Regular expressions

- used for describing string patterns, e.g.

\[(0 \cup 1)0^*\]  e.g. 000, 10, 1

\[(0 \cup 1)^*\]  e.g. \(\varepsilon, 0, 100\)
Regular expressions

Formal definition:

$\Sigma = \{0, 1\}$

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

1. $a$ for some $a \in \Sigma$, $\rightarrow$ e.g. $R = 0$
2. $\varepsilon$
3. $\emptyset$
4. $(R_1 \cup R_2)$, where $R_1, R_2$ are regular expressions
5. $(R_1 . R_2)$, 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:

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

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

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

- $\{ w \in \{0,1\}^* \mid |w| < 4 \}$
  
  $(\varepsilon u 0u1)^3$
Examples: let $R$ be any regular expression

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

The language defined by $R$ is denoted $L(R)$. We’ll often abuse notation and use $R$ to denote the language $L(R)$. 
Equivalence of reg. expr. and FA's

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

Pf: by "structural induction" we will show that for $R$ there exists a NFA $M$ s.t. $L(M) = L(R)$ and $M$ has a single accept. state.

**BASE CASES:**
1. $R = a$ for some $a \in \Sigma$ then $M$: $\xrightarrow{a} \circ$
2. $R = \varepsilon$ then $M$: $\xrightarrow{\circ}$
3. $R = \emptyset$ then $M$: $\xrightarrow{\circ \circ}$

**INDUCTIVE CASES:**
4. $R = (R_1 \cup R_2)$ assume (by IH) that there is a NFA $N_1$ for $R_1$, $(N_2)$ $(R_2)$ then just construct $N$ by following the lemma's from previous class e.g. $\xrightarrow{\circ \circ} \emptyset$
5. $R = (R_1 \cdot R_2)$
6. $R = (R_1)^*$
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

(yellow was original automaton)

(colorful mess is an equivalent GNFA)
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 $\delta: (Q-\{q_{\text{accept}}\}) \times (Q-\{q_{\text{start}}\}) \rightarrow \mathcal{R}$ where $\mathcal{R}$ 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: \textit{DONE}
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?