Regular expressions

- used for describing string patterns, e.g.

\[(0 \cup 1)0^*\]  - 10000 follows the pattern but 101 does not

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

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

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

- $\{ w \in \{0,1\}^* \mid \text{\# of }0s \text{ is divisible by 2 or 3} \}$
  $$ ((0u1)(0u1))^* \cup ((0u1)(0u1)(0u1))^* $$
  short-hand: $$ (0u1)^2)^* \cup (0u1)^3)^* $$

- $\{ w \in \{0,1\}^* \mid |w| < 4 \}$
  $$ \varepsilon \cup 0u1 \cup (0u1)^2 \cup (0u1)^3 \cup (0u1\varepsilon)^3 $$
Regular expressions

Examples: let R be any regular expression

- $R \cdot \emptyset = \emptyset$
- $R \cdot \varepsilon = R$
- $\emptyset^* = \varepsilon$
- $\varepsilon^* = \varepsilon$

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

Proof: by structural induction, we'll show that for any $R$, there exists an equiv. NFA $M$

**BASE CASES:**
1) $R = a$ for some $a \in \Sigma$ then $M: \overset{a}{\rightarrow} N$
2) $R = \varepsilon$ then $M: \rightarrow N$
3) $R = \phi$ then $M: \rightarrow O$

**IND. CASES:**
4) $R = (R_1 U R_2)$ for some reg. expr. $R_1, R_2$
   
   by the inductive hypothesis: we have NFAs $M_1$ and $M_2$ for $R_1$ and $R_2$
   
   then, we'll construct NFA $M$ as follows: use construction from Thm 1.45

5) 6) similar to 4)
   
   but use constructions from Thms 1.47 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:

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

![Diagram](image.png) until we have only 2 states:
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?

\[ \text{update } R_{pq} : \quad UR_{pr} R_{rr}^* R_{rq} \]

do for all transitions: \( \forall p \in Q - \{ r, q_{\text{accept}} \} \)
\( \forall q \in Q - \{ r, q_0 \} \)