Regular expressions

- used for describing string patterns, e.g.

\[(0 \cup 1)0^*\] means any string from the language \((\{0\} \cup \{1\}). \{0\}^*\)

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

For convenience: $R^+ = RR^*$

Examples: give regular expressions for the following languages:

- $\{ w \in \{0,1\}^* \mid w \text{ contains the substring 001 } \}$
  
  $$(01)^* 001 (01)^*$$

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

- $\{ w \in \{0,1\}^* \mid |w| \text{ is divisible by 2 or 3 } \}$
  
  $$((01)^2)^* \cup ((01)^3)^* \quad \text{ means: } \quad ((01)(01))^* \cup ((01)(01)(01))^*$$

- $\{ w \in \{0,1\}^* \mid |w| < 4 \}$
  
  $$\epsilon \cup (01) \cup (01)^2 \cup (01)^3 \quad \text{ or } \quad (01 \epsilon)^3$$
Examples: let R be any regular expression

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

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 \( L(R) = L(M) \).
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 $L(R) = L(M)$.

Proof idea:

Generalized NFA (GNFA)

- transitions may be marked by reg.expr. (not just $\Sigma \cup \{\epsilon\}$)
- 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:
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?