Regular Expressions

Formal Definition

- For primitive regular expressions:
  \[ L(\emptyset) = \emptyset \]
  \[ L(\lambda) = \{ \lambda \} \]
  \[ L(a) = \{ a \} \]

Definition (continued)

- For regular expressions \( r_1 \) and \( r_2 \):
  \[ L(r_1 + r_2) = L(r_1) \cup L(r_2) \]
  \[ L(r_1 \cdot r_2) = L(r_1) \cdot L(r_2) \]
  \[ L(r_1^*) = (L(r_1))^* \]
  \[ L(\langle r_1 \rangle) = L(r_1) \]
Example

\[ L((a + b) \cdot a^*) = L((a + b)) L(a^*) \]
\[ = L(a + b) L(a^*) \]
\[ = (L(a) \cup L(b)) (L(a))^* \]
\[ = (\{a\} \cup \{b\}) (\{a\})^* \]
\[ = \{a, b\} \{\lambda, a, aa, aaa, \ldots\} \]
\[ = \{a, aa, aaa, \ldots, b, ba, baa, \ldots\} \]

Example

- Regular expression
  \[ r = (a + b)^* (a + bb) \]

\[ L(r) = \{a, bb, aa, abb, ba, bbb, \ldots\} \]

Example

- Regular expression
  \[ r = (aa)^* (bb)^* b \]

\[ L(r) = \{a^{2n} b^{2m} b : n, m \geq 0\} \]

Example

- Regular expression
  \[ r = (0 + 1)^* 00 (0 + 1)^* \]

\[ L(r) = \{\text{all strings with at least two consecutive } 0\} \]

Can we make an expression for the complement of this language?
All strings without two consecutive 0’s?
Example

• Regular expression

\[ r = (1 + 01)^* (0 + \lambda) \]

\[ L(r) = \{ \text{all strings without two consecutive 0} \} \]

Equivalent Regular Expressions

• Definition:

Regular expressions \( r_1 \) and \( r_2 \) are equivalent if \( L(r_1) = L(r_2) \).

Example

\[ L = \{ \text{all strings without two consecutive 0} \} \]

\[ r_1 = (1 + 01)^* (0 + \lambda) \]

\[ r_2 = (1*011*)^* (0 + \lambda) + 1^* (0 + \lambda) \]

\[ L(r_1) = L(r_2) = L \quad \Rightarrow \quad r_1 \text{ and } r_2 \text{ are equivalent regular expressions.} \]
Theorem

\[ \{ \text{Languages Generated by Regular Expressions} \} = \{ \text{Regular Languages} \} \]

Proof - Part 1

1. For any regular expression \( r \), the language \( L(r) \) is regular

Proof by induction on the size of \( r \)

\[ \text{Induction Basis} \quad \emptyset, \lambda, \alpha \]

- NFA \( M_1 \):
  \[ L(M_1) = \emptyset = L(\emptyset) \]

- NFA \( M_2 \):
  \[ L(M_2) = \{\lambda\} = L(\lambda) \]

- NFA \( M_3 \):
  \[ L(M_3) = \{a\} = L(a) \]

Regular Languages
Inductive Hypothesis

Assume for regular expressions \( r_1 \) and \( r_2 \) that \( L(r_1) \) and \( L(r_2) \) are regular languages.

Inductive Step

Prove:

- \( L(r_1 + r_2) \)
- \( L(r_1 \cdot r_2) \)
- \( L(r_1^*) \)
- \( L((r_1)) \)

By definition of regular expressions:

- \( L(r_1 + r_2) = L(r_1) \cup L(r_2) \)
- \( L(r_1 \cdot r_2) = L(r_1)L(r_2) \)
- \( L(r_1^*) = (L(r_1))^* \)
- \( L((r_1)) = L(r_1) \)

By inductive hypothesis we know:

- \( L(r_1) \) and \( L(r_2) \) are regular languages.

Can we construct a DFA (or equivalently NFA) for:

- **Union**: \( L(r_1) \cup L(r_2) \)
- **Concatenation**: \( L(r_1)L(r_2) \)
- **Star**: \( (L(r_1))^* \)

Let's Work Out a DFA for Each of These
• Since we can construct those DFA:

\[
L(n_1 + r_2) = L(n_1) \cup L(r_2)
\]

\[
L(n_1 \cdot r_2) = L(n_1) L(r_2)
\]  
\[
L(n_1)^* = (L(n_1))^*
\]

Are regular languages

To Complete the Proof…

And trivially:

\[
L((n_1))\text{ is a regular language}
\]

Theorem - Part 2

\[
\left\{ \text{Regular Languages} \right\} \subseteq \left\{ \text{Languages Generated by Regular Expressions} \right\}
\]

2. For any regular language \( L \) there is a regular expression \( r \) with \( L(r) = L \)

Proof – Part 2

2. For any regular language \( L \) there is a regular expression \( r \) with \( L(r) = L \)

Proof by construction of regular expression
Since \( L \) is regular take the NFA \( M \) that accepts it

\[
L(M) = L
\]

Single final state

- From \( M \) construct the equivalent Generalized Transition Graph in which transition labels are regular expressions

Example: \( M \)

\[ a 
\] \[ a+b 
\] \[ c 
\]
The final transition graph:

The resulting regular expression:

\[ r = \eta_1 \ast r_2 (r_4 + \eta_1 \ast r_2) \ast \]

\[ L(r) = L(M) = L \]