1 Equivalence of Finite Automata and Regular Expressions

Finite Automata Recognize Regular Languages

Theorem 1. *L is a regular language iff there is a regular expression R such that $\mathbf{L}(R) = L$ iff there is a DFA M such that $\mathbf{L}(M) = L$ iff there is a NFA N such that $\mathbf{L}(N) = L$.*

i.e., regular expressions, DFAs and NFAs have the same computational power.

Proof.
- Given regular expression R, will construct *NFA* N such that $\mathbf{L}(N) = \mathbf{L}(R)$
- Given *DFA* M, will construct regular expression R such that $\mathbf{L}(M) = \mathbf{L}(R)$ □

---

2 Regular Expressions to NFA

Regular Expressions to Finite Automata
... to Non-determinstic Finite Automata

Lemma 2. *For any regex R, there is an NFA N_R s.t. $\mathbf{L}(N_R) = \mathbf{L}(R)$.*

Proof Idea
We will build the NFA N_R for R, inductively, based on the number of operators in R, $\#(R)$.

- *Base Case:* $\#(R) = 0$ means that R is $\emptyset, \epsilon$, or a (from some $a \in \Sigma$). We will build NFAs for these cases.
- *Induction Hypothesis:* Assume that for regular expressions R, with $\#(R) < n$, there is an NFA N_R s.t. $\mathbf{L}(N_R) = \mathbf{L}(R)$.
- *Induction Step:* Consider R with $\#(R) = n$. Based on the form of R, the NFA N_R will be built using the induction hypothesis.

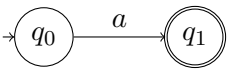
Regular Expression to NFA

Base Cases
If R is an elementary regular expression, NFA N_R is constructed as follows.

$R = \emptyset$

$R = \epsilon$

$R = a$



Induction Step: Union

Case $R = R_1 \cup R_2$
By induction hypothesis, there are N_1, N_2 s.t. $\mathbf{L}(N_1) = \mathbf{L}(R_1)$ and $\mathbf{L}(N_2) = \mathbf{L}(R_2)$. Build NFA N s.t. $\mathbf{L}(N) = \mathbf{L}(N_1) \cup \mathbf{L}(N_2)$

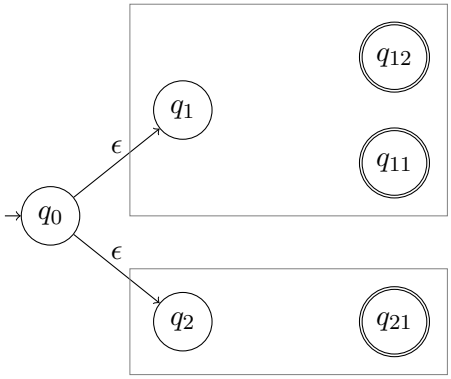


Figure 1: NFA for $\mathbf{L}(N_1) \cup \mathbf{L}(N_2)$

Induction Step: Union
Formal Definition

Case $R = R_1 \cup R_2$
Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ and $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ (with $Q_1 \cap Q_2 = \emptyset$) be such that $\mathbf{L}(N_1) = \mathbf{L}(R_1)$ and $\mathbf{L}(N_2) = \mathbf{L}(R_2)$. The NFA $N = (Q, \Sigma, \delta, q_0, F)$ is given by

- $Q = Q_1 \cup Q_2 \cup \{q_0\}$, where $q_0 \notin Q_1 \cup Q_2$
- $F = F_1 \cup F_2$
- δ is defined as follows

$$\delta(q,a) = \begin{cases} \delta_1(q,a) & \text{if } q \in Q_1 \\ \delta_2(q,a) & \text{if } q \in Q_2 \\ \{q_1, q_2\} & \text{if } q = q_0 \text{ and } a = \epsilon \\ \emptyset & \text{otherwise} \end{cases}$$

Induction Step: Union
Correctness Proof

Need to show that $w \in \mathbf{L}(N)$ iff $w \in \mathbf{L}(N_1) \cup \mathbf{L}(N_2)$.

- $\Rightarrow$ $w \in \mathbf{L}(N)$ implies $q_0 \xrightarrow{w}_N q$ for some $q \in F$. Based on the transitions out of q_0, $q_0 \xrightarrow{\epsilon}_N q_1 \xrightarrow{w}_N q$ or $q_0 \xrightarrow{\epsilon}_N q_2 \xrightarrow{w}_N q$. Consider $q_0 \xrightarrow{\epsilon}_N q_1 \xrightarrow{w}_N q$. (Other case is similar) This means $q_1 \xrightarrow{w}_{N_1} q$ (as N has the same transition as N_1 on the states in Q_1) and $q \in F_1$. This means $w \in \mathbf{L}(N_1)$.

$\Leftarrow$ $w \in \mathbf{L}(N_1) \cup \mathbf{L}(N_2)$. Consider $w \in \mathbf{L}(N_1)$; case of $w \in \mathbf{L}(N_2)$ is similar. Then, $q_1 \xrightarrow{w}_{N_1} q$ for some $q \in F_1$. Thus, $q_0 \xrightarrow{\epsilon}_N q_1 \xrightarrow{w}_N q$, and $q \in F$. This means that $w \in \mathbf{L}(N)$.

Induction Step: Concatenation

Case $R = R_1 \circ R_2$

- By induction hypothesis, there are N_1, N_2 s.t. $\mathbf{L}(N_1) = \mathbf{L}(R_1)$ and $\mathbf{L}(N_2) = \mathbf{L}(R_2)$
- Build NFA N s.t. $\mathbf{L}(N) = \mathbf{L}(N_1) \circ \mathbf{L}(N_2)$

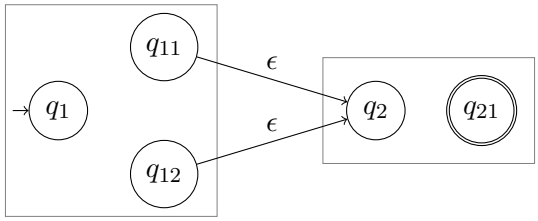


Figure 2: NFA for $\mathbf{L}(N_1) \circ \mathbf{L}(N_2)$

Induction Step: Concatenation
Formal Definition

Case $R = R_1 \circ R_2$
Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ and $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ (with $Q_1 \cap Q_2 = \emptyset$) be such that $\mathbf{L}(N_1) = \mathbf{L}(R_1)$ and $\mathbf{L}(N_2) = \mathbf{L}(R_2)$. The NFA $N = (Q, \Sigma, \delta, q_0, F)$ is given by

- $Q = Q_1 \cup Q_2$
- $q_0 = q_1$
- $F = F_2$
- δ is defined as follows

$$\delta(q,a) = \begin{cases} \delta_1(q,a) & \text{if } q \in (Q_1 \setminus F_1) \text{ or } a \neq \epsilon \\ \delta_1(q,a) \cup \{q_2\} & \text{if } q \in F_1 \text{ and } a = \epsilon \\ \delta_2(q,a) & \text{if } q \in Q_2 \\ \emptyset & \text{otherwise} \end{cases}$$

Induction Step: Concatenation
Correctness Proof

Need to show that $w \in \mathbf{L}(N)$ iff $w \in \mathbf{L}(N_1) \circ \mathbf{L}(N_2)$.

$w \in \mathbf{L}(N)$ iff $q_0 \xrightarrow{w}_N q$ for some $q \in F = F_2$. The computation of N on w starts in a state of N_1 (namely, $q_0 = q_1$) and ends in a state of N_2 (namely, $q \in F_2$). The only transitions from a state of N_1 to a state of N_2 is from a state in F_1 which have ϵ-transitions to q_2, the initial state of N_2. Thus, we have

$$q_0 = q_1 \xrightarrow{w}_N q \text{ with } q \in F = F_2$$
$$\text{iff}$$
$$\exists q' \in F_1.\ \exists u, v \in \Sigma^*.\ w = uv \text{ and } q_0 = q_1 \xrightarrow{u}_N q' \xrightarrow{\epsilon}_N q_2 \xrightarrow{v}_N q$$

This means that $q_1 \xrightarrow{u}_{N_1} q'$ (with $q' \in F_1$) and $q_2 \xrightarrow{v}_{N_2} q$ (with $q \in F_2$). Hence, $u \in \mathbf{L}(N_1)$ and $v \in \mathbf{L}(N_2)$, and so $w = uv \in \mathbf{L}(N_1) \circ \mathbf{L}(N_2)$. Conversely, if $u \in \mathbf{L}(N_1)$ and $v \in \mathbf{L}(N_2)$ then for some $q' \in F_1$ and $q \in F_2$, we have $q_1 \xrightarrow{u}_{N_1} q'$ and $q_2 \xrightarrow{v}_{N_2} q$. Then,

$$q_0 = q_1 \xrightarrow{u}_N q' \xrightarrow{\epsilon}_N q_2 \xrightarrow{v}_N q$$

Thus, $q_0 \xrightarrow{w=uv}_N q$ and so $uv \in \mathbf{L}(N)$.

Induction Step: Kleene Closure
First Attempt

Case $R = R_1^*$

- By induction hypothesis, there is N_1 s.t. $\mathbf{L}(N_1) = \mathbf{L}(R_1)$
- Build NFA N s.t. $\mathbf{L}(N) = (\mathbf{L}(N_1))^*$

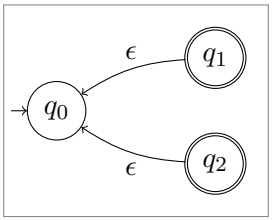


Figure 3: NFA accepts $(\mathbf{L}(N_1))^+$

Problem: May not accept ϵ! One can show that $\mathbf{L}(N) = (\mathbf{L}(N_1))^+$.

Induction Step: Kleene Closure
Second Attempt

Case $R = R_1^*$

- By induction hypothesis, there is N_1 s.t. $\mathbf{L}(N_1) = \mathbf{L}(R_1)$
- Build NFA N s.t. $\mathbf{L}(N) = (\mathbf{L}(N_1))^*$

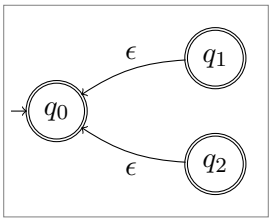


Figure 4: NFA accepts $\supseteq (\mathbf{L}(N_1))^*$

Problem: May accept strings that are not in $(\mathbf{L}(N_1))^*$!

Example demonstrating the problem

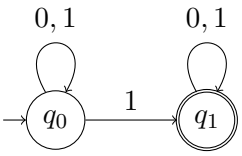


Figure 5: Example NFA N

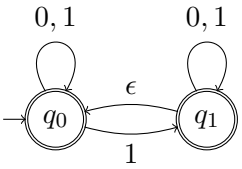


Figure 6: Incorrect Kleene Closure of N

$\mathbf{L}(N) = (0 \cup 1)^*1(0 \cup 1)^*$. Thus, $(\mathbf{L}(N))^* = \epsilon \cup (0 \cup 1)^*1(0 \cup 1)^*$. The previous construction, gives an NFA that accepts $0 \notin (\mathbf{L}(N))^*$!

Induction Step: Kleene Closure
Correct Construction

Case $R = R_1^*$

- First build N_1 s.t. $\mathbf{L}(N_1) = \mathbf{L}(R_1)$
- Given N_1 build NFA N s.t. $\mathbf{L}(N) = \mathbf{L}(N_1)^*$

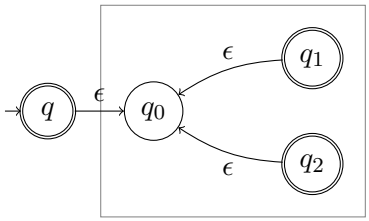


Figure 7: NFA for $\mathbf{L}(N_1)^*$

Induction Step: Kleene Closure
Formal Definition

Case $R = R_1^*$
Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ be such that $\mathbf{L}(N_1) = \mathbf{L}(R_1)$. The NFA $N = (Q, \Sigma, \delta, q_0, F)$ is given by

- $Q = Q_1 \cup \{q_0\}$ with $q_0 \notin Q_1$
- $F = F_1 \cup \{q_0\}$
- δ is defined as follows

$$\delta(q,a) = \begin{cases} \delta_1(q,a) & \text{if } q \in (Q_1 \setminus F_1) \text{ or } a \neq \epsilon \\ \delta_1(q,a) \cup \{q_1\} & \text{if } q \in F_1 \text{ and } a = \epsilon \\ \{q_1\} & \text{if } q = q_0 \text{ and } a = \epsilon \\ \emptyset & \text{otherwise} \end{cases}$$

Induction Step: Kleene Closure
Correctness Proof

Let us begin by stating what our goal is. We would like to show $w \in \mathbf{L}(N)$ iff $w \in L^*$. If we choose to prove this statement by induction, most induction proofs will fail because this statement is too weak to be established by induction. How we choose to strengthen it depends on what parameter we will choose to induct over. One possibility is $|w|$. If we do induction on the length of w, then we need to strengthen the statement by saying which strings are accepted from any state $q \in Q$, and not just the initial state q_0 as in the above statement. We can carry such a proof out, but it is long. We instead present a proof that does induction over a parameter different than length of w, but before presenting this proof we need to introduce some notation and terminology that we will find convenient.

Observe that we construct N from N_1 by adding some ϵ-transitions: one from q_0 to q_1, and others from $q \in F_1$ to q_1. We will call these "new" transitions. Recall that an accepting computation is a sequence of steps starting from the initial state q_0 and ending in some accept state, such that every step conforms to the transition relation. Let us call a computation ρ as having n new steps, if exactly n steps in ρ are according to the new ϵ-transitions. For any n, let us define

$$A_n = \{w \in \Sigma^* \mid w \text{ has an accepting computation where exactly } n \text{ new transitions are used}\}$$

Observe that if w has an accepting computation then $w \in A_n$ for some $n \geq 0$.

We will prove by induction on n, the following statement

$$\forall n \in \mathbb{N}.\ w \in A_n \text{ iff } w \in L^n$$

Before proving the above stronger statement by induction, let us see how proving the above statement establishes the correctness of the construction. Suppose $w \in L^*$ then (by definition of Kleene closure) $w \in L^i$ for some $i \in \mathbb{N}$. By the above statement, it would mean that $w \in A_i$. In other words, w has an accepting computation that uses exactly i new transitions, which just implies that N accepts w. On the other hand, suppose N accepts w. Since N has an accepting computation on w, it must have an accepting computation that uses exactly i new transitions, for some value of i. In other words, $w \in A_i$. By the above statement that means that $w \in L^i$ which implies that $w \in L^*$. Thus we can establish both sides of the correctness claim.

Let us now prove by induction on n

$$\forall n \in \mathbb{N}.\ w \in A_n \text{ iff } w \in L^n$$

Base Case For this statement we need to establish two base cases: one when $n = 0$ and the other when $n = 1$.

Case 1: Let $n = 0$. Since the only transition out of the initial state q_0 is a new transition, $w \in A_0$ means that the computation takes no steps and stays in q_0. If the computation on w has no transition steps, it means that $w = \epsilon$ and clearly $w \in L^0$. On the other hand, if $w \in L^0$ then $w = \epsilon$ and N accepts w by taking no steps as $q_0 \in F$. Thus, we have established the base case for $n = 0$.

Case 2: Let $n = 1$. Suppose $w \in L$, then N_1 has an accepting computation. Thus, there is $q \in F_1$ such that $q_1 \xrightarrow{w}_{N_1} q$. Observe that since every transition of N_1 is a transition of N (which is not new), and $F_1 \subseteq F$ we have the following accepting computation of N with exactly one new transition

$$q_0 \xrightarrow{\epsilon}_N q_1 \xrightarrow{w}_{N_1} q$$

Thus $w \in A_1$. Conversely, suppose $w \in A_1$. Again, the only transition out of q_0 is a new transition. Thus the accepting computation of N on w must be of the form

$$q_0 \xrightarrow{\epsilon}_N q_1 \xrightarrow{w}_{N_1} q$$

for some $q \in F_1$; the reason that q must be in F_1 is because q_0 (the only other accept state) has no incoming transitions. Thus, $q_1 \xrightarrow{w}_{N_1} q$ for $q \in F_1$, which means that w is accepted by N_1, and from the fact that $\mathbf{L}(N_1) = L$, we can conclude that $w \in L = L^1$. We have, therefore, established the base case for $n = 1$.

Ind. Hyp. Assume that for all $i < n$, $w \in A_i$ iff $w \in L^i$, where $n > 1$.

Ind. Step Suppose $w \in L^n$. Then there are u, v such that $w = uv$, $u \in L^{n-1}$ and $v \in L$. By induction hypothesis, we have $u \in A_{n-1}$. Now, since $n > 1$, the accepting computation on u must end in a state $q \in F_1$ (because once you leave q_0 you can never get back to it). Moreover since $v \in L$, from the correctness of N_1, we have $q_1 \xrightarrow{v}_{N_1} q'$ for some $q' \in F_1$. Putting all of this together we have the following accepting computation

$$q_0 \xrightarrow{u}_N q \xrightarrow{\epsilon}_N q_1 \xrightarrow{v}_{N_1} q'$$

which has exactly n new transitions. Thus, $w \in A_n$. To prove the converse, suppose $w \in A_n$. Since $n > 1$, the accepting computation on w must be of the form

$$q_0 \xrightarrow{u}_N q \xrightarrow{\epsilon}_N q_1 \xrightarrow{v}_{N_1} q'$$

where $w = uv$, and q and q' are some states in F_1. Thus, $u \in A_{n-1}$. By induction hypothesis, we have $u \in L^{n-1}$. From the correctness of N_1, $q_1 \xrightarrow{v}_{N_1} q'$ for $q' \in F_1$ means that $v \in L$. Putting this together, we get that $w = uv \in (L^{n-1})L = L^n$.

Regular Expressions to NFA
To Summarize

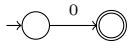
We built an NFA N_R for each regular expression R inductively

- When R was an elementary regular expression, we gave an explicit construction of an NFA recognizing $\mathbf{L}(R)$
- When $R = R_1 \text{ op } R_2$ (or $R = \text{op}(R_1)$), we constructed an NFA N for R, using the NFAs for R_1 and R_2.

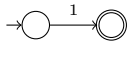
Regular Expressions to NFA
An Example

Build NFA for $(1 \cup 01)^*$

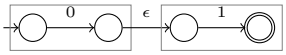
N_0



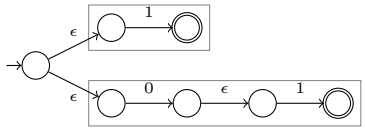
N_1



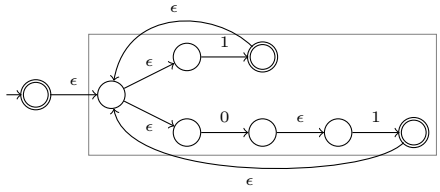
N_{01}



$N_{1\cup01}$



$N_{(1\cup01)^*}$



---

3 DFAs to Regular Expressions

DFA to Regular Expression

- Given DFA M, will construct regular expression R such that $L(M) = L(R)$. *In two steps:*
  - Construct a "Generalized NFA" (GNFA) G from the DFA M
  - And then convert G to a regex R

3.1 Generalized NFA

Generalized NFA

- A GNFA is similar to an NFA, but:
  - There is a single accept state which is not the start state.
  - The start state has no incoming transitions, and the accept state has no outgoing transitions.
    - These are "cosmetic changes": Any NFA can be converted to an equivalent NFA of this kind.
  - The transitions are labeled not by characters in the alphabet, but by *regular expressions.*
    - For *every* pair of states (q_1, q_2), the transition from q_1 to q_2 is labeled by a regular expression $\rho(q_1, q_2)$.
  - "Generalized NFA" because a normal NFA has transitions labeled by ϵ, elements in Σ (a union of elements, if multiple edges between a pair of states) and $\emptyset$ (missing edges).

Generalized NFA

- Transition: GNFA *non-deterministically* reads a block of characters from the input, chooses an edge from the current state q_1 to another state q_2, and if the block of symbols matches the regex $\rho(q_1, q_2)$, then moves to q_2.
- Acceptance: G accepts w if there exists some sequence of valid transitions such that on starting from the start state, and after finishing the entire input, G is in the accept state.

Generalized NFA: Example

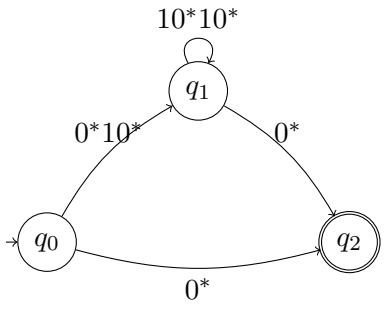


Figure 8: Example GNFA G

Accepting run of G on 11110100 is $q_0 \xrightarrow{1}_G q_1 \xrightarrow{11}_G q_1 \xrightarrow{101}_G q_1 \xrightarrow{00}_G q_2$

Generalized NFA: Definition

Definition 3. A generalized nondeterministic finite automaton (GNFA) is $G = (Q, \Sigma, q_0, q_F, \rho)$, where

- Q is the finite set of states
- Σ is the finite alphabet
- $q_0 \in Q$ initial state
- $q_F \in (Q \setminus \{q_0\}$, a single accepting state
- $\rho : (Q \setminus \{q_F\}) \times (Q \setminus \{q_0\}) \to \mathcal{R}_\Sigma$, where $\mathcal{R}_\Sigma$ is the set of all regular expressions over the alphabet Σ

Generalized NFA: Definition

Definition 4. For a GNFA $M = (Q, \Sigma, q_0, q_F, \rho)$ and string $w \in \Sigma^*$, we say M *accepts* w iff there exist $x_1, \ldots, x_t \in \Sigma^*$ and states $r_0, \ldots, r_t$ such that

- $w = x_1 x_2 x_3 \cdots x_t$
- $r_0 = q_0$ and $r_t = q_F$
- for each $i \in [1, t]$, $x_i \in \mathbf{L}(\rho(r_{i-1}, r_i))$,

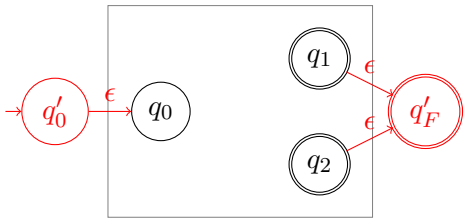
---

3.2 Converting DFA to GNFA

Converting DFA to GNFA

A DFA $M = (Q, \Sigma, \delta, q_0, F)$ can be easily converted to an equivalent GNFA $G = (Q', \Sigma, q_0', q_F', \rho)$:

- $Q' = Q \cup \{q_0', q_F'\}$ where $Q \cap \{q_0', q_F'\} = \emptyset$
- $\rho(q_1, q_2) = \begin{cases} \epsilon, & \text{if } q_1 = q_0' \text{ and } q_2 = q_0 \\ \epsilon, & \text{if } q_1 \in F \text{ and } q_2 = q_F' \\ \bigcup_{\{a \mid \delta(q_1, a) = q_2\}} a & \text{otherwise} \end{cases}$

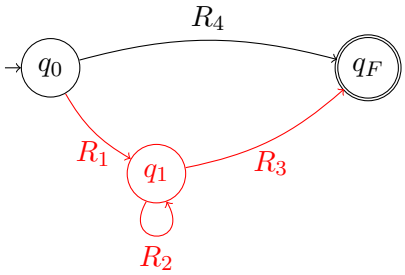


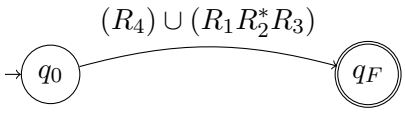
Prove: $\mathbf{L}(G) = \mathbf{L}(M)$.

3.3 Converting GNFA to Regular Expression

GNFA to Regex

- Suppose G is a GNFA with only two states, q_0 and q_F.
- Then $\mathbf{L}(R) = \mathbf{L}(G)$ where $R = \rho(q_0, q_F)$.
- How about G with three states?





- Plan: Reduce any GNFA G with $k > 2$ states to an equivalent GFA with $k - 1$ states.

GNFA to Regex: From k states to $k - 1$ states

Definition 5 (Deleting a GNFA State)**.** Given GNFA $G = (Q, \Sigma, q_0, q_F, \rho)$ with $|Q| > 2$, and any state $q^* \in Q \setminus \{q_0, q_F\}$, define GNFA $\mathsf{rip}(G, q^*) = (Q', \Sigma, q_0, q_F, \rho')$ as follows:

- $Q' = Q \setminus \{q^*\}$.
- For any $(q_1, q_2) \in Q' \setminus \{q_F\} \times Q' \setminus \{q_0\}$ (possibly $q_1 = q_2$), let

$$\rho'(q_1, q_2) = (R_1 R_2^* R_3) \cup R_4,$$

  where $R_1 = \rho(q_1, q^*)$, $R_2 = \rho(q^*, q^*)$, $R_3 = \rho(q^*, q_2)$ and $R_4 = \rho(q_1, q_2)$.

GNFA to Regex: From k states to $k - 1$ states
Correctness

Proposition 6. *For any $q^* \in Q \setminus \{q_0, q_F\}$, G and $\mathsf{rip}(G, q^*)$ are equivalent.*

Proof. Let $G' = \mathsf{rip}(G, q^*)$. We need to show that $\mathbf{L}(G) = \mathbf{L}(G')$. We will prove this in two steps: we will show $\mathbf{L}(G) \subseteq \mathbf{L}(G')$ and then show $\mathbf{L}(G') \subseteq \mathbf{L}(G)$.
$\mathbf{L}(G) \subseteq \mathbf{L}(G')$: First we show $w \in \mathbf{L}(G) \implies w \in \mathbf{L}(G')$. $w \in \mathbf{L}(G) \implies \exists q_0 = r_0, r_1, \ldots, r_t = q_F$ and $x_1, \ldots, x_t \in \Sigma^*$ such that $w = x_1 x_2 x_3 \cdots x_t$ and for each i, $x_i \in L(\rho(r_{i-1}, r_i))$.

We need to show $y_1, \ldots, y_d \in \Sigma^*$ and $q_0 = s_0, s_1, \ldots, s_d = q_F$ such that $w = y_1 \cdots y_d$, and for each i, $y_i \in L(\rho'(s_{i-1}, s_i))$.

Define $(s_0 = q_0, \ldots, s_d = q_F)$ to be the sequence obtained by deleting all occurrences of q^* from $(r_0 = q_0, r_1, \ldots, r_t = q_F)$.

To formally define y_j, first we define σ as follows:

$$\sigma(j) = \begin{cases} 0 & \text{if } j = 0 \\ i & \text{if } 0 < \sigma(j-1) < t, \text{ where } i = \min_{i > \sigma(j-1)} (r_i \neq q^*) \\ \text{undefined} & \text{otherwise.} \end{cases}$$

The range of σ is the set of indices i such that $r_i \neq q^*$. Let $d = \min_k(\sigma(k) = t)$. Then, $s_j = r_{\sigma(j)}$, for $j = 0, \ldots, d$.

Now we define $y_j = x_{\sigma(j-1)+1} \cdots x_{\sigma(j)}$ for $j = 1, \ldots, d$

Then $y_1 \cdots y_d = x_1 \cdots x_t = w$.

We need to show that $y_j \in \mathbf{L}(\rho'(s_{j-1}, s_j))$ for all j. We consider the following cases for j:

- $\sigma(j) = \sigma(j-1) + 1$ (i.e., $r_{\sigma(j-1)+1} \neq q^*$). Then $y_j = x_i$ and $s_{j-1} = r_{i-1}$ and $s_j = r_i$, where $i = \sigma(j)$. $y_j = x_i \in \mathbf{L}(\rho(r_{i-1}, r_i)) \subseteq \mathbf{L}(\rho'(r_{i-1}, r_i)) = \mathbf{L}(\rho'(s_{j-1}, s_j))$.

- $\sigma(j) > \sigma(j-1)+1$ (i.e., $r_{\sigma(j-1)+1} = q^*$). Then $y_j = x_\ell \cdots x_i$ and $s_{j-1} = r_{\ell-1}$ and $s_j = r_i$, where $\ell = \sigma(j-1)+1$ and $i = \sigma(j)$.

$$\begin{aligned} y_j = x_\ell \cdots x_i &\in \mathbf{L}(\rho(r_{\ell-1}, r_\ell)\rho(r_\ell, r_{\ell+1}) \cdots \rho(r_{i-1}, r_i)\rho(r_i, r_{i+1})) \\ &= \mathbf{L}(\rho(r_{\ell-1}, q^*)\rho(q^*, q^*)^{i-\ell}\rho(q^*, r_i)) \\ &\subseteq \mathbf{L}(\rho(r_{\ell-1}, r_\ell)\rho(q^*, q^*)^*\rho(q^*, r_i)) \\ &\subseteq \mathbf{L}(\rho(s_{j-1}, q^*)\rho(q^*, q^*)^*\rho(q^*, s_j)) \\ &\subseteq \mathbf{L}(\rho'(s_{j-1}, s_j)) \end{aligned}$$

Thus $w \in \mathbf{L}(G')$ as we set out to prove.
$\mathbf{L}(G') \subseteq \mathbf{L}(G)$: Next we need to show that $w \in \mathbf{L}(G') \implies w \in \mathbf{L}(G)$. $w \in \mathbf{L}(G') \implies \exists q_0 = s_0, s_1, \ldots, s_d = q_F$ and $y_1, \ldots, y_d \in \Sigma^*$ such that $w = y_1 y_2 y_3 \cdots y_d$ and for each j, $y_j \in \mathbf{L}(\rho'(s_{j-1}, s_j)) = \mathbf{L}\left((\rho(s_{j-1}, q^*)\rho(q^*, q^*)^*\rho(q^*, r_i)) \cup \rho(s_{j-1}, s_j)\right)$

Define σ as follows, for $j = 0, \ldots, d$:

$$\sigma(j) = \begin{cases} 0 & \text{if } j = 0, \\ \sigma(j-1)+1 & \text{if } y_j \in \mathbf{L}(\rho(s_{j-1}, s_j)) \\ \sigma(j-1)+u+2 & \text{otherwise, where } u = \min_v \left(y_j \in \mathbf{L}(\rho(s_{j-1}, q^*)\rho(q^*, q^*)^v\rho(q^*, s_j))\right) \end{cases}$$

Let $t = \sigma(d)$. For $i = 0, \ldots, t$ define r_i as follows:

$$r(i) = \begin{cases} s_j & \text{if there exists } j \text{ such that} i = \sigma(j), \\ q^* & \text{otherwise.} \end{cases}$$

Finally, define x_i $(i = 1, \ldots, t)$ as follows: if $i = \sigma(j)$ and $i - 1 = \sigma(j-1)$, then let $x_i = y_j$. For other i $(\sigma(j-1) < i - 1 < i \leq \sigma(j)$ for some $j)$, we have $y_j \in \mathbf{L}(\rho(s_{j-1}, q^*)\rho(q^*, q^*)^u\rho(q^*, s_j))$ where $u = \sigma(j) - \sigma(j-1) - 2$. Therefore we can write $y_j = x_\ell \cdots x_{\sigma(j)}$, where $\ell = \sigma(j-1)+1$, such that $x_\ell \in \mathbf{L}(\rho(s_{j-1}, q^*))$, $x_{\sigma(j)} \in \mathbf{L}(\rho(q^*, s_j))$ and $x_{\ell+1}, \ldots, x_{\sigma(j)-1} \in \mathbf{L}(\rho(q^*, q^*))$. Verify that all x_i $(i = 1, \ldots, t)$ are well-defined by this.

With these definitions it can be easily verified that $x_0 \cdots x_t = y_0 \cdots y_d = w$ and $x_i \in \mathbf{L}(\rho(r_{i-1}, r_i))$. □

DFA to Regex: Summary

Lemma 7. *For every DFA M, there is a regular expression R such that $\mathbf{L}(M) = \mathbf{L}(R)$.*

- Any DFA can be converted into an equivalent GNFA. So let G be a GNFA s.t. $\mathbf{L}(M) = \mathbf{L}(G)$.
- For any GNFA $G = (Q, \Sigma, q_0, q_F, \rho)$ with $|Q| > 2$, for any $q^* \in Q \setminus \{q_0, q_F\}$, G and $\mathsf{rip}(G, q^*)$ are equivalent. $\mathsf{rip}(G, q^*)$ has one fewer state than G.
- So given G, by applying rip repeatedly (choosing q^* arbitrarily each time), we can get a GNFA G' with two states s.t. $\mathbf{L}(G) = \mathbf{L}(G')$. Formally, by induction on the number of states in G.
- For a 2-state GNFA G', $\mathbf{L}(G') = \mathbf{L}(R)$, where $R = \rho(q_0, q_F)$.

DFA to Regex: Example

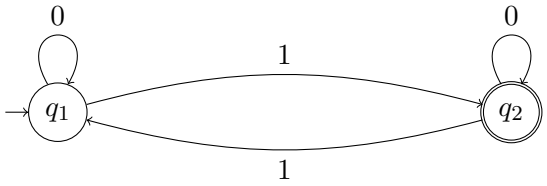


Figure 9: Example DFA D

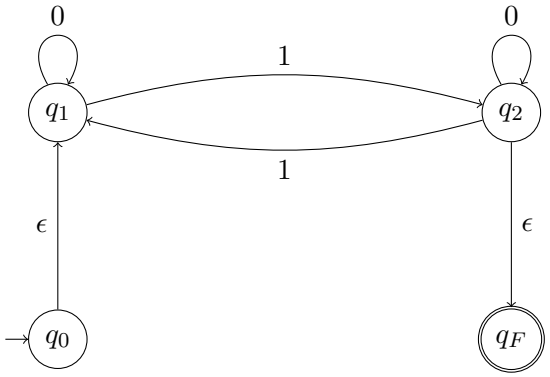


Figure 10: GNFA G equivalent to D, ignoring transitions labelled $\emptyset$

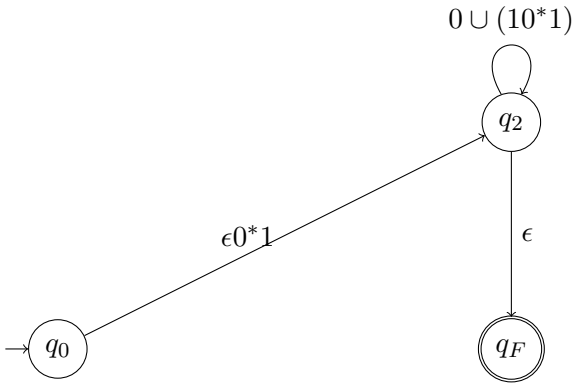


Figure 11: Ripping q_1

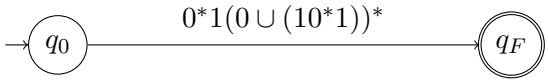


Figure 12: Ripping q_2
