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

Siu On CHAN

Chinese University of Hong Kong

Fall 2016
Three ways of doing it

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

DFA

\[
\begin{align*}
q_0 & \quad 0 \\
q_1 & \quad 1 \quad 0 \\
q_2 & \quad 0 \quad 1 \quad 0
\end{align*}
\]

NFA

\[
\begin{align*}
q_0 & \quad 0,1 \\
q_1 & \quad 0 \\
q_2 & \quad 1 \\
\end{align*}
\]

Regular expressions

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

regular expressions → NFA → NFA without $\varepsilon$ → DFA

DFA → regular expressions
Examples: regular expression → NFA

\[ R_1 = 0 \]

\[ q_0 \xrightarrow{0} q_1 \]

\[ R_2 = 01 \]

\[ q_0 \xrightarrow{0} q_1 \xrightarrow{1} q_2 \]
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 \( a \) in \( \Sigma \) is a regular expression.
- If \( R \) and \( S \) are regular expressions, so are \( R + S \), \( RS \) and \( R^* \).
General method

regular expression $\implies$ NFA

$\emptyset \quad \implies q_0$

$\varepsilon \quad \implies q_0$

$a \in \Sigma \quad \implies q_0 \xrightarrow{a} q_1$
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
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
Simplify the NFA

- It has exactly one accepting state: $q_3$.
- No arrows come into the start state: $q_0$.
- No arrows go out of the accepting state: $q_3$.

Diagram:

- States: $q_0$, $q_1$, $q_2$, $q_3$.
- Transitions:
  - $q_0$ to $q_1$: 0
  - $q_1$ to $q_2$: 1
  - $q_2$ to $q_0$: 0
  - $q_2$ to $q_3$: 1
Simplify the NFA

- It has exactly one accepting state ✓
- No arrows come into the start state ✓
- No arrows go out of the accepting state ✓
A generalized NFA is an NFA whose transitions are labeled by regular expressions, like

![Diagram of a generalized NFA]

- $q_0$ to $q_1$: $\varepsilon + 10^*$
- $q_1$ to $q_2$: $0^*11$
- $q_1$ to $q_0$: $01$
We will **eliminate** every state but the start and accepting states.
State elimination

\[ q_0 \xrightarrow{\varepsilon + 10^*} q_1 \xrightarrow{0^*1} q_2 \]

\[ q_0 \xrightarrow{01} q_1 \xrightarrow{0^*11} q_2 \]

\[ (\varepsilon + 10^*) (0^*1)^* 0^*11 \]

\[ (\varepsilon + 10^*) (0^*1)^* 0^*11 + 01 \]
State elimination: general method

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

Replace

\[ R_1, R_2, R_3, R_4 \]

by

\[ R_1 R_2^* R_3 + R_4 \]

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

regular expressions

2-state GNFA

GNFA

NFA

$q_0 \xrightarrow{R} q_1$
Conversion example

Eliminate $q_1$:

![Diagram](attachment:image.png)
Conversion example

Eliminate $q_1$: $0*1 + 1$

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

Eliminate $q_1$:

Eliminate $q_2$:

Check:

$0^*1(00^*1 + 1)^* = 0^*1(00^*1 + 1)^*$
All strings ending in 1

$(0 + 1)^*1$
Check your answer!

All strings ending in 1 

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

\[0^* 1(00^* 1 + 1)^*\] 

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

Always ends in 1

Does every string ending in 1 have this form?

Yes