CSCI 341–Fall 2024: Lecture Notes
Set 6: Regular Expressions

Edward Talmage

September 20, 2024

We now have two different ways to specify regular languages. Unfortunately, it can be tricky both to determine the language recognized by an NFA or DFA, and to represent that language precisely. We want a (more) human-readable representation than set-builder that we know describes a regular language, not just any language. To get such a notation, we will introduce shorthand for our regular operations to build complex regular languages.

Examples: Let R be the set of all regular languages.

- $\emptyset, \varepsilon, \{0\}, \{1\}$ are all regular.
- $\{0\} \cup \{1\}$is regular, since R is closed under union.
- $\{0101\}$ is regular, since R is closed under concatenation.
- $\{0101, 01\}$ is regular
- $(\{0\} \cup \{1\})^*$ is regular (closure under Kleene star)
- $\{0,1\}^* \circ \{1\}$ is regular (union, star, concatenation)

Exercise: Consider the following regular languages we have seen previously. Can you build them from known regular languages using regular operations?

1. All odd-length strings.
2. All even-length strings.
3. Σ^*
4. $\{w \mid |w| \equiv 1 \pmod 4\}$
5. $\{w \mid w[0] = w[-1]\}$
6. $\{w \mid |w| \equiv 0 \pmod 3 \text{ or } w \text{ ends in “R”}\}$ over $\Sigma = \{0,1,2,R\}$
7. $\{w \mid \ w \text{ contains } 00 \text{ or } 11\}$

Definition 1. We define *Regular Expressions* inductively:

- $\emptyset$ is a regular expression denoting $L = \emptyset$.
- ε is a regular expression denoting $L = \{\varepsilon\}$.
- $a \in \Sigma$ is a regular expression denoting $L = \{a\}$.
- if r and s are regular expressions, then each of the following is a regular expression:
  - $(r \cup s)$, denoting $L = L(r) \cup L(s)$
  - $(r \circ s) = rs$, denoting $L = L(r) \circ L(s)$
  - r^*, denoting $L = L(r)^*$

$*$ has higher precedence than $\circ$, which has higher precedence than $\cup$.

Examples:

Language	Regular Expression
$\{w \mid\ w \text{ ends in } 101\}$	$(0 \cup 1)^*101$
$\{w \mid\ w \text{ contains } 101\}$	$(0 \cup 1)^*101(0 \cup 1)^*$
$\{w \mid\ \lvert w\rvert \text{ is even}\}$	$(00 \cup 01 \cup 10 \cup 11)^*$ or $((0 \cup 1)(0 \cup 1))^*$
$\{w \mid\ w \text{ has an even number of 0s}\}$	$(1^*01^*0)^*1^*$
$\{w \mid\ w \text{ has a positive even number of 0s}\}$	$(1^*01^*0)(1^*01^*0)^*1^* = (1^*01^*0)^+1^*$
$\{w \mid\ \lvert w\rvert \text{ is divisible by } 3\}$	$((0 \cup 1)(0 \cup 1)(0 \cup 1))^*$
$\{w \mid\ w \text{ has a 0 at every even position}\}$	$((0 \cup 1)0)^*(1 \cup 0 \cup \varepsilon)$

Claim 1. *The set of regular languages is the set of languages which can be described by a regular expression.*

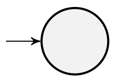
We need to show

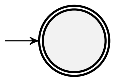
1. Every language described by a regular expression is regular and
2. every regular language (described by a DFA or NFA) can also be described by a regular expression.

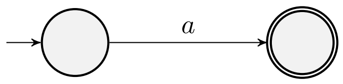
Lemma 1. *Let r be any regular expression over alphabet Σ. Then there is an NFA N s.t. $L(N) = L(r)$.*

Proof. We construct N inductively.

BC: We have three base cases:

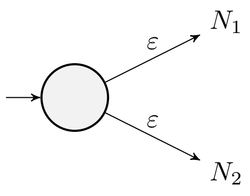
– If $r = \emptyset$, then $N =$ 

– If $r = \varepsilon$, then $N =$ 

– If $r = a \in \Sigma$, then $N =$ 

IH: For each of the following cases, assume there are NFAs N_1 and N_2 with $L(N_1) = L(r_1)$ and $L(N_2) = L(r_2)$.

IS: We again have three cases, one for each of the regular operations:

– If $r = r_1 \cup r_2$, then $N =$ 

– If $r = r_1 \circ r_2$, then $N =$ $\longrightarrow N_1, F_1 \xrightarrow{\varepsilon} N_2$

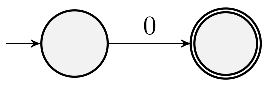
– If $r = r_1^*$, then $N = (Q_1 \cup \{q_0\}, \Sigma, \delta, q_0, F_1)$, where $q_0 \notin Q_1$ and $\delta = \begin{cases} \delta_1 \cup \{\delta(f, \varepsilon) = q_1\} & f \in F_1 \\ \delta(q_0, \varepsilon) = q_1 \end{cases}$

□

Exercise: Use this construction to build an NFA recognizing $(01 \cup 1)^*$.

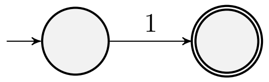
Solution:

We build up the different pieces from base cases:

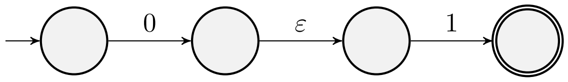
- 0: 

Solution, continued

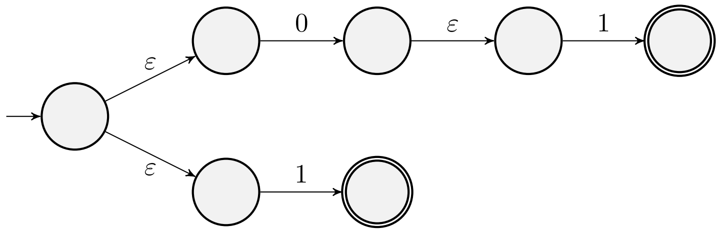
- 1:



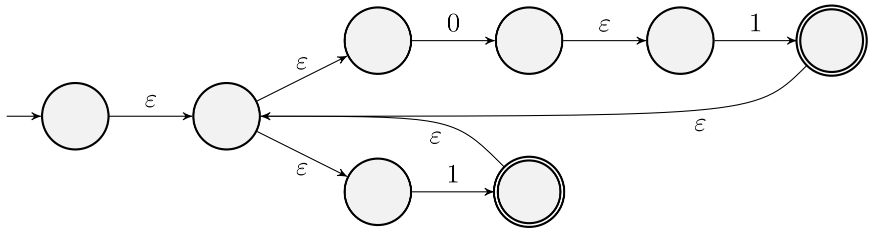
- 01:



- $01 \cup 1$:



- $(01 \cup 1)^*$:



Lemma 2. *Let L be any regular language. Then there is some regular expression r s.t. $L(r) = L$.*

This is much trickier than the previous lemma, as we have to encode all the logic possible (branching, loops, etc.) in a DFA in regular expressions.

- Idea 1: We can use *Generalized NFAs*. These allow edges marked with regular expressions. With careful setup, we can remove nodes one at a time, updating other edges' regular expressions to account for the now-missing paths. Eventually, we have just one edge from start state to accept state with a complete regular expression on it.
- Idea 2: Directly show that we can give an equivalent RE by building REs describing paths from any state in the machine to any other.

Recall the definition of $\hat{\delta}$ for DFAs.

Proof. Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA with $L(M) = L$, where WLOG $Q = \{1, 2, \ldots, n\}, n \in \mathbb{N}, q_0 = 1$.
Define

$$S^k_{i,j} := \{w \mid w \text{ drives } M \text{ from } i \text{ to } j, \text{ without using any intermediate states numbered above } k\}$$

where "drives" means that $j = \hat{\delta}(i, w)$. Note that "intermediate states" does not include the paths' endpoints i and j. We will show by induction on k that $S^k_{i,j}$ has an equivalent RE, $r^k_{i,j}$. Then $L(M) = \cup_{q \in F} S^n_{1,q} = \cup_{q \in F} L(r^n_{1,q})$, and we have a regular expression equivalent to the original DFA.

Base Case:

- $S^0_{i,j} = \begin{cases} \{a \in \Sigma \mid \delta(i,a) = j\} & i \neq j \\ \{\varepsilon\} \cup \{a \in \Sigma \mid \delta(i,a) = j\} & i = j \end{cases}$
- $r^0_{i,j} = \begin{cases} \bigcup\limits_{\substack{a \in \Sigma \\ \delta(i,a) = j}} a & i \neq j \\ \bigcup\limits_{\substack{a \in \Sigma \\ \delta(i,a) = j}} a \cup \varepsilon & i = j \end{cases}$

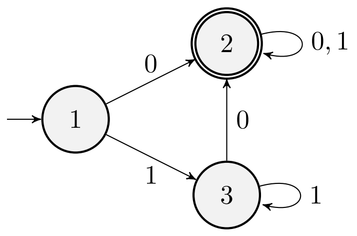
Inductive Hypothesis: Assume that $S_{i,j}^{k-1} = L(r_{i,j}^{k-1})$ for arbitrary $1 \leq k < n$.

Inductive Step: Let $r_{i,j}^{k} = r_{i,j}^{k-1} \cup (r_{i,k}^{k-1}(r_{k,k}^{k-1})^* r_{k,j}^{k-1})$. This is a regular expression for $S_{i,j}^{k}$, since any string that drives M from state i to state j using states numbered no more than k can either not use state k at all, which means it is a string in $S_{i,j}^{k-1}$, which we know by the inductive hypothesis equals $L(r_{i,j}^{k-1}$, or may consist of a string that drives M to state k using only lower-numbered states, loops back to state k some number of times using only states numbered below k in each loop, then drives M from state k to state n using only states numbered below k. But each of these sets of strings that drive M on a subpath uses only states numbered below k, so by the inductive hypothesis, we have regular expressions for each such set.

Combining the subpaths, $r_{i,j}^{k}$ describes all strings which drive M from i to j using only intermediate states numbered k and below, so is equivalent to the set of strings driving M as required.

Now, $L(M) = \bigcup_{q \in F} r_{1,q}^{n}$, so we have the claim. □

Example:



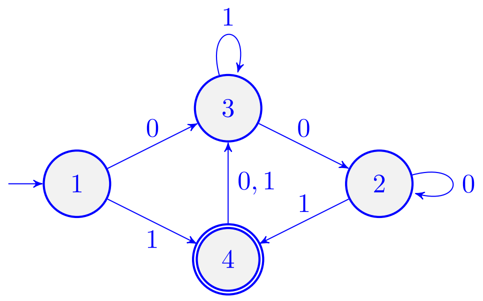
Many of the subsets are of little interest, since they represent impossible (or nearly) paths.

- $S_{11}^0 = L(\varepsilon)$
- $S_{11}^1 = L(\varepsilon)$
- $S_{11}^2 = L(\varepsilon)$
- $S_{11}^3 = L(\varepsilon)$
- $S_{21}^0 = L(\emptyset)$
- $S_{21}^1 = L(\emptyset)$
- $S_{21}^2 = L(\emptyset)$
- $S_{21}^3 = L(\emptyset)$
- $S_{23}^0 = L(\emptyset)$
- $S_{23}^1 = L(\emptyset)$
- $S_{23}^2 = L(\emptyset)$
- $S_{23}^3 = L(\emptyset)$
- $S_{31}^0 = L(\emptyset)$
- $S_{31}^1 = L(\emptyset)$
- $S_{31}^2 = L(\emptyset)$
- $S_{31}^3 = L(\emptyset)$
- $S_{33}^0 = L(1)$
- $S_{33}^1 = L(1)$
- $S_{33}^2 = L(1)$
- $S_{33}^3 = L(1^*)$

Those of more interest are

- $S_{12}^0 = L(0)$
- $S_{12}^1 = L(0)$
- $S_{12}^2 = L(0(0 \cup 1)^*)$
- $S_{12}^3 = L(0(0 \cup 1)^* \cup 11^*0(0 \cup 1)^*)$
- $S_{13}^0 = L(1)$
- $S_{13}^1 = L(1)$
- $S_{13}^2 = L(1)$
- $S_{13}^3 = L(11^*)$
- $S_{22}^0 = L(0 \cup 1)$
- $S_{22}^1 = L((0 \cup 1)^*)$
- $S_{22}^2 = L((0 \cup 1)^*)$
- $S_{22}^3 = L((0 \cup 1)^*)$
- $S_{32}^0 = L(0)$
- $S_{32}^1 = L(0)$
- $S_{32}^2 = L(0(0 \cup 1)^*)$
- $S_{32}^3 = L(1^*0(0 \cup 1)^*)$

Exercise: Find a Regular Expression equivalent to the following DFA:


1
2
3
4
1
0
0
0, 1
1
1
0