# Lecture Notes: Theory of Computation — Regular Pumping Lemma, Conversion of FA to Regular Expressions (Course: MIT 18.404J)

Thobias K. Høivik

April 3, 2025

## Regular Expressions to Finite Automata

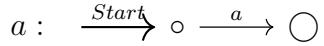
**Theorem 1.** *If $R$ is a regular expression and $A = L(R)$ the language of strings which $R$ describes. Then $A$ is regular.*

*Proof.* We shall convert $R$ to an equivalent NFA $M$. Then if $M$ recognizes A, A is regular. There are two cases to consider:
**Case 1:** R is atomic, i.e

- $R = a$ for $a \in \Sigma$
- $R = \epsilon$
- $R = \emptyset$

If $R = a$ we construct $M$ such that $\delta(q_0, a) = q_1 \in F$. Then the string $a$ will be accepted and any string not equal to $a$ will go to a terminated state or not enter the accept state. Thus $M$ accepts $R$.
If $R = \epsilon$ we construct $M$ such that $q_0 \in F$. Then the empty string $\epsilon$ will be accepted and any non-empty string will go to a terminated state. So $M$ accepts $R$.
If $R = \emptyset$, construct $M$ to accept no strings. Then since there are no strings in $R$, $R$ is recognized by $M$.
**Case 2:** $R$ is composite. If $R$ is not atomic it is the combination of some finite number of the operations: union, concatenation and star. Since regular languages are closed under these operations, $M$ can be constructed to recognize $R$.
Thus, for any regular expression $R$, there exists an NFA $M$ that recognizes $L(R)$, proving that $A$ is regular. □

Note how this makes it possible to convert from regular expressions to nondeterministic finite automata which can then be turned into deterministic finite automata.
**Example:** Convert $(a \cup ab)^*$ to an NFA.

$a$ : 

$$b: \quad \xrightarrow{Start} \circ \xrightarrow{b} \bigcirc$$

$$ab: \quad \xrightarrow{Start} \circ \xrightarrow{a} \circ \xrightarrow{\epsilon} \circ \xrightarrow{b} \bigcirc$$

For $a \cup ab$ we let $\delta(q_0, \epsilon) = \{\text{Start of ab}, \text{Start of a}\}$. For $(a \cup ab)^*$ we let $q_0 \in F, \delta(q_0, \epsilon) = \{q_1\}, \delta(q_1, \epsilon) = \{\text{Start of a}, \text{Start of ab}\}, \delta(F_a, \epsilon) = q_1 = \delta(F_{ab}, \epsilon)$.

# Generalized Nondeterministic Finite Automata

**Definition 1** (GNFA)**.** *A generalized nondeterministic finite automaton (GNFA) is similar to an NFA, but allows regular expressions as transitions.*

**Definition 2** (GNFA — Rigorous Def)**.** *A* ***generalized nondeterministic finite automaton (GNFA)*** *is a 5-tuple*

$$(Q, \Sigma, \delta, q_{start}, q_{accept})$$

*where:*

- $Q$ *is a finite set of states.*
- $\Sigma$ *is the input alphabet.*
- $\delta : Q \times Q \to \mathcal{R}(\Sigma)$ *is the* ***transition function****, where each pair of states* $(q_i, q_j)$ *is mapped to a* ***regular expression*** $R_{i,j}$ *over* $\Sigma$*. This expression represents the set of strings that cause a transition from* $q_i$ *to* $q_j$*.*
- $q_{start} \in Q$ *is the* ***start state****, and it has no incoming transitions.*
- $q_{accept} \in Q$ *is the* ***accept state****, and it has no outgoing transitions.*

*A GNFA accepts an input string* $w \in \Sigma^*$ *if and only if* $w$ *belongs to the language described by the regular expressions along some path from* $q_{start}$ *to* $q_{accept}$*. Formally, there exists a sequence of states*

$$q_0, q_1, \ldots, q_k \quad \text{where } q_0 = q_{start} \text{ and } q_k = q_{accept}$$

*such that*

$$w \in L(R_{q_0,q_1} \cdot R_{q_1,q_2} \cdots R_{q_{k-1},q_k}).$$

**Lemma 1.** *Every GNFA* $G$ *has an equivalent regular expression* $R$*.*

*Proof.* We shall prove the lemma with induction where $k$ denotes the number of states.
**Basis** $(k = 2)$**:**
$G = (Q, \Sigma, \delta, q_{start}, q_{accept})$ where $\delta(q_{start}, q_{accept}) = r$. Then the equivalent regular expression is $R = r$, the regular expression that takes $q_{start}$ to $q_{accept}$.
**Hypothesis:**
We assume that there exists a regular expression equivalent to the $k$-state GNFA $G$, $k \geq 2$.
**Induction:**

Consider the GNFA $G_{k+1}$ with $k+1$ states. Since $k+1 > 2$ there must be at least one intermediate state $x$ which is, then, not a start- or an accept state. We construct a new GNFA $G$ with $k$ states, identical to $G_{k+1}$, but without $x$. Let $y, z$ denote arbitrary states in $G$ (also in $G_{+1}$) where $\delta(y, x) = r_1, \delta(x, x) = r_2, \delta(x, z) = r_3$. Then in $G$ we define $\delta(y, z) = r_1(r_2)^* r_3$ which is a new composite regular expression which adheres to the language. We do this for every pair $y, z$ of states which are ingoing and outgoing with respect to $x$. Then the regular expression that would go through $y, x, z$ to be accepted or not accepted will go through $y, z$ and reach the same final state. Thus $G_{k+1}$ is equivalent to the same regular expression as $G$. Now we have created a GNFA $G$ equivalent to $G_{k+1}$, but with $k$ states.

**Conclusion:**

Since we know the $k$-state GNFA $G$ to be equivalent to a regular expression $R$, and we know that a $k+1$-state GNFA $G_{k+1}$ can be transformed into $G$, it follows that $G_{k+1}$ has an equivalent regular expression $R$. This completes the proof via the principle of mathematical induction.

□

**Corollary 1.** *Every DFA has an equivalent regular expression.*

*Proof.* Every DFA is a special case of an NFA. Every NFA can be converted into an equivalent GNFA. By lemma 1, every GNFA can be converted into an equivalent regular expression. Therefore every DFA can be converted into an equivalent regular expression. □

# Non-Regular Languages

**Lemma 2** (Pumping Lemma)**.** *For every regular language $A$, there is a number $p$, the pumping length, such that if $s \in A$ and $|s| \geq p$ then $s = xyz$ where:*

1. $xy^i z \in A, \forall i \geq 0$
2. $y \neq \epsilon$
3. $|xy| \leq p$

*Proof.* Let DFA $M$ recognize $A$. Let $p$ be the number of states in $M$. Pick $s \in A$ where $|s| \leq p$. Since $s$ is longer than the number of states in $M$ it is guaranteed to repeat a state $q_j$ more than once. If we divide up $s = xyz$ such that there is some sequence of states that leads $x$ to the state $q_j$ which is repeated, $\delta(q_j, y) = q_j$ or otherwise leads backe to $q_j$ and $z$ is the sequence of characters which takes $q_j$ to an accept state. Then we can repeat $y$ any number of times and still end up at $q_j$ which will go to the accept state through $z$. Since $M$ is deterministic it follows that $y \neq \epsilon$. Also $|xy| \leq p$, because it corresponds to the first occurence of a repeated state in a DFA with $p$ states.

□

**Example:** Let $D = \{0^k 1^k | k \geq 0\}$. $D$ is not regular.

*Proof.* Assume that $D$ is regular. The pumping lemma gives $p$. Let $s = 0^p1^p \in D$. $|s| = 2p > p$. We can divide $s = xyz$ satisfying the three conditions of the pumping lemma. However $xyyz = 0^k1^j, k \neq j \notin D$. This contradicts our assumption that $D$ is regular and so it must, in fact, not be regular. □