Building a Regular Expression from an NFA

Given an NFA $M$, build an equivalent regular expression $\alpha$.

That is, the language accepted by the NFA $M$ is identical to the language defined by the regular expression $\alpha$: $L(M) = L(\alpha)$.

For this construction, we will use a new kind of finite automation called a generalized nondeterministic finite automaton (GNFA).
Generalized NFAs

A GNFA is an extension of an NFA where state transitions are defined by regular expressions. Edge labels are now regular expressions.

Motivation:

1. NFAs and regular expressions can be viewed as different representations of the same thing: regular languages.
2. A GNFA is a hybrid representation, an NFA with transitions based on regular expressions.
Generalized NFAs

A GNFA is an extension of an NFA where state transitions are defined by regular expressions. Edge labels are now regular expressions.

Motivation:

1. NFAs and regular expressions can be viewed as different representations of the same thing: regular languages.
2. A GNFA is a hybrid representation, an NFA with transitions based on regular expressions.
3. As such, useful for conversion from NFAs to regular expressions.
Generalized NFAs

A GNFA is an extension of an NFA where state transitions are defined by regular expressions. Edge labels are now regular expressions.

Motivation:

1. NFAs and regular expressions can be viewed as different representations of the same thing: regular languages.
2. A GNFA is a hybrid representation, an NFA with transitions based on regular expressions.
3. As such, useful for conversion from NFAs to regular expressions.
4. Two step conversion: from NFA to GNFA to regular expression.
Generalized NFAs

We define GNFAs so that they are always in a form that is convenient for use in constructing a regular expression from an NFA.

The form has the following conditions:

1. The start state has a transition to every other state but there are no transitions to it.
2. There is only a single accept state, and it has a transition from every other state but no transitions to another state. It is not the start state.
3. Have a single transition between every pair of states \((p, q)\), including from a state to itself (except no transitions into start state or out of final state).
Building a Regular Expression from an NFA

Construction of regular expression from NFA:

1. Convert NFA to GNFA.

2. Convert GNFA to regular expression by eliminating one state at a time until down to a GNFA with only a start state and an accepting state, labeled by the equivalent regular expression.
Building a Regular Expression from an NFA

Construction of regular expression from NFA:

1. Convert NFA to GNFA.
2. Convert GNFA to regular expression by eliminating one state at a time until down to a GNFA with only a start state and an accepting state, labeled by the equivalent regular expression.

▶ For each GNFA, every edge label is a regular expression that represents a set of paths between corresponding nodes in the NFA.
Let $R(p, q)$ be the regular expression that labels the transition in a GNFA from $p$ to $q$.

From an input NFA $M$, construct an output GNFA $M'$ with two states, a start state $q_1$ and an accepting state $q_2$, where $R(q_1, q_2)$ describes $L(M)$.

For $\alpha = R(q_1, q_2)$, $L(\alpha) = L(M)$. 
Transformations of NFA $M$ to satisfy the first two conditions of GNFA form:

1. If the start state of $M$ has any transitions coming into it, create a new start state $s$ and connect $s$ to $M$’s start state via an $\epsilon$ transition.

2. If there is more than one accepting state of $M$ or if there is just one but there are transitions out of it, create a new accepting state and connect each of $M$’s final states to it via an $\epsilon$ transition.

Remove the original accepting states from the set of accepting states.
Building a Regular Expression from an NFA

To satisfy the third condition of standard GNFA form:

- If the set of labels on the set of transitions from $p$ to $q$ is $\{c_1, c_2, ..., c_n\}$, then delete them and replace them with a single transition with the label $\{c_1 + c_2 + ... + c_n\}$.
- For each state pair that does not yet have a transition, create a transition from $p$ to $q$ labeled $\emptyset$. 
Conversion to GNFA

Figure 6.3
Building a Regular Expression from an NFA

Having performed the above transformations, have converted NFA $M$ to GNFA $M'$. Now perform state elimination on $M'$ to build the regular expression $\alpha$ corresponding to $M$.

State elimination: eliminate states one by one, until only the start and final states with transition $\alpha$ remain.

Select a state $r$ and remove it and the transitions into it and out of it, while retaining information about paths through that state. Must modify remaining transitions so the functionality of $M'$ is the same.
To remove state $r$ from GNFA $M'$:

Consider any pair of states $p$ and $q$ distinct from $r$.

To update $M'$ after removing state $r$:

- Update $R(p, q)$ to include transitions from $p$ to $q$ thru $r$:
  - Transition from $p$ to $r$;
  - $r$ to itself zero or more times;
  - transition from $r$ to $q$.

$$R'(p, q) = R(p, q) + R(p, r)R(r, r)^*R(r, q).$$
Function $\text{buildregex}(M')$: For GNFA $M'$, returns the regular expression that accepts the same language as $M'$.

Until only the start state and the final state remain do:

1. Select some state $r$ of $M'$ other than the start or end state.
2. For every transition from a state $p$ to a state $q$, where $p$ and $q$ are distinct from $r$, compute the new label $R'(p, q)$ using the formula:
   \[ R'(p, q) = R(p, q) + R(p, r)R(r, r)^*R(r, q). \]
3. Remove $r$ and all transitions into and out of it.

Return the regular expression $\alpha$ that labels the transition from the start state to the final state.
Example 6.9-1

Let \( \text{rip} \) be state 2. Then

\[
R'(1, 3) = R(1, 3) \cup R(1, \text{rip})R(\text{rip}, \text{rip})^*R(\text{rip}, 3)
\]

\[
= R(1, 3) \cup R(1, 2)R(2, 2)^*R(2, 3)
\]

\[
= \emptyset \cup a \ b^* \ a
\]

\[
= a \ b^* \ a
\]
Remove state 2.

\[
R'(1, 4) = R(1, 4) \cup R(1, 2)R(2, 2)^*R(2, 4)
\]
\[
= b \cup a \ b^* \ \emptyset
\]
\[
= b
\]

\[
R'(4, 3) = R(4, 3) \cup R(4, 2)R(2, 2)^*R(2, 3)
\]
\[
= \emptyset \cup b \ b^* \ a
\]
\[
= b \ b^* \ a
\]

There are no outgoing edges from 3 and no incoming edges to 1.
Example 6.9-3

Remove state 4.

\[ R'(1,3) = R(1,3) \cup R(1,4)R(4,4)^*R(4,3) \]
\[ = a b^* a \cup bbb^* a \]

Result
Example 6.9-4

Original FSM:

```
 start 1 (a) 2 (a) 3
        b     b
```

Regular Expression:

```
ab^a \cup bbb^a
```