MTL783: Theory of Computation 1st Semester, 2025-2026

Lecture 8 — 21/08/2025

Lecturer: Prof Minati De *Scribe: Team 8*

Scribed by:

1. Agrim Chandra (2022MT11952)
2. Arnab Goyal (2022MT61963)
3. Naman Goel (2022MT11272)
4. Rakshit Sohlot (2022MT11919)
5. Sarthak Maheshwari (2022MT11258)

1 Overview

In the last lecture, we explored regular expressions and the languages they represent. We demonstrated that every regular expression can be transformed into an equivalent NFA, using examples of concatenation, union, and Kleene star to illustrate the construction process.

In this lecture, we prove the converse, that every regular language has a regular expression.

2 Main Section

So far, we have seen that language represented by regular expression is a subset of regular language since it is possible to construct a DFA/NFA corresponding to regular expression. Now we prove the equivalence of the two.

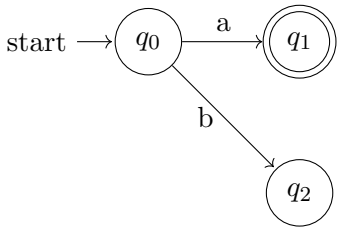
2.1 Regular Expressions for Regular Languages

Theorem 1. *Let L be a regular language. Then there exists a regular expression r such that $L = L(r)$.*

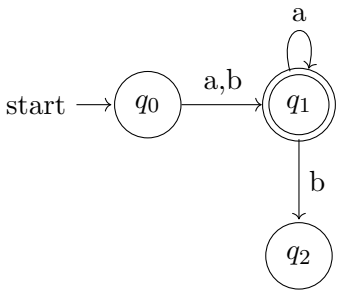
Theorem 1 together with the already known result of every regular expression having a regular language implies that regular expression and regular language are equivalent.

2.1.1 Intuition for Theorem 1

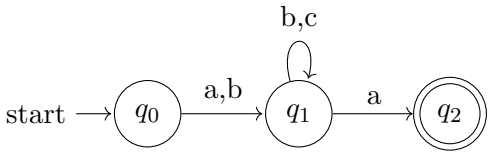
We first develop some intuition for the above theorem using the following examples.



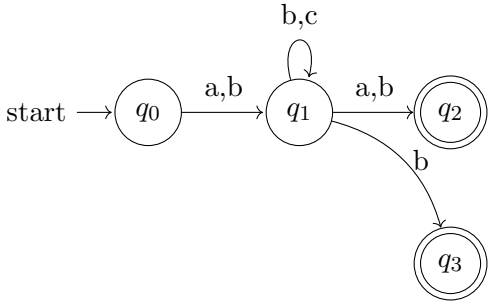
(a) Regular Expression: a



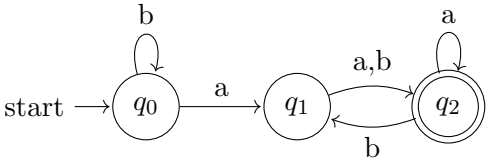
(b) Regular Expression: $(a+b)a^*$



(c) Regular Expression: $(a+b)(b+c)^*a$



(d) Regular Expression: $(a+b)(b+c)^*(a+b)$



(e) Regular Expression: $b^*a(a+b)(a^*b(a+b))^*$

2.1.2 Generalized non-deterministic finite automata (G.N.F.A)

Definition 2. *A G.N.F.A is a 5-tuple* $(Q, \Sigma, \delta, q_{start}, q_{accept})$ *where:*

- Q *: a finite set of states,*
- Σ *: a finite input alphabet,*
- $\delta : (Q - \{q_{accept}\}) \times (Q - \{q_{start}\}) \rightarrow \mathcal{R}$*, where* $\mathcal{R}$ *is the collection of all possible regular expressions in* Σ^**,*
- $q_{start} \in Q$ *is the unique start state,*
- $q_{accept} \in Q$ *is the unique accept state.*

A string $w \in \Sigma^*$ is accepted *by a G.N.F.A if $w = w_0 w_1 \dots w_k$ and there exists a sequence of states $q_0, q_1, \dots, q_k$ such that*

- $q_0 = q_{start}$,
- $q_k = q_{accept}$,
- *for each i, $w_i \in L(R_i)$ where $R_i = \delta(q_i, q_{i+1})$.*

Using GNFA to Derive Regular Expressions: We use GNFAs to find a regular expression for any regular language. The procedure is as follows:

1. Convert any given DFA/NFA into an equivalent GNFA.
2. The language of the GNFA can then be derived from its transition graph as a regular expression.
3. Hence, we obtain a regular expression representing the language of the original DFA/NFA.

Converting a DFA/NFA to a GNFA:

- Begin with a DFA or NFA for L.
- Add a new start state with a λ-transition to the original start state.
- Add a new accept state with λ-transitions from all original final states.
- Replace any missing edges with $\varnothing$.
- Ensure that every edge is labeled by a regular expression.

State Elimination Process: If the GNFA has more than two states (i.e., some state other than the start state and the accept state), choose an intermediate state q_k with transitions $q_i \to q_k \to q_j$. Modify the GNFA by adding a direct transition $q_i \to q_j$ with a label that represents the concatenation of the expressions along $q_i \to q_k \to q_j$. Then remove the edges (q_i, q_k) and (q_k, q_j).

This process is repeated until only the start and accept states remain, yielding a single regular expression for the language.

2.1.3 Proof Sketch for Theorem 1

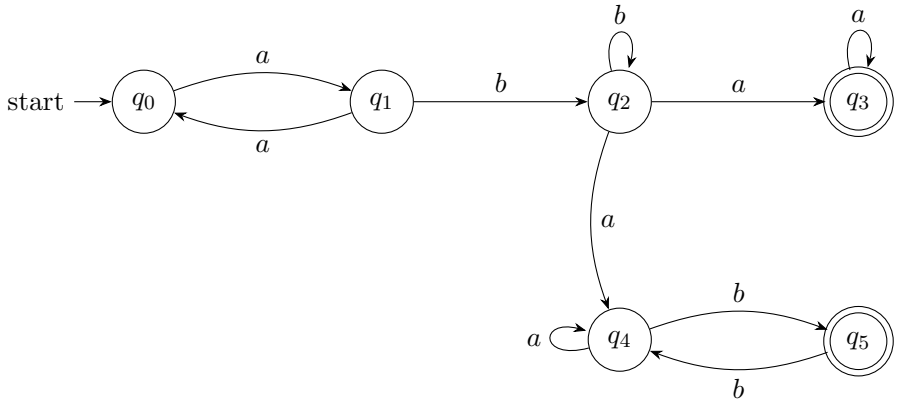
We sketch the proof for the equivalence of regular languages and regular expressions.

1. First convert the DFA/NFA that accepts L to an equivalent GNFA. Let G be the GNFA.
2. Let k be the number of states in G.
3. If $k = 2$, return $\delta(q_{\text{start}}, q_{\text{accept}})$.

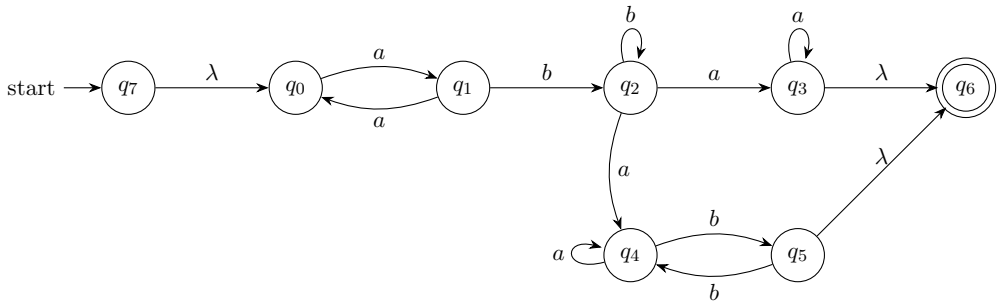
4. Else ($k \geq 2$), choose any $q \in Q \setminus \{q_{\text{start}}, q_{\text{accept}}\}$ and remove it. Obtain an equivalent GNFA G' and go to Step 2.

Note: The equivalence proof in Step 4 is omitted (not covered in class).

2.1.4 Example: Determine the regular expression corresponding to the following NFA

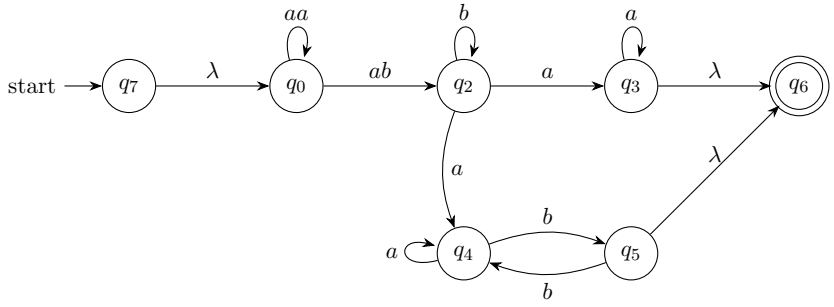


Solution: Convert the NFA to a GNFA by adding a new start state s and a new final state f, and replace multi-edge transitions by regular-expression labels on GNFA edges. (All λ edges are explicitly shown.)

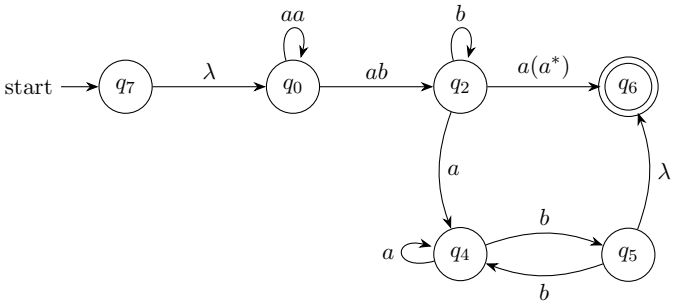


Now we do the state elimination process for the obtained GNFA.

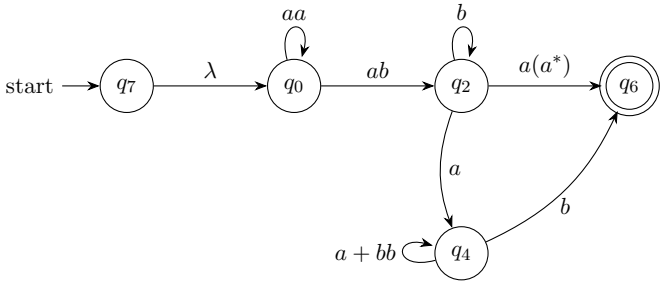
Step 1: Eliminate q_1



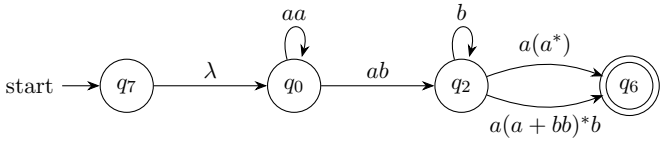
Step 2: Eliminate q_3



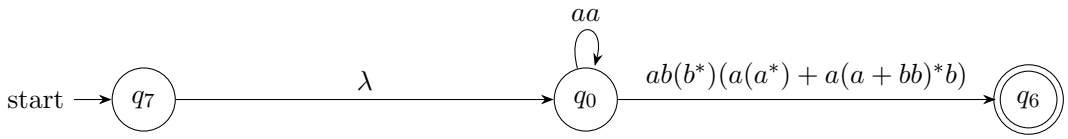
Step 3: Eliminate q_5



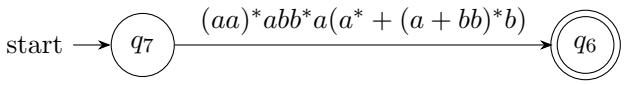
Step 4: Eliminate q_4



Step 5: Eliminate q_2



Step 6: Eliminate q_0



2.2 Proving language is regular or not

Problem. Prove that $L = \{a^n b^n \mid n \geq 0\}$ is not a regular language.

Proof. Assume, for the sake of contradiction, that there exists a DFA M that accepts L. Since M has finitely many states, by the Pigeonhole Principle, there exist n_1, n_2 with $n_1 \neq n_2$ such that after reading a^{n_1} and a^{n_2}, M reaches the same state.

Now, since M accepts $a^{n_1}b^{n_1}$, it will also accept $a^{n_2}b^{n_1}$ starting from the same state. But $a^{n_2}b^{n_1} \notin L$ whenever $n_1 \neq n_2$. This is a contradiction.

Hence, L is not regular.

More Problems.

$$L_2 = \{w \in \{0,1\}^* \mid w \text{ has the same number of 0s and 1s}\},$$
$$L_3 = \{w \in \{0,1\}^* \mid w \text{ has the same number of 01 and 10 substrings}\}.$$

3 Conclusion

In this class, we proved the equivalence of regular language and regular expression and saw how to use GNFA to find regular expressions. We briefly started a discussion on proving regularity/non-regularity of any language and will see more general ways to deal with such problems in the next class.

References

[1] Peter Linz, *An Introduction to Formal Languages and Automata*, 6th edition, Jones & Bartlett Learning, Burlington, MA, 2016.