Equivalence of DFA and Regular Expressions
CSCI 3130 Formal Languages and Automata Theory

Siu On CHAN

Chinese University of Hong Kong

Fall 2017
Three ways of doing it

\[ L = \{ x \in \Sigma^* \mid x \text{ ends in } 01 \} \quad \Sigma = \{0, 1\} \]

**Deterministic Finite Automaton (DFA)**

- Start state: \( q_0 \)
- Accepting state: \( q_2 \)
- Transitions:
  - \( q_0 \to q_1 \) on input 0
  - \( q_0 \to q_0 \) on input 1
  - \( q_1 \to q_2 \) on input 1
  - \( q_1 \to q_1 \) on input 0

**Non-deterministic Finite Automaton (NFA)**

- Start state: \( q_0 \)
- Accepting state: \( q_2 \)
- Transitions:
  - \( q_0 \to q_0 \) on input 0/1
  - \( q_0 \to q_1 \) on input 0
  - \( q_1 \to q_2 \) on input 1

**Regular Expressions**

\[(0 + 1)^*01\]
They are equally powerful

DFA  NFA  regular expressions

regular languages
Roadmap

regular expressions

NFA

DFA

✓
Examples: regular expression $\rightarrow$ NFA

$R_1 = 0$ $\rightarrow$ \[
\begin{array}{c}
q_0 \\
0
\end{array}
\begin{array}{c}
\rightarrow \\
\rightarrow \\
q_1
\end{array}
\]

$R_2 = 01$ $\rightarrow$ \[
\begin{array}{c}
q_0 \\
0
\end{array}
\begin{array}{c}
\rightarrow \\
\rightarrow \\
q_1 \\
1
\end{array}
\begin{array}{c}
\rightarrow \\
\rightarrow \\
q_2
\end{array}
\]
Examples: regular expression → NFA

\[ R_3 = 0 + 01 \]

\[ R_4 = (0 + 01)^* \]
Regular expressions

In general, how do we convert a regular expression to an NFA?

A regular expression over $\Sigma$ is an expression formed by the following rules

- The symbols $\emptyset$ and $\varepsilon$ are regular expressions.
- Every symbol in $\Sigma$ is a regular expression.
  - If $\Sigma = \{0, 1\}$, then 0 and 1 are both regular expressions.
- If $R$ and $S$ are regular expressions, so are $R + S$, $RS$ and $R^*$.
General method when $\Sigma = \{0, 1\}$
General method

Regular expression $\Rightarrow$ NFA

$RS$

$R + S$

$R^*$
Roadmap

regular expressions

NFA
Roadmap

- regular expressions
- 2-state GNFA
- GNFA
- NFA
Simplify the NFA

First we simplify the NFA so that

- It has exactly one accepting state
- No arrows come into the start state
- No arrows go out of the accepting state
First we simplify the NFA so that

- It has exactly one accepting state
- No arrows come into the start state
- No arrows go out of the accepting state
Simplify the NFA

- It has exactly one accepting state.
- No arrows come into the start state.
- No arrows go out of the accepting state.
Simplify the NFA

- It has exactly one accepting state ✓
- No arrows come into the start state ✓
- No arrows go out of the accepting state ✓
Generalized NFAs

A generalized NFA is an NFA whose transitions are labeled by regular expressions, like

\[ q_0 \xrightarrow{\varepsilon + 10^*} q_1 \xrightarrow{0^*1} q_2 \]
We will **eliminate** every state but the start and accepting states.
State elimination: general method

To eliminate state $q$, for every pair of states $(u, v)$

Replace

\[ u \xrightarrow{R_1} q \xrightarrow{R_2} v \]

by

\[ u \xrightarrow{R_1 R_2^* R_3 + R_4} v \]

Remember to do this even when $u = v$
A 2-state GNFA is the same as a regular expression $R$. 

The image illustrates the relationship between regular expressions, 2-state GNFA, GNFA, and NFA. The diagram shows that a 2-state GNFA is equivalent to a regular expression $R$. This equivalence is indicated by the arrows and check marks in the diagram.
Conversion example

Eliminate $q_1$:
Conversion example

Eliminate $q_1$:

Eliminate $q_2$: $0^*1(00^*1 + 1)^*$
Conversion example

\[ q_0 \xrightarrow{\varepsilon} q_1 \xrightarrow{1} q_2 \xrightarrow{\varepsilon} q_3 \]

Eliminate \( q_1 \):

\[ q_0 \xrightarrow{0 \ast 1} q_2 \xrightarrow{\varepsilon} q_3 \]

Eliminate \( q_2 \):

\[ q_0 \xrightarrow{0 \ast 1 \left(00 \ast 1 + 1\right)^\ast} q_3 \]

Check:

\[ 0 \ast 1 \left(00 \ast 1 + 1\right)^\ast \overset{?}{=} 0 \xrightarrow{1} q_1 \xrightarrow{0} q_2 \]
Check your answer!

All strings ending in 1

\[(0 + 1)^*1\]

Yes
Check your answer!

All strings ending in 1

$(0 + 1)^* 1$

$0^* 1 (00^* 1 + 1)^*$

Always ends in 1

Does every string ending in 1 have this form?

Yes