Regular Expressions

Carnegie Mellon University in Qatar
SUMMARY

- Nondeterminism
  - Clone the FA at choice points
Nondeterminism

- Clone the FA at choice points
- Guess and verify
Summary

- Nondeterminism
  - Clone the FA at choice points
  - Guess and verify
- Nondeterministic FA

Multiple transitions from a state with the same input symbol

\( \epsilon \)-transitions

NFAs are equivalent to DFAs

Determinization procedure builds a DFA with up to \( 2^k \) states for an NFA with \( k \) states.
**Summary**

- **Nondeterminism**
  - Clone the FA at choice points
  - Guess and verify

- **Nondeterministic FA**
  - Multiple transitions from a state with the same input symbol

=$(C_{A R N E G I E ~ M E L L O N ~ U N I V E R S I T Y ~ I N ~ Q A T A R})$
**Summary**

- **Nondeterminism**
  - Clone the FA at choice points
  - Guess and verify

- **Nondeterministic FA**
  - Multiple transitions from a state with the same input symbol
  - $\epsilon$-transitions
Summary

- Nondeterminism
  - Clone the FA at choice points
  - Guess and verify

- Nondeterministic FA
  - Multiple transitions from a state with the same input symbol
  - $\epsilon$-transitions

- NFAs are equivalent to DFAs
**Summary**

- **Nondeterminism**
  - Clone the FA at choice points
  - Guess and verify

- **Nondeterministic FA**
  - Multiple transitions from a state with the same input symbol
  - $\epsilon$-transitions

- **NFAs are equivalent to DFAs**
  - Determinization procedure builds a DFA with up to $2^k$ states for an NFA with $k$ states.
The class of regular languages is closed under the union operation.
The class of regular languages is closed under the union operation.
The class of regular languages is closed under the concatenation operation.
**Theorem**

The class of regular languages is closed under the concatenation operation.

**Proof Idea based on NFAs**
The class of regular languages is closed under the star operation.
**Theorem**
The class of regular languages is closed under the star operation.
DFAs are finite descriptions of (finite or infinite) sets of strings
DFAs are **finite** descriptions of (finite or infinite) sets of strings
- Finite number of symbols, states, transitions
**Regular Expressions**

- DFAs are **finite** descriptions of (finite or infinite) sets of strings
  - Finite number of symbols, states, transitions
- **Regular Expressions** provide an algebraic expression framework to describe the same class of strings
REGULAR EXPRESSIONS

- DFAs are **finite** descriptions of (finite or infinite) sets of strings
  - Finite number of symbols, states, transitions
- **Regular Expressions** provide an algebraic expression framework to describe the same class of strings
- Thus, DFAs and Regular Expressions are equivalent.
For every regular expression, there is a corresponding regular set or language
For every regular expression, there is a corresponding regular set or language

$R, R_1, R_2$ are regular expressions; $L(R)$ denotes the corresponding regular set
For every regular expression, there is a corresponding regular set or language.

- $R, R_1, R_2$ are regular expressions; $L(R)$ denotes the corresponding regular set.

<table>
<thead>
<tr>
<th>Regular Expression</th>
<th>Regular Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\phi$</td>
<td>${} \cup {\epsilon}$</td>
</tr>
<tr>
<td>$a$ for $a \in \Sigma$</td>
<td>${a}$</td>
</tr>
<tr>
<td>$R_1 \cup R_2$</td>
<td>$L(R_1) \cup L(R_2)$</td>
</tr>
<tr>
<td>$R_1 R_2$</td>
<td>$L(R_1) L(R_2)$</td>
</tr>
<tr>
<td>$R^*$</td>
<td>$L(R)$</td>
</tr>
</tbody>
</table>

(Carnegie Mellon University in Qatar)
For every regular expression, there is a corresponding regular set or language 

$R, R_1, R_2$ are regular expressions; $L(R)$ denotes the corresponding regular set

<table>
<thead>
<tr>
<th>Regular Expression</th>
<th>Regular Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\phi$</td>
<td>${}$</td>
</tr>
</tbody>
</table>
For every regular expression, there is a corresponding regular set or language.

- $R, R_1, R_2$ are regular expressions; $L(R)$ denotes the corresponding regular set.

<table>
<thead>
<tr>
<th>Regular Expression</th>
<th>Regular Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\phi$</td>
<td>${} \cup {a}$</td>
</tr>
<tr>
<td>$a$ for $a \in \Sigma$</td>
<td>${a}$</td>
</tr>
</tbody>
</table>
For every regular expression, there is a corresponding regular set or language.

\( R, R_1, R_2 \) are regular expressions; \( L(R) \) denotes the corresponding regular set.

<table>
<thead>
<tr>
<th>Regular Expression</th>
<th>Regular Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \phi )</td>
<td>{ }</td>
</tr>
<tr>
<td>( a ) for ( a \in \Sigma )</td>
<td>{ ( a ) }</td>
</tr>
<tr>
<td>( \epsilon )</td>
<td>{ ( \epsilon ) }</td>
</tr>
</tbody>
</table>
For every regular expression, there is a corresponding regular set or language.

- \( R, R_1, R_2 \) are regular expressions; \( L(R) \) denotes the corresponding regular set.

<table>
<thead>
<tr>
<th>Regular Expression</th>
<th>Regular Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \phi )</td>
<td>{ }</td>
</tr>
<tr>
<td>( a ) for ( a \in \Sigma )</td>
<td>{a}</td>
</tr>
<tr>
<td>( \epsilon )</td>
<td>{\epsilon}</td>
</tr>
<tr>
<td>( (R_1 \cup R_2) )</td>
<td>( L(R_1) \cup L(R_2) )</td>
</tr>
</tbody>
</table>
For every regular expression, there is a corresponding regular set or language.

- $R, R_1, R_2$ are regular expressions; $L(R)$ denotes the corresponding regular set.

<table>
<thead>
<tr>
<th>Regular Expression</th>
<th>Regular Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\phi$</td>
<td>${}$</td>
</tr>
<tr>
<td>$a$ for $a \in \Sigma$</td>
<td>${a}$</td>
</tr>
<tr>
<td>$\epsilon$</td>
<td>${\epsilon}$</td>
</tr>
<tr>
<td>$(R_1 \cup R_2)$</td>
<td>$L(R_1) \cup L(R_2)$</td>
</tr>
<tr>
<td>$(R_1 R_2)$</td>
<td>$L(R_1)L(R_2)$</td>
</tr>
</tbody>
</table>
For every regular expression, there is a corresponding regular set or language.

$R$, $R_1$, $R_2$ are regular expressions; $L(R)$ denotes the corresponding regular set.

<table>
<thead>
<tr>
<th>Regular Expression</th>
<th>Regular Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\phi$</td>
<td>${} {}$</td>
</tr>
<tr>
<td>$a$ for $a \in \Sigma$</td>
<td>${a}$</td>
</tr>
<tr>
<td>$\epsilon$</td>
<td>${\epsilon}$</td>
</tr>
<tr>
<td>$(R_1 \cup R_2)$</td>
<td>$L(R_1) \cup L(R_2)$</td>
</tr>
<tr>
<td>$(R_1 R_2)$</td>
<td>$L(R_1)L(R_2)$</td>
</tr>
<tr>
<td>$(R^*)$</td>
<td>$L(R)^*$</td>
</tr>
</tbody>
</table>
### Regular Expressions – More Syntax

<table>
<thead>
<tr>
<th>Regular Expression</th>
<th>Regular Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>φ</td>
<td>{}</td>
</tr>
<tr>
<td>a for ∈ Σ</td>
<td>{a}</td>
</tr>
<tr>
<td>ε</td>
<td>{ε}</td>
</tr>
<tr>
<td>(R₁ ∪ R₂)</td>
<td>L(R₁) ∪ L(R₂)</td>
</tr>
<tr>
<td>(R₁ ◦ R₂)</td>
<td>L(R₁) ◦ L(R₂)</td>
</tr>
<tr>
<td>(R*)</td>
<td>L(R)*</td>
</tr>
</tbody>
</table>

Also some books use \( R₁ + R₂ \) to denote union.
### Regular Expressions – More Syntax

<table>
<thead>
<tr>
<th>Regular Expression</th>
<th>Regular Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\phi$</td>
<td>${} \cup {\epsilon}$</td>
</tr>
<tr>
<td>a for $a \in \Sigma$</td>
<td>${a}$</td>
</tr>
<tr>
<td>$\epsilon$</td>
<td>${\epsilon}$</td>
</tr>
<tr>
<td>($R_1 \cup R_2$)</td>
<td>$L(R_1) \cup L(R_2)$</td>
</tr>
<tr>
<td>($R_1 \circ R_2$)</td>
<td>$L(R_1) \circ L(R_2)$</td>
</tr>
<tr>
<td>($R^*$)</td>
<td>$L(R)^*$</td>
</tr>
</tbody>
</table>

- Also some books use $R_1 + R_2$ to denote union.
- In $(\ldots)$, the parenthesis can be deleted.
### Regular Expressions – More Syntax

<table>
<thead>
<tr>
<th>Regular Expression</th>
<th>Regular Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\phi$</td>
<td>${}$</td>
</tr>
<tr>
<td>$a$ for $a \in \Sigma$</td>
<td>${a}$</td>
</tr>
<tr>
<td>$\epsilon$</td>
<td>${\epsilon}$</td>
</tr>
<tr>
<td>$(R_1 \cup R_2)$</td>
<td>$L(R_1) \cup L(R_2)$</td>
</tr>
<tr>
<td>$(R_1 \circ R_2)$</td>
<td>$L(R_1) \circ L(R_2)$</td>
</tr>
<tr>
<td>$(R^*)$</td>
<td>$L(R)^*$</td>
</tr>
</tbody>
</table>

Also some books use $R_1 + R_2$ to denote union.

In (…), the parenthesis can be deleted

- In which case, interpretation is done in the precedence order: star, concatenation and then union.
### Regular Expressions—More Syntax

<table>
<thead>
<tr>
<th>Regular Expression</th>
<th>Regular Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \emptyset )</td>
<td>( { } )</td>
</tr>
<tr>
<td>( a ) for ( a \in \Sigma )</td>
<td>( { a } )</td>
</tr>
<tr>
<td>( \epsilon )</td>
<td>( { \epsilon } )</td>
</tr>
<tr>
<td>( (R_1 \cup R_2) )</td>
<td>( L(R_1) \cup L(R_2) )</td>
</tr>
<tr>
<td>( (R_1 \circ R_2) )</td>
<td>( L(R_1) \circ L(R_2) )</td>
</tr>
<tr>
<td>( (R^*) )</td>
<td>( L(R)^* )</td>
</tr>
</tbody>
</table>

- Also some books use \( R_1 + R_2 \) to denote union.

- In \((\ldots)\), the parenthesis can be deleted
  - In which case, interpretation is done in the precedence order: star, concatenation and then union.

- \( R^+ = RR^* \) and \( R^k \) for \( k \)-fold concatenation are useful shorthands.
Regular Expression Examples

<table>
<thead>
<tr>
<th>Regular Expression</th>
<th>Regular Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0^<em>10^</em>$</td>
<td>${\omega</td>
</tr>
<tr>
<td>$0^*10^<em>1^</em>$</td>
<td>${\omega</td>
</tr>
<tr>
<td>$0^*10^*0^<em>10^</em>$</td>
<td>${\omega</td>
</tr>
<tr>
<td>$(0^<em>10^<em>10^</em>)^</em>$</td>
<td>${\omega</td>
</tr>
</tbody>
</table>
**Regular Expression Examples**

<table>
<thead>
<tr>
<th>Regular Expression</th>
<th>Regular Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0^<em>10^</em>$</td>
<td>${\omega</td>
</tr>
</tbody>
</table>
**Regular Expression Examples**

Regular Expression

\[ 0^*10^* \]
\[ (0 \cup 1)^*1(0 \cup 1)^* \]

→ \[ \{ \omega | \omega \text{ contains a single 1} \} \]
Regular Expression Examples

<table>
<thead>
<tr>
<th>Regular Expression</th>
<th>Regular Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0^<em>10^</em>$</td>
<td>${\omega</td>
</tr>
<tr>
<td>$(0 \cup 1)^<em>1(0 \cup 1)^</em>$</td>
<td>${\omega</td>
</tr>
</tbody>
</table>
Regular Expression Examples

<table>
<thead>
<tr>
<th>Regular Expression</th>
<th>Regular Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0^<em>10^</em>$</td>
<td>${ \omega</td>
</tr>
<tr>
<td>$(0 \cup 1)^<em>1(0 \cup 1)^</em>$</td>
<td>${ \omega</td>
</tr>
</tbody>
</table>
| $0(0 \cup 1)^*0 \cup 1(0 \cup 1)^*1 \cup 0 \cup 1$ | }
Regular Expression Examples

<table>
<thead>
<tr>
<th>Regular Expression</th>
<th>Regular Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0^<em>10^</em>$</td>
<td>${\omega</td>
</tr>
<tr>
<td>$(0 \cup 1)^<em>1(0 \cup 1)^</em>$</td>
<td>${\omega</td>
</tr>
<tr>
<td>$0(0 \cup 1)^*0 \cup 1(0 \cup 1)^*1 \cup 0 \cup 1$</td>
<td>${\omega</td>
</tr>
</tbody>
</table>
Regular Expression Examples

Regular Expression

\(0^*10^*\)

\((0 \cup 1)^*1(0 \cup 1)^*\)

\(0(0 \cup 1)^*0 \cup 1(0 \cup 1)^*1 \cup 0 \cup 1\)

\((0^*10^*1)^*0^*\)

Regular Language

\(\{\omega|\omega \text{ contains a single } 1\}\)

\(\{\omega|\omega \text{ has at least one } 1\}\)

\(\{\omega|\omega \text{ starts and ends with the same symbol}\}\)
Regular Expression Examples

<table>
<thead>
<tr>
<th>Regular Expression</th>
<th>Regular Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0^<em>10^</em>$</td>
<td>${ \omega \mid \omega \text{ contains a single } 1 }$</td>
</tr>
<tr>
<td>$(0 \cup 1)^<em>1(0 \cup 1)^</em>$</td>
<td>${ \omega \mid \omega \text{ has at least one } 1 }$</td>
</tr>
<tr>
<td>$0(0 \cup 1)^*0 \cup 1(0 \cup 1)^*1 \cup 0 \cup 1$</td>
<td>${ \omega \mid \omega \text{ starts and ends with the same symbol} }$</td>
</tr>
<tr>
<td>$(0^*10^*1)^<em>0^</em>$</td>
<td>${ \omega \mid n_1(\omega) \text{ is even} }$</td>
</tr>
</tbody>
</table>
All strings with at least one pair of consecutive 0s
Writing Regular Expressions

- All strings with **at least** one pair of consecutive 0s
  - \((0 \cup 1)^*00(0 \cup 1)^*\)
All strings with at least one pair of consecutive 0s
- \((0 \cup 1)^*00(0 \cup 1)^*\)

All strings such that fourth symbol from the end is a 1
- \((0 \cup 1)^*1(0 \cup 1)(0 \cup 1)(0 \cup 1)^\epsilon\)
All strings with at least one pair of consecutive 0s
  \((0 \cup 1)^*00(0 \cup 1)^*\)

All strings such that fourth symbol from the end is a 1
  \((0 \cup 1)^*1(0 \cup 1)(0 \cup 1)(0 \cup 1)\)
- All strings with at least one pair of consecutive 0s
  \[(0 \cup 1)^*00(0 \cup 1)^*\]
- All strings such that fourth symbol from the end is a 1
  \[(0 \cup 1)^*1(0 \cup 1)(0 \cup 1)(0 \cup 1)\]
- All strings with no pair of consecutive 0s
All strings with **at least** one pair of consecutive 0s

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

All strings such that fourth symbol from the end is a 1

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

All strings with **no** pair of consecutive 0s

\[(1^*011^*)^*(0 \cup \epsilon) \cup 1^*\]
Writing Regular Expressions

- All strings with at least one pair of consecutive 0s
  \((0 \cup 1)^*00(0 \cup 1)^*\)

- All strings such that fourth symbol from the end is a 1
  \((0 \cup 1)^*1(0 \cup 1)(0 \cup 1)(0 \cup 1)\)

- All strings with no pair of consecutive 0s
  \((1^*011^*)^*(0 \cup \epsilon) \cup 1^*\)
  - Strings consist of repetitions of 1 or 01 or two boundary cases: \((1 \cup 01)^*(0 \cup \epsilon)\)
W R I T I N G  R E G U L A R  E X P R E S S I O N S

- All strings with **at least** one pair of consecutive 0s
  - \((0 \cup 1)^*00(0 \cup 1)^*\)

- All strings such that fourth symbol from the end is a 1
  - \((0 \cup 1)^*1(0 \cup 1)(0 \cup 1)(0 \cup 1)\)

- All strings with **no** pair of consecutive 0s
  - \((1^*011^*)^*(0 \cup \epsilon) \cup 1^*\)
  - Strings consist of repetitions of 1 or 01 or two boundary cases: \((1 \cup 01)^*(0 \cup \epsilon)\)

- All strings that **do not end** in 01.
All strings with at least one pair of consecutive 0s

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

All strings such that fourth symbol from the end is a 1

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

All strings with no pair of consecutive 0s

\[(1*011^*)^*(0 \cup \epsilon) \cup 1^*\]

Strings consist of repetitions of 1 or 01 or two boundary cases: \((1 \cup 01)^*(0 \cup \epsilon)\)

All strings that do not end in 01.

\[(0 \cup 1)^*(00 \cup 10 \cup 11) \cup 0 \cup 1 \cup \epsilon\]
All strings over \( \Sigma = \{a, b, c\} \) that contain every symbol at least once.
All strings over $\Sigma = \{a, b, c\}$ that contain every symbol at least once.

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

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

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

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

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

$$(a \cup b \cup c)^*c(a \cup b \cup c)^*b(a \cup b \cup c)^*a(a \cup b \cup c)^*\cup$$
Writing Regular Expressions

- All strings over $\Sigma = \{a, b, c\}$ that contain every symbol at least once.
All strings over $\Sigma = \{a, b, c\}$ that contain every symbol at least once.

DFAs and REs may need different ways of looking at the problem.

- For the DFA, you count symbols
- For the RE, you enumerate all possible patterns
RE IDENTITIES

- $\mathbb{R} \cup \emptyset = \mathbb{R}$
RE IDENTITIES

- \( R \cup \emptyset = R \)
- \( R\epsilon = \epsilon R = R \)
RE Identities

- $R \cup \emptyset = R$
- $R \epsilon = \epsilon R = R$
- $\emptyset^* = \epsilon$
RE IDENTITIES

- \( R \cup \emptyset = R \)
- \( R\epsilon = \epsilon R = R \)
- \( \emptyset^* = \epsilon \)

Note that we do not have explicit operators for intersection or complementation!
Digression: REs in Real Life

- Linux/Unix Shell, Perl, Awk, Python all have built in regular expression support for pattern matching functionality

See [http://www.wdvl.com/Authoring/Languages/Perl/PerlfortheWeb/perlintro2_table1.html](http://www.wdvl.com/Authoring/Languages/Perl/PerlfortheWeb/perlintro2_table1.html)
Digression: REs in Real Life

- Linux/Unix Shell, Perl, Awk, Python all have built in regular expression support for pattern matching functionality.

See [http://www.wdvl.com/Authoring/Languages/Perl/PerlfortheWeb/perlintro2_table1.html](http://www.wdvl.com/Authoring/Languages/Perl/PerlfortheWeb/perlintro2_table1.html)

- Mostly some syntactic extensions/changes to basic regular expressions with some additional functionality for remembering matches.

Substring matches in a string!

Search for and download Regex Coach to learn and experiment with regular expression matching.

(Carnegie Mellon University in Qatar) Slides for 15-453 Lecture 4 Spring 2011 14 / 26
Digression: REs in Real Life

- Linux/Unix Shell, Perl, Awk, Python all have built in regular expression support for pattern matching functionality.

See [http://www.wdvl.com/Authoring/Languages/Perl/PerlfortheWeb/perlintro2_table1.html](http://www.wdvl.com/Authoring/Languages/Perl/PerlfortheWeb/perlintro2_table1.html)

- Mostly some syntactic extensions/changes to basic regular expressions with some additional functionality for remembering matches.

- Substring matches in a string!

- Search for and download *Regex Coach* to learn and experiment with regular expression matching.
Equivalence with Finite Automata

**Theorem**

A language is regular if and only if some regular expression describes it.
Theorem

A language is regular if and only if some regular expression describes it.

Lemma - The if part

If a language is described by a regular expression, then it is regular.
Equivalence with Finite Automata

**Theorem**

A language is regular if and only if some regular expression describes it.

**Lemma - The if part**

If a language is described by a regular expression, then it is regular.

**Proof Idea**

Inductively convert a given regular expression to an NFA.
CONVERTING REs TO NFAs: BASIS CASES

Regular Expression  Corresponding NFA

\( \phi \)
CONVERTING REs TO NFAs: BASIS CASES

Regular Expression | Corresponding NFA
φ | q₀ → qᶠ
ε | q₀ → qᶠ

Regular Expression Corresponding NFA
φ
ε
CONVERTING REs TO NFAs: BASIS CASES

Regular Expression       Corresponding NFA

\emptyset

\epsilon

a for a \in \Sigma

\begin{figure}
\centering
\begin{tikzpicture}
  \node[state, initial, accepting] (q0) at (0,0) {q0};
  \node[state, accepting] (qf) at (1,0) {qf};
  \draw[->] (q0) -- node[above] {\epsilon} (qf);
  \draw[->] (q0) -- node[above] {a} (qf);
\end{tikzpicture}
\end{figure}
**Union**

Let $N_1$ and $N_2$ be NFAs for $R_1$ and $R_2$ respectively. Then the NFA for $R_1 \cup R_2$ is
CONVERTING REs TO NFAs

Concatenation

Let $N_1$ and $N_2$ be NFAs for $R_1$ and $R_2$ respectively. Then the NFA for $R_1 R_2$ is
Converting REs to NFAs: Star

Star

- Let $N$ be NFAs for $R$. Then the NFA for $R^*$ is
Let’s convert \((a \cup b)^*aba\) to an NFA.
RE TO NFA TO DFA

- Regular Expression $\rightarrow$ NFA (possibly with $\epsilon$-transitions)
RE TO NFA TO DFA

- Regular Expression $\rightarrow$ NFA (possibly with $\epsilon$-transitions)
- NFA $\rightarrow$ DFA via determinization
THEOREM

A language is regular if and only if some regular expression describes it.
**Theorem**

A language is regular if and only if some regular expression describes it.

**Lemma – The only if part**

If a language is regular then it is described by a regular expression.
A language is regular if and only if some regular expression describes it.

If a language is regular then it is described by a regular expression.

**Generalized transitions:** label transitions with regular expressions.
EQUIVALENCE WITH FINITE AUTOMATA

THEOREM
A language is regular if and only if some regular expression describes it.

LEMMMA – THE only if PART
If a language is regular then it is described by a regular expression

PROOF IDEA
- Generalized transitions: label transitions with regular expressions
- Generalized NFAs (GNFA)
A language is regular if and only if some regular expression describes it.

If a language is regular then it is described by a regular expression.

Generalized transitions: label transitions with regular expressions

Generalized NFAs (GNFA)

Iteratively eliminate states of the GNFA one by one, until only two states and a single generalized transition is left.
**Generalized Transitions**

- DFAs have single symbols as transition labels

  ![Diagram](image)

- If you are in state $p$ and the next input symbol matches $a$, go to state $q$
**GENERALIZED TRANSITIONS**

- DFAs have single symbols as transition labels

```
\[ p \xrightarrow{a} q \]
```

- If you are in state $p$ and the next input symbol matches $a$, go to state $q$

- Now consider

```
\[ p \xrightarrow{ab^* \cup bc^*} q \]
```
GENERALIZED TRANSITIONS

- DFAs have single symbols as transition labels

- If you are in state $p$ and the next input symbol matches $a$, go to state $q$

- Now consider

- If you are in state $p$ and a prefix of the remaining input matches the regular expression $ab^* \cup bc^*$ then go to state $q$
A generalized transition is a transition whose label is a regular expression
A generalized transition is a transition whose label is a regular expression.
A generalized transition is a transition whose label is a regular expression.

A Generalized NFA is an NFA with generalized transitions.
A generalized transition is a transition whose label is a regular expression.

A Generalized NFA is an NFA with generalized transitions.

In fact, all standard DFA transitions are generalized transitions with regular expressions of a single symbol!
GENERALIZED TRANSITIONS

Consider the 2-state DFA
Consider the 2-state DFA
Consider the 2-state DFA

0*1 takes the DFA from state $q_0$ to $q_1$
Consider the 2-state DFA

- $0^*1$ takes the DFA from state $q_0$ to $q_1$
- $(0 \cup 10^*1)^*$ takes the machine from $q_1$ back to $q_1$
GENERALIZED TRANSITIONS

Consider the 2-state DFA

- $0^*1$ takes the DFA from state $q_0$ to $q_1$
- $(0 \cup 10^*1)^*$ takes the machine from $q_1$ back to $q_1$
- So $? = 0^*1(0 \cup 10^*1)^*$ represents all strings that take the DFA from state $q_0$ to $q_1$
GENERALIZED NFAs

- Take any NFA and transform it into a GNFA
  - with only two states: one start and one accept
GENERALIZED NFAs

Take any NFA and transform it into a GNFA

- with only two states: one start and one accept
- with one generalized transition
GENERALIZED NFAs

- Take any NFA and transform it into a GNFA
  - with only two states: one start and one accept
  - with one generalized transition

- then we can “read” the regular expression from the label of the generalized transition (as in the example above)