Computer Science 313

Regular Expressions, Part Two

Instructor: Wayne Eberly

Department of Computer Science
University of Calgary

Lecture #9

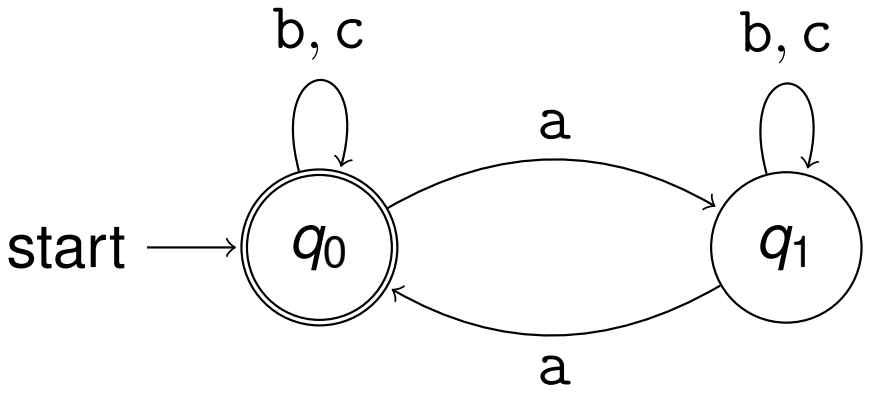
Goal for Today

- Prove that every regular language is the language of a regular expression — by giving a ***construction*** that can be used to convert any given DFA or NFA into a regular expression that has the same language.
- ***Reference:*** *Introduction to the Theory of Computation*, Section 1.3.

  The material in today's class is essentially the same as Lemma 1.60 (pages 69–76).This reference only describes a construction to produce a regular expression from a ***deterministic*** finite automaton, but almost the same construction can be used to produce a regular expression from a ***nondeterministic*** finite automaton too.

A First Example

This construction will be used to produce a regular expression for the language of the following DFA over the alphabet $\Sigma_1 = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$.

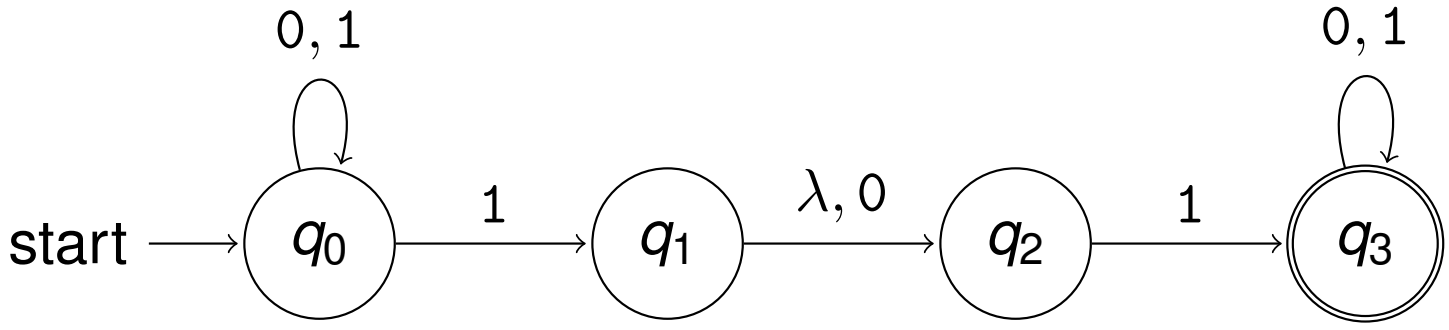


The language L_1 of this DFA is

$$L_1 = \{\omega \in \Sigma_1^\star \mid \omega \text{ includes an even number of a's}\}.$$

A Second Example

This construction will be used to produce a regular expression for the language of the following NFA over the alphabet $\Sigma_2 = \{0, 1\}$.



The language L_2 of this NFA is

$$L_2 = \{\omega \in \Sigma_2^\star \mid \text{either } \texttt{11} \text{ or } \texttt{101} \text{ (or both) is a substring of } \omega.\}.$$

Overview

Steps in This Construction:

1. Convert the given DFA or NFA into a ***generalized nondeterministic finite automaton*** (as defined next) with the same language $L \subseteq \Sigma^\star$
2. Repeatedly eliminate a state from the GNFA, without changing its language, L, until only two states remain.
3. Read a regular expression with the same language L from the GNFA.

Generalized Nondeterministic Finite Automata

A ***generalized nondeterministic finite automaton*** M is yet another kind of "finite state machine:"

- As usual, M has a finite set Q of ***states*** — which includes a ***start state*** q_0
- Q also includes a single ***accepting state*** q_A, which is different from q_0.
- For every state $q \in Q \setminus \{q_A\}$ (that is, for each state except q_A) and for every state $r \in Q \setminus \{q_0\}$ (that is, for every state except q_0) there is a transition from q to r that is labelled by some ***regular expression*** $R_{q,r}$ over the alphabet Σ.

Generalized Nondeterministic Finite Automata

- In other words, the ***transition function*** is a total function

$$\delta : (Q \setminus \{q_A\}) \times (Q \setminus \{q_0\}) \to \mathcal{R}_\Sigma$$

  where $\mathcal{R}_\Sigma$ is the set of regular expressions over the alphabet Σ.

- The GNFA M ***accepts*** a string $\omega \in \Sigma$ if and only if there is a sequence

$$r_0, r_1, r_2, \ldots, r_m$$

  of the states in Q such that $r_0 = q_0$, $r_1, r_2, \ldots, r_{m-1} \in Q \setminus \{q_0, q_A\}$, $r_m = q_A$, and

$$\omega = \omega_0 \omega_1 \ldots, \omega_{m-1}$$

  where ω_i is in the language of the regular expression $R_{i,i+1} = \delta(r_i, r_{i+1})$ labelling the transition from r_i to r_{i+1}, for $0 \leq i \leq m - 1$.

Converting a Given DFA or NFA into a GNFA

To convert a DFA $M = (Q, \sigma, \delta, q_0, F)$ into a GNFA $\widehat{M} = (\widehat{Q}, \Sigma, \widehat{\delta}, \widehat{q}_0, q_A)$ with the same language,

1. Rename states in Q if necessary to make sure that no state in Q is named either q_0 or q_A — changing the name of the original start state to $\widehat{q}_0$; then set $\widehat{Q} = Q \cup \{q_0, q_A\}$.

   The new state, "q_0," will now be used as the start state of the GNFA being defined, and the new state, "q_A" that has been introduced will be the accepting state.

Converting a Given DFA or NFA into a GNFA

2. For every state $r \in Q \cup \{q_A\}$, set

$$\widehat{\delta}(q_0, r) = \begin{cases} \lambda & \text{if } r = \widehat{q}_0, \\ \emptyset & \text{otherwise} \end{cases}$$

— recalling that $\widehat{q}_0$ was the start state of the DFA that we started with.

3. For each pair of states $r, s \in Q$, set $\widehat{\delta}(r, s)$ to be $\emptyset$ if there are no symbols $\sigma \in \Sigma$ such that $s = \delta(r, \sigma)$. Otherwise, if $\{\sigma_1, \sigma_2, \ldots, \sigma_k\}$ is the set of symbols $\sigma \in \Sigma$ such that $\delta(r, \sigma) = s$, set $\widehat{\delta}(r, s)$ to be

$$(\sigma_1 \cup \sigma_2 \cup \cdots \cup \sigma_k).$$

The brackets can be deleted if $k = 1$.

Converting a Given DFA or NFA into a GNFA

4. Finally, for every state $q \in Q$, set

$$\widehat{\delta}(q, q_{\mathrm{A}}) = \begin{cases} \lambda & \text{if } q \in F, \\ \emptyset & \text{if } q \notin F. \end{cases}$$

It is possible to prove the following (by induction on the length $|\omega|$ of the string ω): For every string $\omega \in \Sigma^\star$, it is possible to reach the state q_A by processing (all of) the string ω, using the GNFA that has been produced, if and only if $\omega \in L(M)$.

That is: The DFA that you started with and the GNFA that you produced have the same language.

Converting a Given DFA or NFA into a GNFA

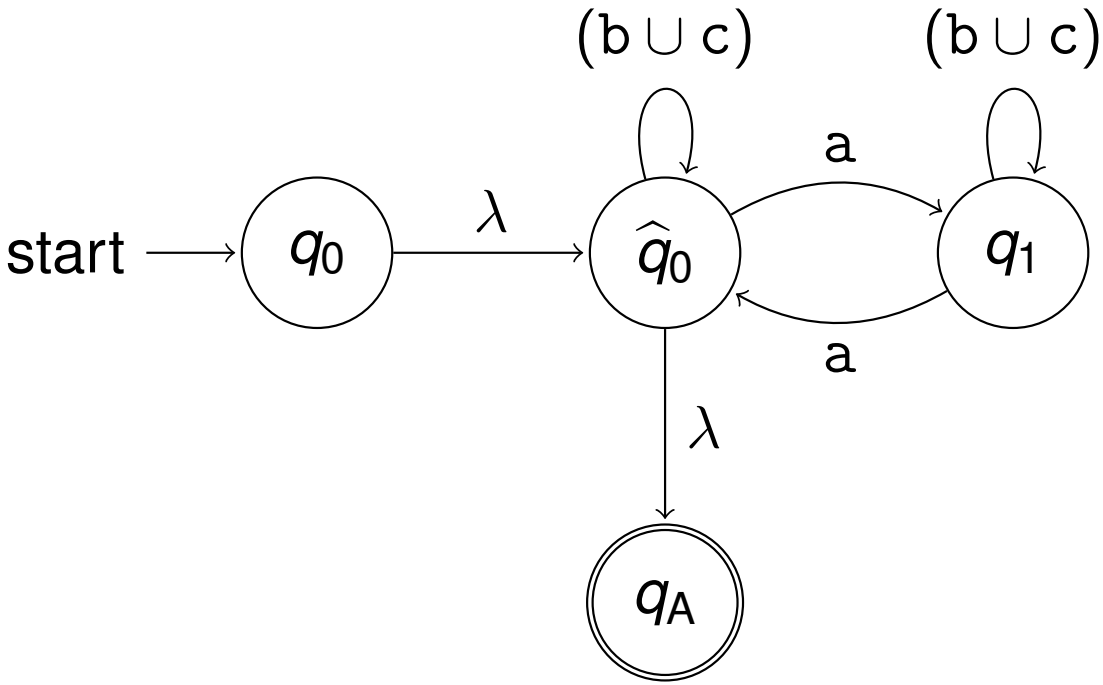
Example: This can be used to convert the DFA M_1 into a GNFA $\widehat{M} = (\widehat{Q}, \widehat{\delta}, q_0, q_A)$ such that

- $\widehat{Q} = \{q_0, \widehat{q}_0, q_1, q_A\}$.
- The transition function $\widehat{\delta}$ is as follows: The contents of the cell in the row for a state r and the column for a state s is the regular expression $\widehat{\delta}(r, s)$ for the transition from r to s:

	$\widehat{q}_0$	q_1	q_A
q_0	λ	$\emptyset$	$\emptyset$
$\widehat{q}_0$	$(\mathtt{b} \cup \mathtt{c})$	$\mathtt{a}$	λ
q_1	$\mathtt{a}$	$(\mathtt{b} \cup \mathtt{c})$	$\emptyset$

Converting a Given DFA or NFA into a GNFA

A picture of this GNFA is as follows. To simplify the drawing transitions (from q_0 to each of q_1 and q_A and from q_1 to q_A) with label $\emptyset$ are not shown.



Converting a Given DFA or NFA into a GNFA

Producing a GNFA from an NFA is done in almost the same way: Step 3 should be replaced by the following.

3. For each pair of states $r, s \in Q$, set $\widehat{\delta}(r, s)$ to be $\emptyset$ if there are no symbols $\sigma \in \Sigma_\lambda$ such that $s \in \delta(r, \sigma)$. Otherwise, if $\{\sigma_1, \sigma_2, \ldots, \sigma_k\}$ is the set of symbols $\sigma \in \Sigma_\lambda$ such that $s \in \delta(r, \sigma)$, set $\widehat{\delta}(r, s)$ to be

$$(\sigma_1 \cup \sigma_2 \cup \cdots \cup \sigma_k).$$

   Once again, the brackets can be deleted if $k = 1$.

It is possible to prove the following (by induction on the length $|\omega|$ of the string ω): For every string $\omega \in \Sigma^\star$, it is possible to reach the state q_A by processing (all of) the string ω, using the GNFA that has been produced, if and only if $\omega \in L(M)$.

That is: The NFA that you started with and the GNFA that you produced have the same language.

Converting a Given DFA or NFA into a GNFA

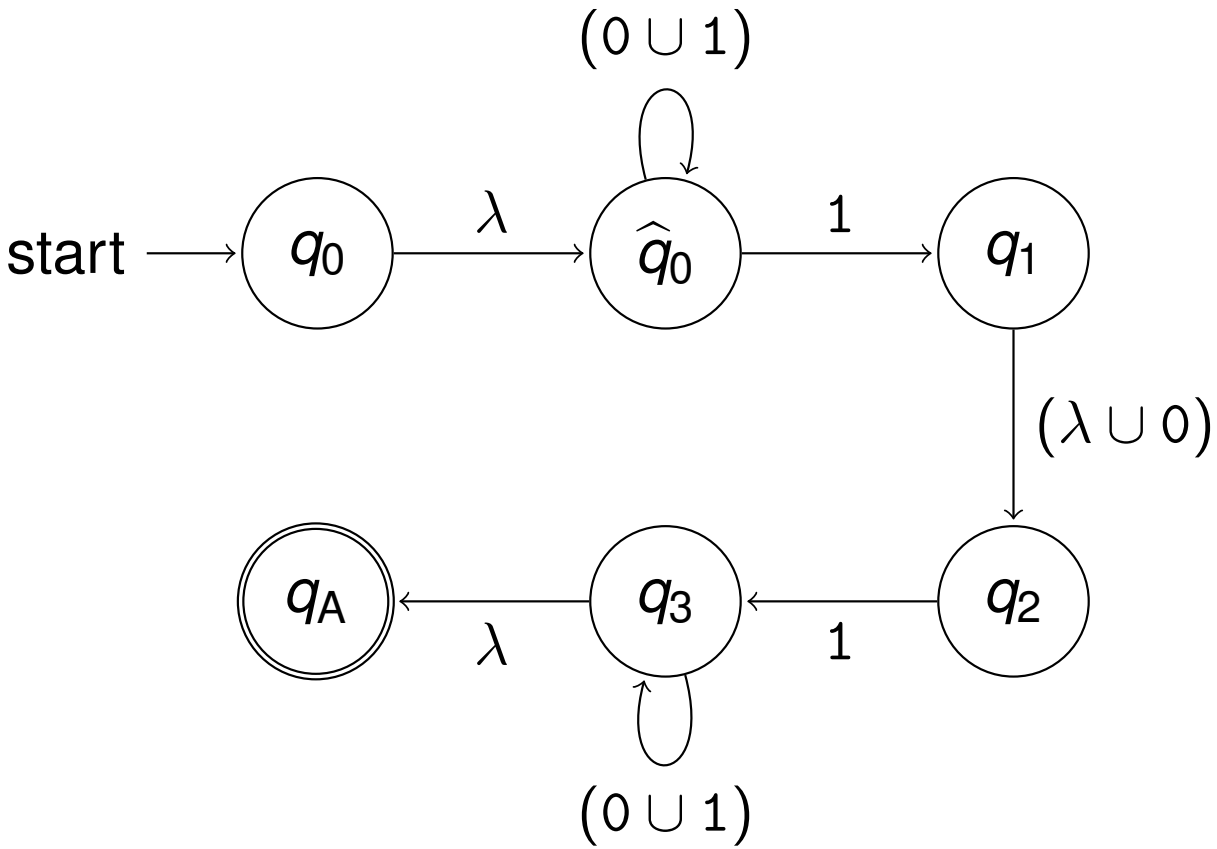
Example: This can be used to convert the NFA M_2 into a GNFA $\widehat{M} = (\widehat{Q}, \widehat{\delta}, q_0, q_A)$ such that

- $\widehat{Q} = \{q_0, \widehat{q}_0, q_1, q_2, q_3, q_A\}$;
- The transition function $\widehat{\delta}$ is as follows: The contents of the cell in the row for a state r and the column for a state s is the regular expression $\widehat{\delta}(r, s)$ for the transition from r to s:

	$\widehat{q}_0$	q_1	q_2	q_3	q_A
q_0	λ	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$\widehat{q}_0$	$(0 \cup 1)$	1	$\emptyset$	$\emptyset$	$\emptyset$
q_1	$\emptyset$	$\emptyset$	$(\lambda \cup 0)$	$\emptyset$	$\emptyset$
q_2	$\emptyset$	$\emptyset$	$\emptyset$	1	$\emptyset$
q_3	$\emptyset$	$\emptyset$	$\emptyset$	$(0 \cup 1)$	λ

Converting a Given DFA or NFA into a GNFA

A picture of the GNFA that has been produced is as follows. Once again, the picture has been simplified by leaving out the transitions whose label is $\emptyset$.



Eliminating States

- To continue we must repeatedly remove a state — that is neither the start state nor the accepting state — until only these two states are left.
- It is necessary to change the regular expressions that label the remaining transitions when we do this, in order to make sure that the language of the GNFA has not been changed.
- Suppose, in particular, that we are removing a state q. Let r and s be states (possibly the same!) that are both different from q.

Eliminating States

The GNFA can move from r to s, by processing some string $\mu \in \Sigma^\star$...

- ***directly*** by following the transition from r to s. This can happen if and only if μ is in the language of the regular expression $R_{r,s} = \widehat{\delta}(r, s)$.
- ***indirectly*** by following the transition from r to the state q that is about to be deleted, following the transition from q to itself zero or more times, and then following the transition from q to s. This can happen if and only if ω is in the language of the regular expression $(R_{r,q}) \circ (R_{q,q})^\star \circ (R_{q,s})$, where
  - $R_{r,q} = \widehat{\delta}(r, q)$,
  - $R_{q,q} = \widehat{\delta}(q, q)$, and
  - $R_{q,s} = \widehat{\delta}(q, s)$.

Eliminating States

- With a bit of work we can show that — when the state q is deleted — if we replace the label $R_{r,s}$ of the transition from r to s (for each pair of states r and s for which a transition should be defined) with the new label

$$R_{r,s} \cup ((R_{r,q}) \circ (R_{q,q})^\star \circ (R_{q,s}))$$

  then the language of the GNFA will not be changed.
- There will often be a shorter and *simpler* regular expression than the above one that has the same language. We can also (safely) use that regular expression instead.

Simplification Rules

1. In particular, if either $R_{r,q} = \emptyset$ or $R_{q,s} = \emptyset$ then the language of $(R_{r,q}) \circ (R_{q,q})^\star \circ (R_{q,s})$ is the empty language $\emptyset$. In this case the regular expression labelling the transition from r to s does not need to be changed.
2. If $R_{q,q} = \emptyset$ then the language of $(R_{r,q}) \circ (R_{q,q})^\star \circ (R_{q,s})$ is the same as the language of the simpler regular expression $(R_{r,q}) \circ (R_{q,s})$ — because $\emptyset^\star = \{\lambda\}$, and the concatenation of any string $\mu \in \Sigma^\star$ and the empty string is the same string μ.

Simplification Rules

3. Finally, if $R_{r,s} = \emptyset$ then the language of the regular expression

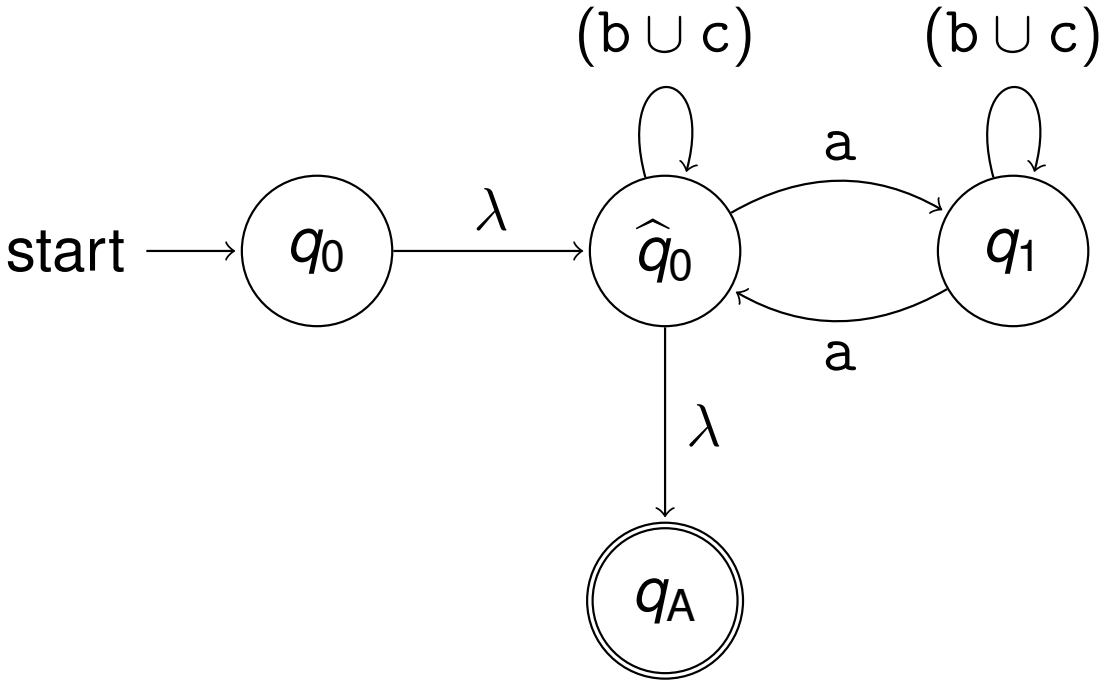
$$R_{r,s} \cup ((R_{r,q}) \circ (R_{q,q})^\star \circ (R_{q,s}))$$

   is the same as the language of the simpler regular expression $(R_{r,q}) \circ (R_{q,q})^\star \circ (R_{q,s})$.

4. These simplifications can be used in combination too!

Eliminating States

First Example, Continued: Consider, once again, the GNFA that we obtained for the language $L_1 \subseteq \Sigma_1^\star$, where $\Sigma_1 = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$:



We will continue by eliminating the state $q = q_1$.

Eliminating States

- Suppose $r = q_0$. Then, since $R_{r,q} = R_{q_0,q_1} = \emptyset$, it follows by the first "simplification rule" given above that $\widehat{\delta}(r, s) = \widehat{\delta}(q_0, s)$ does not need to be changed (for any state s such that this transition is defined).
- The label for the transition from $\widehat{q}_0$ to $\widehat{q}_0$ should be changed to

$$(\mathtt{b} \cup \mathtt{c}) \cup (\mathtt{a})(\mathtt{b} \cup \mathtt{c})^{\star}(\mathtt{a})$$

— and, apart from removing unnecessary brackets and changing this to

$$\mathtt{b} \cup \mathtt{c} \cup \mathtt{a}(\mathtt{b} \cup \mathtt{c})^{\star}\mathtt{a}$$

— there is no (obvious) way to simplify this regular expression.

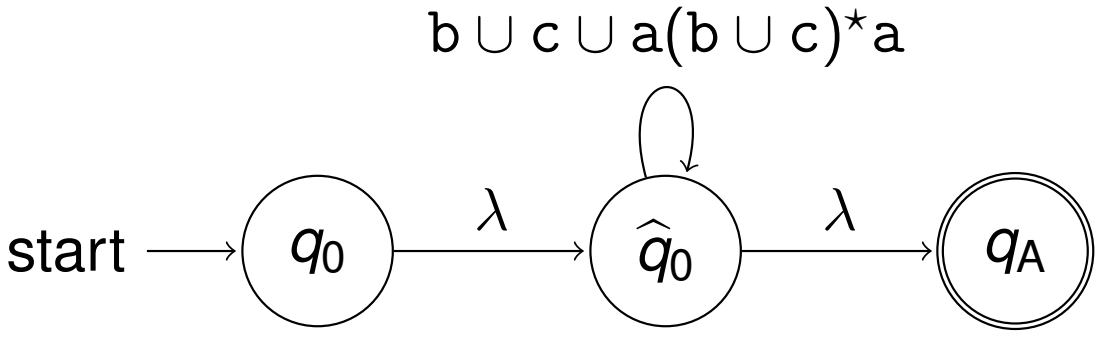
Eliminating States

- Finally, since $\widehat{\delta}(q_1, q_A) = \emptyset$ it follows by the first simplification rule that $\widehat{\delta}(\widehat{q}_0, q_A)$ does not need to be changed either.
- A table for the revised transition function is as follows.

	$\widehat{q}_0$	q_A
q_0	λ	$\emptyset$
$\widehat{q}_0$	$\mathtt{b} \cup \mathtt{c} \cup \mathtt{a}(\mathtt{b} \cup \mathtt{c})^\star\mathtt{a}$	λ

Eliminating States

A picture of the resulting GNFA is as follows.



Eliminating States

We must continue by eliminating the state $\widehat{q}_0$.

- The label for the transition from q_0 to q_A should now be changed to

$$(\lambda)(b \cup c \cup a(b \cup c)^\star a)^\star(\lambda).$$

Since the concatenation of the empty string, λ, and a string μ, is always equal to the same string μ, the language of the above regular expression is the same as the language of the *slightly* simpler regular expression

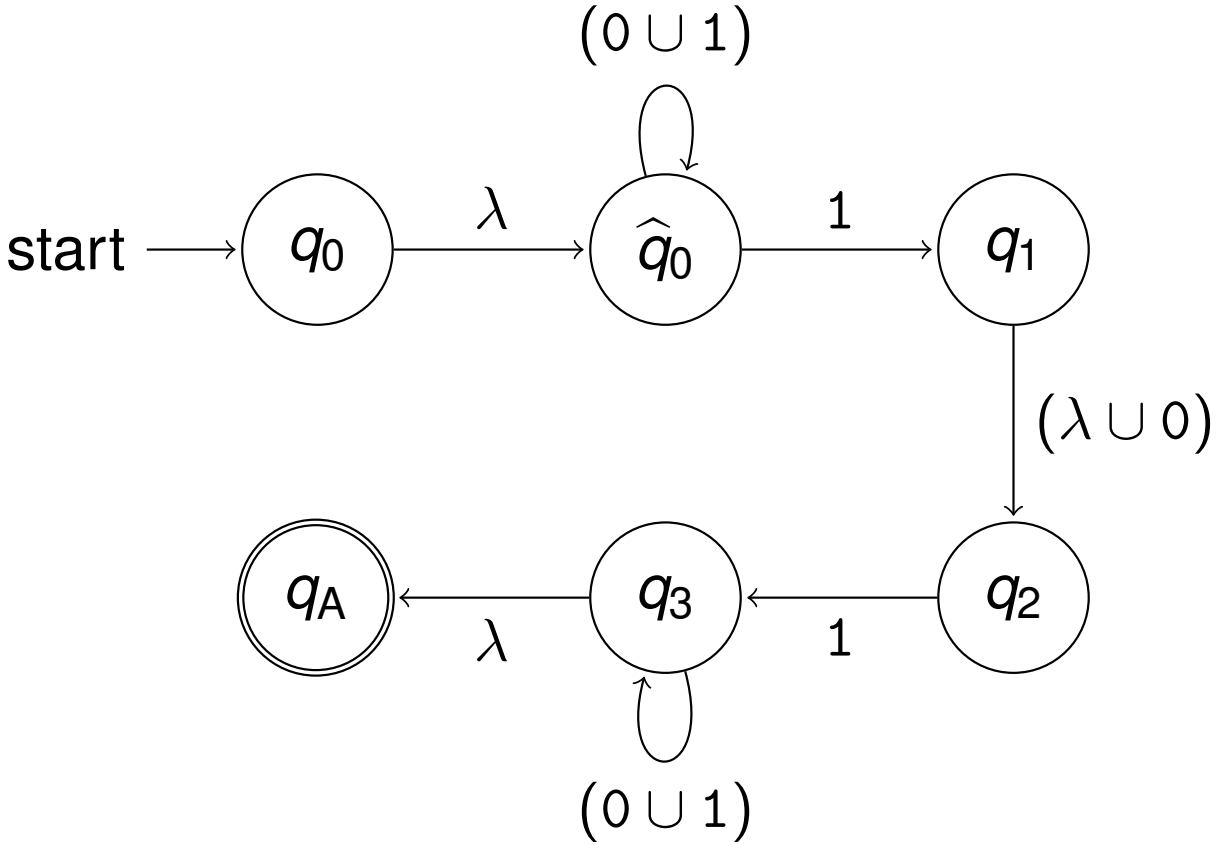
$$(b \cup c \cup a(b \cup c)^\star a)^\star$$

Eliminating States

A picture of the resulting GNFA is as follows.

Eliminating States

Second Example, Continued: Consider, once again, the GNFA that we obtained for the language $L_2 \subseteq \Sigma_2^\star$, where $\Sigma_2 = \{0, 1\}$:



Eliminating States

Let us begin by eliminating the state $q = q_3$.

- Since $\widehat{\delta}(r, q_3) = \emptyset$ for $r \in \{q_0, \widehat{q}_0, q_1\}$, it follows by the first simplification rule, given above, that the regular expressions $\widehat{\delta}(q_0, s)$, $\widehat{\delta}(\widehat{q}_0, s)$, and $\widehat{\delta}(q_1, s)$ do not need to be changed for any states s for which these transitions are defined.
- Since $\widehat{\delta}(q_3, s) = \emptyset$ as well, for $s \in \{\widehat{q}_0, q_1, q_2\}$, it also follows by the first simplification rule, given above, that the regular expressions $\widehat{\delta}(q_2, \widehat{q}_0)$, $\widehat{\delta}(q_2, q_1)$ and $\widehat{\delta}(q_2, q_2)$ do not need to be changed either.
- The only transition left to worry about is the transition from q_2 to q_A.

Eliminating States

- The transition from q_2 to q_A should be replaced with

$$\emptyset \cup (1)(0 \cup 1)^\star(\lambda)$$

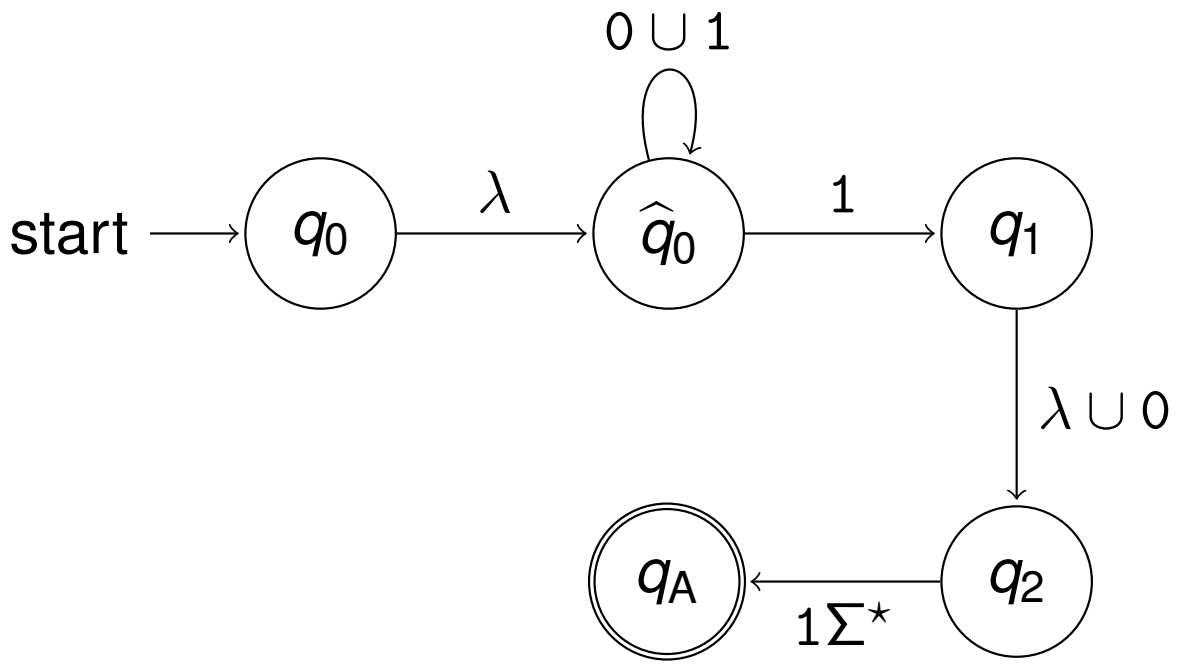
— which has the same language as the simpler regular expression $1\Sigma^\star$, since $\Sigma = \{0, 1\}$.

- We have now produced a GNFA whose transition table is as follows.

	$\widehat{q}_0$	q_1	q_2	q_A
q_0	λ	$\emptyset$	$\emptyset$	$\emptyset$
$\widehat{q}_0$	$(0 \cup 1)$	1	$\emptyset$	$\emptyset$
q_1	$\emptyset$	$\emptyset$	$(\lambda \cup 0)$	$\emptyset$
q_2	$\emptyset$	$\emptyset$	$\emptyset$	$1\Sigma^\star$

Eliminating States

A picture of this GNFA is as follows.



Eliminating States

Let us continue by eliminating the state $\widehat{q}_0$.

- The transition from q_0 to q_1 should be replaced by

$$\emptyset \cup (\lambda)(0 \cup 1)^\star(1)$$

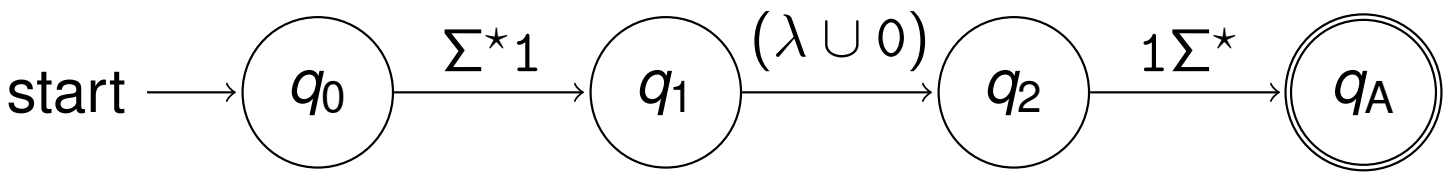
  — which has the same language as the simpler regular expression $\Sigma^\star 1$.
- Since the label for the transition from $\widehat{q}_0$ to s is $\emptyset$ for $s \in \{q_2, q_A\}$, neither of the labels for the transitions from q_0 to q_2 or q_A need to be changed.
- Since the label for the transition from r to $\widehat{q}_0$ is $\emptyset$ for $r \in \{q_1, q_2\}$, none of the labels for any other transitions need to be changed either.

Eliminating States

- We have now produced a GNFA whose transition table is as follows.

	q_1	q_2	q_A
q_0	$\Sigma^\star 1$	$\emptyset$	$\emptyset$
q_1	$\emptyset$	$(\lambda \cup 0)$	$\emptyset$
q_2	$\emptyset$	$\emptyset$	$1\Sigma^\star$

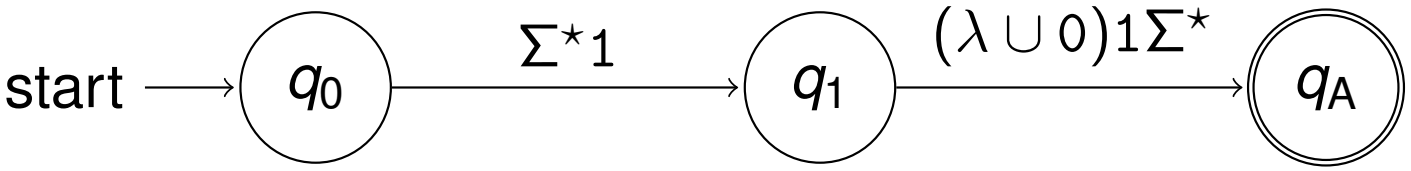
A picture of this GNFA is as follows.



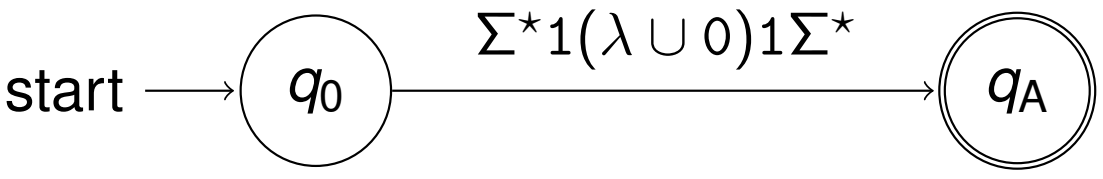
Eliminating States

Exercises:

1. Confirm that if q_2 is eliminated next then the result is a GNFA whose picture is as follows.



2. Confirm that if q_1 is eliminated after that then the result is a GNFA whose picture is as follows.



Finding the Resulting Regular Expression

The GNFA we are left with always two states — the start state and the accepting state — and a single transition between them. The label of this transition is a regular expression for the language of the GNFA — and for the language of the DFA or NFA we started with.

Thus a regular expression for the language L_1 is

$$(b \cup c \cup a(b \cup c)^{\star}a)^{\star}$$

and a regular expression for the language L_2 is

$$\Sigma^{\star}1(\lambda \cup 0)1\Sigma^{\star}$$

Conclusions — and Expectations for Students

The following has now been proved: A language is a regular language ***if and only if*** it is the language of a regular expression.

Students will be expected to be able to apply the process described today to reasonably ***simple*** and ***small*** DFAs and NFAs to produce regular expressions for their languages.

In the instructor's opinion the best way to learn how to do this is to ***practice*** ahead of time: Many people find that tasks like this take far less time if they have performed them a few times already.