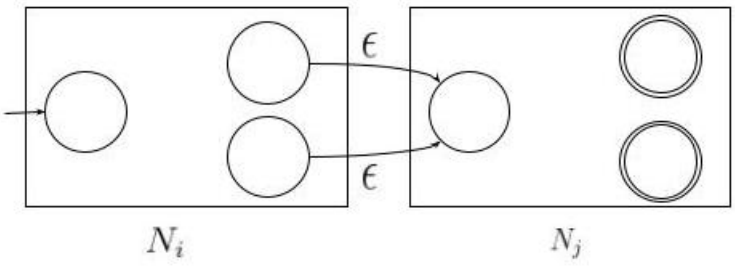
Next, we continue with the inductive steps. Let R_i, R_j be regular expressions that decide regular languages, by strong induction. We will show $R_i^*, R_iR_j, R_i \cup R_j$ are also regular. Since R_i, R_j are regular, there exist NFAs N_i, N_j to decide $L(R_i), L(R_j)$.

1. $R = R_i^*$.



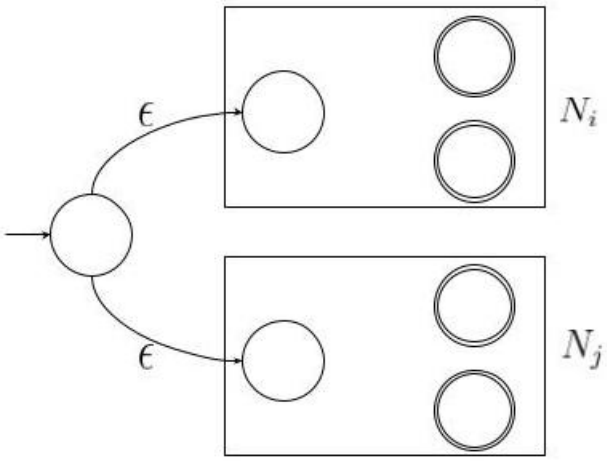
add new state q', ε-transition from q' and all states of F to the old start state q, mark q' as accepting.

2. $R = R_iR_j$.



remove final states F_i and $\forall f \in F_i$, add $\delta(f, \varepsilon) = q_j$ where q_j is the initial state of N_j.

3. $R_i \cup R_j$.



add new start state q and $\delta(q, \varepsilon) = \{q_i, q_j\}$.

3 Example from REX to NFA

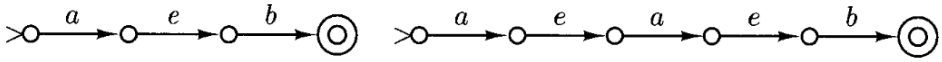
- $(ab \cup aab)^*$

stage 1

a; b

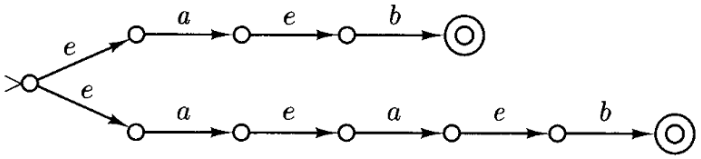
stage 2

ab; aab



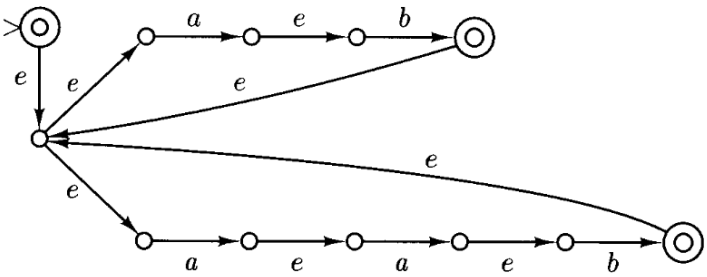
stage 3

$ab \cup aab$



stage 4

$(ab \cup aab)^*$



4 $\mathscr{L}(NFA) \subseteq \mathscr{L}(REX)$

Definition 4.1. The **GNFA** is defined as an NFA with the following properties:

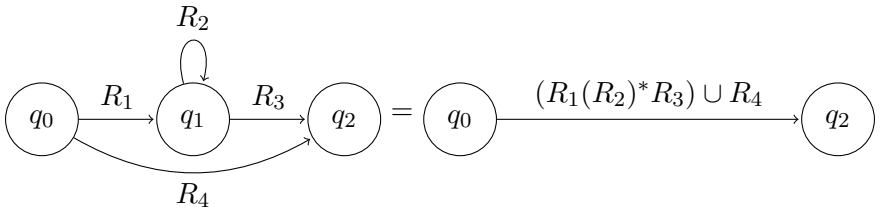
(a) The transitions have a regular expression on them.

(b) The start state has no incoming transitions

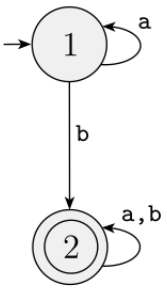
(c) The final state has no outgoing transitions

(d) Every pair of states has a transition.

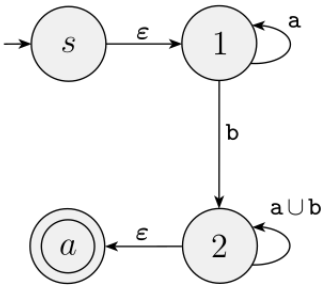
$\diamondsuit$ Taking a transition in a DFA is reading some single symbol of the front of the input. Taking a transition of a GNFA is nondeterministically choosing some prefix of the input which satisfies the regex on the transition. To convert an NFA to a regular expression, we first add a new start and final state. Then, rip out one state at a time using the following rules until only two states and one transition is left.



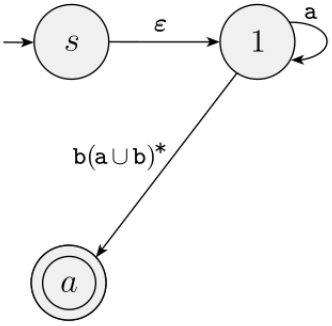
For most very connected NFAs, conversion to a regex will result in one with exponential length. Here is a relatively simple example.



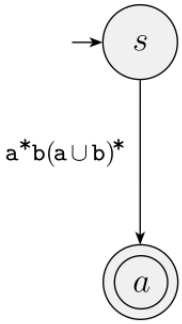
(a)



(b)



(c)



(d)

5 Set Closure

We say a set S is **closed** under an operation Δ if $\forall a, b \in S$, $a \Delta b \in S$.

1. $\mathbb{N}$ is closed under $+, \times$ and not closed under $-, \div$.
2. $\mathbb{Z}/\{0\}$ is closed under $+, -, \times$ and not closed under $\div$.
3. $\mathbb{Q}$ is closed under $+, -, \times$ and not closed under $\div$.
4. Regular languages are closed under $*, \circ, \cup$ and complement.

Although regular languages are closed under complement. Complement is not a valid operation of a regular expression. Using the operations regular languages are known to be closed under, we can prove closure under even more operations without having to contruct messy DFAs or NFAs. Recall we did a cartesian product of DFAs to prove that regular languages were closed under intersection. We give a shorter proof in the syntax of set theory.

Suppose that L_i, L_j are two regular languages. Then surely $\overline{L_i}, \overline{L_j}$ are regular, then surely $\overline{L_i} \cup \overline{L_j}$ is regular. Then so must be $\overline{\overline{L_i} \cup \overline{L_j}}$. From Demorgans law, we know that $\overline{\overline{L_i} \cup \overline{L_j}} = L_i \cap L_j$.

Similarly, we can show regular languages are closed under symmetric difference, or xor. We have a formula under composition of operators that we maintain closure under. $L_i \oplus L_j = (\overline{L_i} \cap \text{Ł}_j) \cup (L_i \cap \overline{L_j})$