COMP2210: Theory of Computation

Lecture 4

Kleene’s Theorem (continuation)

Gennaro Parlato
Learning Outcomes

By the end of this session you should be able to...

• . . . build an NFA from a regular expression
Given $\Sigma$, the set of regular expressions is defined inductively. 

Base cases:

- Every $\sigma \in \Sigma$ is a regular expression
  - $L(\sigma) \overset{\text{def}}{=} \{\sigma\}$
- $\epsilon$ is a reg. exp.
  - $L(\epsilon) \overset{\text{def}}{=} \{\epsilon\}$
- $\emptyset$ is a reg. exp.
  - $L(\emptyset) \overset{\text{def}}{=} \emptyset$
Regular expressions, II (recall)

Operations:

- If $\alpha, \beta$ are regular expressions then $\alpha + \beta$ is a regular expression.
  - $L(\alpha + \beta) \overset{\text{def}}{=} L(\alpha) \cup L(\beta)$
  - Some authors write $\alpha | \beta$ instead of $\alpha + \beta$

- If $\alpha, \beta$ are regular expressions then $\alpha\beta$ is a regular expression.
  - $L(\alpha\beta) \overset{\text{def}}{=} L(\alpha)L(\beta)$

- If $\alpha$ is a regular expression then $\alpha^*$ is a regular expression.
  - $L(\alpha^*) \overset{\text{def}}{=} L(\alpha)^*$
Examples (recall)

- We say that a string $s$ matches a regular expression $\alpha$ whenever $s \in L(\alpha)$.
  - no strings match $\emptyset$ (since $L(\emptyset) = \emptyset$);
  - strings that end in $b$ match $(a + b)^*b$;

$$L((a + b)^*b) = L((a + b)^*)L(b) = L(a + b)^*L(b)$$

$$= (L(a) \cup L(b))^*\{b\} = \{a, b\}^*\{b\}$$

$$= \{x_1 \ldots x_n y \mid n \in \mathbb{N}, x_i \in \{a, b\}, y \in \{b\}\}$$

- strings of even length match $((a + b)(a + b))^*$;
- $abba$ and the empty string match $abba + \epsilon$;
Kleene’s Theorem

Theorem.

• If $\alpha$ is a regexp then $L(\alpha)$ is a regular language (last lecture)

• If $L$ is a regular language then $L = L(\alpha)$ for some regexp $\alpha$ (today)

• In other words, finite automata and regular expressions describe the same languages!
NFA to reg exp

- Let $M = (Q, \Sigma, \Delta, s, F)$ be an NFA. We will first define a regular expression $\alpha_{u,v}^X$ for any $X \subseteq Q$ and $u, v \in Q$.

- $\alpha_{u,v}^X$ is a regular expression that describes all possible paths from $u$ to $v$ that start with $u$, end with $v$ and have all the intermediate states in $X$.

- We will define $\alpha_{u,v}^X$ by recursion on $X$: first let us define formally what the previous point means formally.
NFA to reg exp

- For $X \subseteq Q$ let $q \xrightarrow{\sigma}^X q'$ be shorthand for $\delta(q, \sigma) = q'$;
- We extend the $\rightarrow^X$ notation to arbitrary strings:

$$q \xrightarrow{\epsilon}^X q' \overset{\text{def}}{=} q = q'$$

$$q \xrightarrow{x\sigma}^X q' \overset{\text{def}}{=} \exists q'' \in X. q \xrightarrow{x}^X q'' \land q'' \xrightarrow{\sigma}^X q'$$

- Clearly $q \xrightarrow{x} q'$ is the same as $q \xrightarrow{\cdot}^Q q'$;

Idea: We want $L(\alpha_{u,v}^X) = \{x \mid u \xrightarrow{x}^X v\}$.
Definition of $\alpha_{u,v}^\varnothing$

- we begin with the base case of $X = \varnothing$. Let $a_1, \ldots, a_k$ be all the symbols such that $\delta(u, a_i) = v$ for $1 \leq i \leq k$

  $u \neq v : \quad \alpha_{uv}^\varnothing \overset{\text{def}}{=} \begin{cases} a_1 + \cdots + a_k & \text{if } k > 0 \\ \varnothing & \text{otherwise} \end{cases}$

  $u = v : \quad \alpha_{uv}^\varnothing \overset{\text{def}}{=} \begin{cases} a_1 + \cdots + a_k + \epsilon & \text{if } k > 0 \\ \epsilon & \text{otherwise} \end{cases}$

- Inductive step: we know $\alpha_{u,v}^X$, how can we define $\alpha_{u,v}^{X+\{q\}}$?
Inductive step

- **Idea:** a path from $u$ to $v$ in $X + \{q\}$ goes through $q$ $k$-times for some $k$...

- $\alpha^X_{u,v}$ - paths that don’t go through $q$;

- $\alpha^X_{u,q}\alpha^X_{q,v}$ - paths that go through $q$ once;

- $\alpha^X_{u,q}\alpha^X_{q,q}\alpha^X_{q,v}$ - paths that go through $q$ twice...
Definition of $\alpha_{u,v}^{X+\{q\}}$

$$
\alpha_{u,v}^{X+\{q\}} \overset{\text{def}}{=} \alpha_{u,v}^X + \alpha_{u,q}^X (\alpha_{q,q}^X)^* \alpha_{q,v}^X
$$
NFA to regular expression

- Suppose that $M = (Q, \Sigma, \delta, s, F)$ is a DFA. We know that:

  \[
  L(M) = \{x \mid \hat{\delta}(x) \in F\} = \{x \mid \exists f \in F. s \xrightarrow{x} Q f\}
  \]

- Let $f_1, \ldots, f_k$ be all the states in $F$. Then the regular expression for $L(M)$ is

  \[
  \alpha^Q_{s, f_1} + \alpha^Q_{s, f_2} + \cdots + \alpha^Q_{s, f_k}.
  \]

- The construction works the same way for NFAs.
Example

\[ \alpha_{0,1}^{\{0,1\}} = \alpha_{0,1}^0 + \alpha_{0,1}^0 \left( \alpha_{1,1}^0 \right)^* \alpha_{1,1}^0 \]
\[ \alpha_{0,1}^{\{0\}} = \alpha_{0,1}^0 + \alpha_{0,0}^0 \left( \alpha_{0,0}^\emptyset \right)^* \alpha_{0,1}^0 \]
\[ = b + (a + \epsilon)(a + \epsilon)^* b \ (== a^* b) \]
\[ = a^* b + a^* b(\epsilon + a^* b)^* (\epsilon + a^* b) \]
\[ (== (a+b)^* b) \]

\[ \alpha_{1,1}^{\{0\}} = \alpha_{1,1}^\emptyset + \alpha_{1,0}^\emptyset \left( \alpha_{0,0}^\emptyset \right)^* \alpha_{0,1}^\emptyset \]
\[ = (b + \epsilon) + a(a + \epsilon)^* b \ (== \epsilon + a^* b) \]