Regular Expressions

COMP 455 - 002, Spring 2019
Regular Expressions

- Regular expressions are simply algebraic notation for defining languages.
- A regular expression defines a language.
- In practice, regular expressions can usually define languages in a concise “user-friendly” manner.
  - At least compared to describing a finite automaton…
Definition of Regular Expressions

Regular expressions are defined by taking the union, concatenation, and closure of languages.

- **Union**: $L_1 \cup L_2 \equiv \{ x \mid x \in L_1 \text{ or } x \in L_2 \}$
- **Concatenation**: $L_1 L_2 \equiv \{ xy \mid x \in L_1 \text{ and } y \in L_2 \}$
- **Closure** ("Kleene closure"): $L^* \equiv \bigcup_{i \geq 0} L^i$, where:
  - $L^0 = \{ \varepsilon \}$
  - $L^1 = L$
  - $L^i$ = the concatenation of $i$ copies of $L$
Kleene Closure: Example

- Let language $L = \{00, 11\}$.
- (From the previous definition) $L^i$ represents $i$ strings from $L$ concatenated together.
  - $L^2 = \{0000, 0011, 1100, 1111\}$
  - $L^3 = \{000000, 000011, 001100, \ldots\}$
- $L^*$ is any number of strings from $L$ concatenated together.
  - $L^* = \{\varepsilon, 00, 11, 0000, 0011, 1111, 000000, \ldots\}$
Definition of Regular Expressions

Regular expressions are formally recursively defined (for an alphabet \( \Sigma \)):

1. \( \emptyset \) is a regular expression denoting the empty set.
2. \( \varepsilon \) is a regular expression denoting \( \{ \varepsilon \} \).
3. For each \( a \in \Sigma \), \( a \) is a regular expression denoting \( \{ a \} \).
4. If \( E \) and \( F \) are regular expressions denoting \( L(E) \) and \( L(F) \) (respectively), then:
   - \( (E + F) \) is a regular expression denoting \( L(E) \cup L(F) \)
   - \( (EF) \) is a regular expression denoting \( L(E)L(F) \)
   - \( (E^*) \) is a regular expression denoting \( L(E)^* \).

Note that we will use \textbf{bold} characters for literal symbols in regular expressions.
Example Regular Expressions

(The following examples assume $\Sigma = \{0, 1\}$.)

- $01$
  - The language consisting only of the string $01$.
- $0 + 1$
  - The language consisting of the strings $\{0, 1\}$.
- $01 + 10$
  - The language consisting of the strings $\{01, 10\}$.
Example Regular Expressions

(The following examples assume $\Sigma = \{0, 1\}$.)

- $10(0 + 1)$
  - The language consisting of $\{100, 101\}$.

- $10(0 + 1)^*$
  - The language consisting of all strings that start with 10 and are followed by any number of 0s or 1s.

- $10(0 + 1)^* + \varepsilon$
  - The same language as above, but also containing the empty string.
Regular expressions and finite automata define the same class of languages.

The rest of this presentation covers these proofs.
From DFAs to Regular Expressions

**Theorem 3.4:** If \( L = L(M) \) for some DFA \( M \), then a regular expression \( R \) exists for which \( L = L(R) \).

**Proof:**

- Let \( M = \{\{q_1, q_2, \ldots, q_n\}, \Sigma, \delta, q_1, F\} \)
- Let \( R_{ij}^k \equiv \text{all strings that take us from } q_i \text{ to } q_j \text{ without entering a state higher than } q_k \).
- We can re-label any set of states to start with \( q_1 \).

Note: \( R_{ij}^n = \text{all strings that go from state } q_i \text{ to } q_j \).
Proof of Theorem 3.4, continued

Let $R_{ij}^k \equiv$ all strings that take us from $q_i$ to $q_j$ without entering a state higher than $q_k$.

Define $R_{ij}^k$ as follows:

$$R_{ij}^k = R_{ik}^{k-1}(R_{kk}^{k-1})^* R_{kj}^{k-1} \cup R_{ij}^{k-1}$$

$$R_{ij}^0 = \{a \mid \delta(q_i, a) = q_j\}, \text{ if } i \neq j$$

$$R_{ii}^0 = \{a \mid \delta(q_i, a) = q_i\} \cup \{\varepsilon\}$$
Proof of Theorem 3.4, continued

Define $R_{ij}^k$ as follows:

To get from $q_i$ to $q_j$, potentially through $q_k$...

\[
R_{ij}^k = R_{ik}^{k-1} (R_{kk}^{k-1})^* R_{kj}^{k-1} \cup R_{ij}^{k-1}
\]

\[
R_{ij}^0 = \{a | \delta(q_i, a) = q_j\}, \text{ if } i \neq j
\]

\[
R_{ii}^0 = \{a | \delta(q_i, a) = q_i\} \cup \{\varepsilon\}
\]

Followed by any number of strings that cycle back to state $q_k$...

Followed by strings that get from state $q_k$ to $q_j$.

Also include strings that go from $q_i$ to $q_j$ without going through $q_k$.

$R_{ij}^k \equiv$ all strings that take us from $q_i$ to $q_j$ without entering a state higher than $q_k$. 

Jim Anderson (modified by Nathan Otterness)
Proof of Theorem 3.4, continued

Let $R_{ij}^k \equiv$ all strings that take us from $q_i$ to $q_j$ without entering a state higher than $q_k$.

Define $R_{ij}^k$ as follows:

$R_{ij}^k = R_{ik}^{k-1}(R_{kk}^{k-1})^* R_{kj}^{k-1} \cup R_{ij}^{k-1}$

$R_{ii}^0 = \{ a \mid \delta(q_i, a) = q_i \} \cup \{ \varepsilon \}$

Strings that go directly from $q_i$ to $q_j$.

These are the symbols that cause transitions from $q_i$ to $q_j$. 
Proof of Theorem 3.4, continued

Let $R_{ij}^k \equiv \text{all strings that take us from } q_i \text{ to } q_j \text{ without entering a state higher than } q_k$.

Define $R_{ij}^k$ as follows:

$$R_{ij}^k = R_{ik}^{k-1}(R_{kk}^{k-1})^* R_{kj}^{k-1} \cup R_{ij}^{k-1}$$

Strings that loop from $q_i$ back to $q_i$.

$$R_{ii}^0 = \{a \mid \delta(q_i, a) = q_i\} \cup \{\varepsilon\}$$

$$R_{ij}^0 = \{a \mid \delta(q_i, a) = q_j\}, \text{ if } i \neq j$$
Proof of Theorem 3.4, continued

Claim: There exists a regular expression $r_{ij}^k$ denoting $R_{ij}^k$. We will prove this by induction on $k$.

Basis: $k = 0$

Let $A \equiv \{a_1, a_2, \ldots, a_p\}$ denote the symbols that take $q_i$ to $q_j$. Then,

$$r_{ij}^0 = \begin{cases} 
  a_1 + a_2 + \cdots + a_p & \text{, if } A \neq \emptyset \text{ and } i \neq j \\
  \emptyset & \text{, if } A = \emptyset \text{ and } i \neq j \\
  a_1 + a_2 + \cdots + a_p + \varepsilon & \text{, if } A \neq \emptyset \text{ and } i = j \\
  \varepsilon & \text{, if } A = \emptyset \text{ and } i = j 
\end{cases}$$
Proof of Theorem 3.4, continued

**Claim:** There exists a regular expression $r_{ij}^k$ denoting $R_{ij}^k$. We will prove this by induction on $k$.

**Inductive step:** $k > 0$.

In this case,

$$r_{ij}^k = r_{ik}^{k-1}(r_{kk}^{k-1})^* r_{kj}^{k-1} + r_{ij}^{k-1}$$
Proof of Theorem 3.4, continued

Claim: There exists a regular expression $r_{ij}^k$ denoting $R_{ij}^k$. We will prove this by induction on $k$.

Inductive step: $k > 0$.
In this case,

$$r_{ij}^k = r_{ik}^{k-1} (r_{kk}^{k-1})^* r_{kj}^{k-1} + r_{ij}^{k-1}$$

The inductive hypothesis lets us assume that we already have these regular expressions!
Proof of Theorem 3.4, continued

**Finishing up:**

We still need to show that we can construct a regular expression that matches $L(M)$.

We note that $L(M) = \bigcup_{q_j \in F} R_{1j}^n$.

Therefore, $L(M)$ is denoted by the regular expression:

$$r_{1j_1}^n + r_{1j_2}^n + \cdots + r_{1j_p}^n,$$

where $F = \{q_{j_1}, q_{j_2}, \ldots, q_{j_p}\}$.

Any string that takes us from state $q_1$ to $q_j$, where $q_j$ is a final state.
Example: DFA to Regular Expression

<table>
<thead>
<tr>
<th></th>
<th>$k = 0$</th>
<th>$k = 1$</th>
<th>$k = 2$</th>
<th>$k = 3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r^k_{11}$</td>
<td>$\varepsilon$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$r^k_{12}$</td>
<td>$0$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$r^k_{13}$</td>
<td>$1$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$r^k_{21}$</td>
<td>$\emptyset$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$r^k_{22}$</td>
<td>$0 + 1 + \varepsilon$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$r^k_{23}$</td>
<td>$\emptyset$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$r^k_{31}$</td>
<td>$\emptyset$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$r^k_{32}$</td>
<td>$\emptyset$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$r^k_{33}$</td>
<td>$0 + 1 + \varepsilon$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

If $A$ is the set of symbols going from $q_i$ to $q_j$:

$$r^0_{ij} = \begin{cases} a_1 + a_2 + \cdots + a_p, & \text{if } A \neq \emptyset \text{ and } i \neq j \\ \emptyset, & \text{if } A = \emptyset \text{ and } i \neq j \\ a_1 + a_2 + \cdots + a_p + \varepsilon, & \text{if } A \neq \emptyset \text{ and } i = j \\ \varepsilon, & \text{if } A = \emptyset \text{ and } i = j \end{cases}$$
# Example: DFA to Regular Expression

Consider the DFA and its corresponding regular expression:

<table>
<thead>
<tr>
<th>$r_{ij}^k$</th>
<th>$k = 0$</th>
<th>$k = 1$</th>
<th>$k = 2$</th>
<th>$k = 3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_{11}^k$</td>
<td>$\varepsilon$</td>
<td>$\varepsilon$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$r_{12}^k$</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$r_{13}^k$</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$r_{21}^k$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$r_{22}^k$</td>
<td>$0 + 1 + \varepsilon$</td>
<td>$0 + 1 + \varepsilon$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$r_{23}^k$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$r_{31}^k$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$r_{32}^k$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$r_{33}^k$</td>
<td>$0 + 1 + \varepsilon$</td>
<td>$0 + 1 + \varepsilon$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The transition diagram and the regular expression equation are as follows:

- Transition Diagram:
  - Start state $q_1$ with transitions:
    - $q_1 \xrightarrow{0} q_2$
    - $q_1 \xrightarrow{1} q_3$
    - $q_2 \xrightarrow{0, 1} q_1$
    - $q_3 \xrightarrow{0, 1} q_3$

- Regular Expression Equation:
  $$r_{ij}^k = r_{ik}^{k-1} (r_{kk}^{k-1})^* r_{kj}^{k-1} + r_{ij}^{k-1}$$
Example: DFA to Regular Expression

<table>
<thead>
<tr>
<th></th>
<th>$k = 0$</th>
<th>$k = 1$</th>
<th>$k = 2$</th>
<th>$k = 3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_{11}^k$</td>
<td>$\varepsilon$</td>
<td>$\varepsilon$</td>
<td>$\varepsilon$</td>
<td></td>
</tr>
<tr>
<td>$r_{12}^k$</td>
<td>0</td>
<td>0</td>
<td>$0(0 + 1)^*$</td>
<td></td>
</tr>
<tr>
<td>$r_{13}^k$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$r_{21}^k$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td></td>
</tr>
<tr>
<td>$r_{22}^k$</td>
<td>$0 + 1 + \varepsilon$</td>
<td>$0 + 1 + \varepsilon$</td>
<td>$(0 + 1)^*$</td>
<td></td>
</tr>
<tr>
<td>$r_{23}^k$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td></td>
</tr>
<tr>
<td>$r_{31}^k$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td></td>
</tr>
<tr>
<td>$r_{32}^k$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td></td>
</tr>
<tr>
<td>$r_{33}^k$</td>
<td>$0 + 1 + \varepsilon$</td>
<td>$0 + 1 + \varepsilon$</td>
<td>$0 + 1 + \varepsilon$</td>
<td></td>
</tr>
</tbody>
</table>

$r_{ij}^k = r_{ik}^{k-1} \left(r_{kk}^{k-1}\right)^* r_{kj}^{k-1} + r_{ij}^{k-1}$
### Example: DFA to Regular Expression

<table>
<thead>
<tr>
<th>$r_{11}^k$</th>
<th>$k = 0$</th>
<th>$k = 1$</th>
<th>$k = 2$</th>
<th>$k = 3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\epsilon$</td>
<td>$\epsilon$</td>
<td>$\epsilon$</td>
<td>$\epsilon$</td>
<td></td>
</tr>
<tr>
<td>$r_{12}^k$</td>
<td>0</td>
<td>0</td>
<td>$0(0 + 1)^*$</td>
<td>$0(0 + 1)^*$</td>
</tr>
<tr>
<td>$r_{13}^k$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>$1(0 + 1)^*$</td>
</tr>
<tr>
<td>$r_{21}^k$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>$r_{22}^k$</td>
<td>$0 + 1 + \epsilon$</td>
<td>$0 + 1 + \epsilon$</td>
<td>$(0 + 1)^*$</td>
<td>$(0 + 1)^*$</td>
</tr>
<tr>
<td>$r_{23}^k$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>$r_{31}^k$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>$r_{32}^k$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>$r_{33}^k$</td>
<td>$0 + 1 + \epsilon$</td>
<td>$0 + 1 + \epsilon$</td>
<td>$0 + 1 + \epsilon$</td>
<td>$(0 + 1)^*$</td>
</tr>
</tbody>
</table>

The transition function $r_{ij}^k$ is defined as:

$$r_{ij}^k = r_{ik}^{k-1}(r_{kk}^{k-1})^* r_{kj}^{k-1} + r_{ij}^{k-1}$$
Example: DFA to Regular Expression

<table>
<thead>
<tr>
<th>$r_k^{k_1}$</th>
<th>$k = 0$</th>
<th>$k = 1$</th>
<th>$k = 2$</th>
<th>$k = 3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_{11}^k$</td>
<td>$\varepsilon$</td>
<td>$\varepsilon$</td>
<td>$\varepsilon$</td>
<td>$\varepsilon$</td>
</tr>
<tr>
<td>$r_{12}^k$</td>
<td>0</td>
<td>0</td>
<td>0$(0 + 1)^*$</td>
<td>0$(0 + 1)^*$</td>
</tr>
<tr>
<td>$r_{13}^k$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1$(0 + 1)^*$</td>
</tr>
<tr>
<td>$r_{21}^k$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>$r_{22}^k$</td>
<td>0 + 1 + $\varepsilon$</td>
<td>0 + 1 + $\varepsilon$</td>
<td>(0 + 1)$^*$</td>
<td>(0 + 1)$^*$</td>
</tr>
<tr>
<td>$r_{23}^k$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>$r_{31}^k$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>$r_{32}^k$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>$r_{33}^k$</td>
<td>0 + 1 + $\varepsilon$</td>
<td>0 + 1 + $\varepsilon$</td>
<td>0 + 1 + $\varepsilon$</td>
<td>(0 + 1)$^*$</td>
</tr>
</tbody>
</table>

The expression taking us from $q_1$ to $q_2$ through states up to $q_3$ is $r_{12}^3$: 0$(0 + 1)^*$.
Comments on this Approach

- This approach is an example of dynamic programming, a COMP 550 topic.
- For an $n$-state DFA, this approach may produce a regular expression with $4^n$ symbols!
- Section 3.2.2 of the book presents a more efficient method.
From Regular Expressions to $\varepsilon$-NFAs

**Theorem 3.7** (reworded): If $R$ is a regular expression, $L(R) = L(E)$ for some NFA with $\varepsilon$-transitions $E$.

- We will construct $E$ with one final state and *no transitions out of that state*.
- We will do this by induction on the number of operators in $R$. 
Proof of Theorem 3.7

**Base case:** $R$ has no operators. There are three possibilities here:

- $R = \epsilon$

  ![Diagram](image1)

- $R = \emptyset$

  ![Diagram](image2)

- $R = a$

  ![Diagram](image3)
Proof of Theorem 3.7, continued

Inductive step:
We now need to handle the three regular-expression operators:

- Union ($R_1 + R_2$)
- Concatenation ($R_1 R_2$)
- Closure ($R_1^*$)
**Inductive step:** Union: $R = R_1 + R_2$

We can assume by the inductive hypothesis that we already have $\varepsilon$-NFAs $E_1$ and $E_2$ accepting the same languages as $R_1$ and $R_2$. Construct the $\varepsilon$-NFA $E$ to accept the same language as $R$ like this:
Proof of Theorem 3.7, continued

**Inductive step**: Concatenation: $R = R_1 R_2$

Once again, assume $E_1$ and $E_2$ are $\varepsilon$-NFAs accepting the same languages as $R_1$ and $R_2$. Construct the $\varepsilon$-NFA $E$ to accept the same language as $R$ like this:
Proof of Theorem 3.7, continued

**Inductive step**: Closure: \( R = R_1^* \)

Assume \( E_1 \) is an \( \varepsilon \)-NFA accepting the same language as \( R_1 \). Construct the \( \varepsilon \)-NFA \( E \) to accept the same language as \( R \) like this:
Example Regular Expression Conversion

We’ll convert the regular expression $0(0 + 1)^*$ to an NFA with $\varepsilon$-transitions.

0:
\[
\begin{array}{cc}
\text{Start} & \rightarrow \text{a}^0 \rightarrow \text{b} \\
\end{array}
\]

1:
\[
\begin{array}{cc}
\text{Start} & \rightarrow \text{c}^1 \rightarrow \text{d} \\
\end{array}
\]

$0 + 1$:
\[
\begin{array}{cc}
\text{Start} & \rightarrow \text{e}^\varepsilon \rightarrow \text{a}^0 \rightarrow \text{b}^\varepsilon \\
\end{array}
\]

\[
\begin{array}{cc}
\text{Start} & \rightarrow \text{c}^\varepsilon \rightarrow \text{d}^1 \rightarrow \text{f}^\varepsilon \\
\end{array}
\]
Example Regular Expression Conversion

0 + 1:

(0 + 1)*:
Example Regular Expression Conversion

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

\(0(0 + 1)^*:\)
Example Regular Expression Conversion

$0(0 + 1)^*$:

Note: A much simpler machine exists.