Regular expressions

- used for describing string patterns, e.g.

\[(0 \cup 1)0^*\]  means \[\{0\} \cup \{1\}\{0\}^*\]

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

Formal definition:

R is a **regular expression** if R is one of the following:

1. a for some \( a \in \Sigma \),
2. \( \varepsilon \)
3. \( \emptyset \)
4. \((R_1 \cup R_2)\), where \( R_1, R_2 \) are regular expressions
5. \((R_1 \cdot R_2)\), where \( R_1, R_2 \) are regular expressions
6. \((R_1^*)\), where \( R_1 \) is a regular expression.

Note: this type of definition is called a **recursive/inductive definition** (i.e. the definition is a recursive algorithm).
For convenience: $R^+ = RR^*$

Examples: give regular expressions for the following languages:

- \( \{ w \in \{0,1\}^* \mid w \text{ contains the substring } 001 \} \)
  \[
  (0u1)^*001(0u1)^*
  \]

- \( \{ w \in \{0,1\}^* \mid w \text{ does not contain two consecutive 0's } \} \)
  \[
  1^*(01^*)^*(0u\varepsilon)
  \]

- \( \{ w \in \{0,1\}^* \mid |w| \text{ is divisible by 2 or 3 } \} \)
  \[
  ((0u1)(0u1))^* \cup ((0u1)(0u1)(0u1))^*
  \]

- \( \{ w \in \{0,1\}^* \mid |w| < 4 \} \)
  \[
  (0u1u\varepsilon)^3
  \]
Examples: let \( R \) be any regular expression

- \( R \cdot \emptyset = \emptyset \)
- \( R \cdot \{ \epsilon \} = R \)
- \( \emptyset^* = \{ \epsilon \} \)
- \( \epsilon^* = \{ \epsilon \} \)

The language defined by \( R \) is denoted \( L(R) \). We’ll often abuse notation and use \( R \) to denote the language \( L(R) \).
Thm 1.54: A language is regular iff some regular expression describes it.

Lemma 1.55: Given a regular expression \( R \), there exists a FA \( M \) such that \( L(M) = L(R) \).

Lemma 1.60: Given a FA \( M \), there exists a regular expression \( R \) such that \( L(R) = L(M) \).
Equivalence of reg. expr. and FA's

Lemma 1.55: Given a regular expression \( R \), there exists a FA \( M \) such that \( L(M) = L(R) \).

Proof: by structural induction on \( R \) we will show that there exists an NFA \( N \) for \( R \)

i.e. \( L(N) = L(R) \)

base cases: 1) \( R = a \) for some \( a \in \Sigma \)

then \( N: \begin{array}{c} \circ \rightarrow a \rightarrow \circ \end{array} \)

2) \( R = \varepsilon \), then \( N: \begin{array}{c} \circ \rightarrow \circ \end{array} \)

3) \( R = \emptyset \), then \( N: \begin{array}{c} \circ \rightarrow \circ \end{array} \)

inductive cases: if \( R \) is formed by rule 4):

\( R = R_1 \cup R_2 \) for some \( R_1, R_2 \)

by the inductive hypothesis, we assume:

we have an NFA \( N_i \) for \( R_i \) \( \text{i.e. \{1,2\}} \)

construct \( N \) using Thm 1.45

for rule 5): apply Thm 1.47

6) 1.49
Lemma 1.60: Given a FA $M$, there exists a regular expression $R$ such that $L(R) = L(M)$.

Proof idea:

Generalized NFA (GNFA)
- Transitions may be marked by reg. expr. (not just $\Sigma \cup \{\varepsilon\}$)
- Single accept state that:
  - a) has arrows coming in from every other state,
  - b) does not have any outgoing arrows
- Start state that:
  - a) has arrows to every other state,
  - b) does not have any incoming arrows
- All other states have arrows to all other states
**Lemma 1.60:** Given a FA $M$, there exists a regular expression $R$ such that $L(R) = L(M)$.

**Proof idea:**

Generalized NFA (GNFA) $(Q, \Sigma, \delta, q_{\text{start}}, q_{\text{accept}})$ where all as usual except $\delta : (Q - \{q_{\text{accept}}\}) \times (Q - \{q_{\text{start}}\}) \rightarrow R$ where $R$ is the set of all regular expressions over $\Sigma$.

Idea: start with a GNFA, remove states one by one and redraw arrows as necessary.

**How to get a GNFA:**

1. Start with a GNFA.
2. Remove a state $q_i$.
3. Update the regular expression.
4. Update self-loop $q_i$ and $q_j$.
Lemma 1.60: Given a FA $M$, there exists a regular expression $R$ such that $L(R) = L(M)$.

Proof idea:

How to construct an equivalent GNFA with one fewer state?

In terms of the $\delta$-inc:

$$\delta_{\text{new}}(p,r) = \delta_{\text{old}}(p,r) \cup \delta_{\text{old}}(p,q)\delta_{\text{old}}(q,q)^*\delta_{\text{old}}(q,r)$$

$$\forall q \in Q \setminus \{q_{\text{accept}}\}$$

$$\forall r \in Q \setminus \{q_{\text{start}}\}$$

Ultimately, we get:

$$\xrightarrow{R}$$

where $R$ is the resulting reg. expr.