ECE 664

Lecture 19

Avi Kak

REGULAR EXPRESSIONS AND THE PUMPING LEMMA

- Given an alphabet $A = \{s_1, s_2, \ldots, s_k\}$, some of the smallest regular languages we can construct from this alphabet are

$$\{0\}, \{s_1\}, \{s_2\}, \ldots, \{s_k\}, \{s_1 s_1\}, \{s_1, s_2, s_1 s_1\}, \{s_1 s_2 s_3, s_2 s_3\}, \ldots$$

- We can construct additional regular languages on this alphabet by applying the operators $\cup, \bullet, *$ to the languages already constructed.

- Given the fact that we can construct a large (infinite) number of languages from an alphabet set A, is there some way to name them so that we can form a mental picture of this infinite set of regular languages ?

- The answer to the question posed above is: Yes. The naming convention is called regular expressions.

- To formally define a regular expression, we first need to introduce the notion of the naming alphabet.

THE NAMING ALPHABET : For a given alphabet $A = \{s_1, s_2, \ldots, s_k\}$, we declare a naming alphabet

note that $\mathbf{s}_i$ in $\tilde{A}$ is not the same as s_i in A

$$\tilde{A} = \{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_k, \mathbf{0}, \boldsymbol{\phi}, \mathbf{\cup}, \bullet, *, (,)\}$$

- We now define the set of regular expressions as a subset of $\tilde{A}^*$ provided the elements of the subset obey the following rules :

  1. $\boldsymbol{\phi}, \mathbf{0}, \mathbf{s}_1, \ldots, \mathbf{s}_k$ are regular expressions.
  2. If α and β are regular expressions, so is $(\alpha \cup \beta)$
  3. If α and β are regular expressions, so is $(\alpha \bullet \beta)$
  4. If α is a regular expression, so is α^*
  5. No expression is regular unless it can be generated using a finite number of applications of the above four rules.

- We now declare that a regular expression γ will represent a regular language that will be denoted by $\langle \gamma \rangle$. This regular language is to be obtained from the regex γ by applying the following six rules to γ:

  1. $\langle \mathbf{s}_i \rangle = \{s_i\}$
  2. $\langle \mathbf{0} \rangle = \{0\}$
  3. $\langle \boldsymbol{\phi} \rangle = \phi$
  4. $\langle (\alpha \cup \beta) \rangle = \langle \alpha \rangle \cup \langle \beta \rangle$
  5. $\langle (\alpha \bullet \beta) \rangle = \langle \alpha \rangle \cdot \langle \beta \rangle$
  6. $\langle \alpha^* \rangle = \langle \alpha \rangle^*$

- Two simple examples of regular expressions :

① $\gamma = (c^* \cdot b^*)$ → in Perl regex $c*b*$

The regular language represented by this regex is

$$\langle\gamma\rangle = \langle (c^* \cdot b^*)\rangle = \langle c^*\rangle \cdot \langle b^*\rangle = \{c\}^* \cdot \{b\}^*$$
$$= \{c^{[m]} \mid m \geq 0\} \cdot \{b^{[n]} \mid n \geq 0\}$$
$$= \{c^{[m]} b^{[n]} \mid m, n \geq 0\}$$

② $\gamma = (a \cdot (b^* \cup c^*))$

$$\langle\gamma\rangle = \langle a\rangle \cdot \langle (b^* \cup c^*)\rangle = \{a\} \cdot (\langle b\rangle^* \cup \langle c\rangle^*)$$
$$= \{a\} \cdot (\{b^{[m]} \mid m \geq 0\} \cup \{c^{[n]} \mid n \geq 0\})$$
$$= \{a b^{[m]} \mid m \geq 0\} \cup \{a c^{[n]} \mid n \geq 0\}$$

<u>THEOREM</u> : For every regular language $L \subseteq A^*$, there is a regular expression γ such that $\langle\gamma\rangle = L$.

- <u>The Pigeon-Hole Principle</u> : If $(n+1)$ objects are distributed among n sets, then <u>at least</u> one of the sets must contain <u>at least</u> two objects.

a basic combinatorial fact →

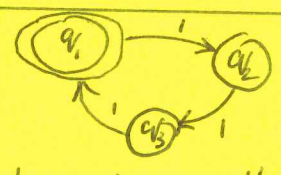
- <u>Pumping Lemma</u> :

> Consider a dfa $\mathcal{M}$ with n states. Let L be the language accepted by this dfa. If $x \in L$ and $|x| \geq n$, then it must be the case that we can write $x = uvw$ with $v \neq 0$ and that $uv^{[i]}w \in L$ for all $i = 0, 1, 2, 3, \ldots$

PROOF : Let x be a string with <u>at least</u> n symbols. $\mathcal{M}$ need n state transitions to scan x. $\mathcal{M}$ needs at least $n+1$ states (not necessarily distinct) to scan x. At least one state must get used at least twice. Let q be that state. Then it must be the case that we can write $x = uvw$ with $v \neq 0$ so that $\delta^*(q_1, u) = q$, $\delta^*(q, v) = q$, and $\delta^*(q, w) \in F$. Implies that $uv^{[m]}w \in L(\mathcal{M})$.

- THEOREM : Let $\mathcal{M}$ be a dfa with n states. If $L(\mathcal{M}) \neq \emptyset$, then there must exist a string $x \in L(\mathcal{M})$ such that $|x| < n$.

PROOF : The same as above but with the additional stipulation that x is the shortest string such that $|x| \geq n$. Obviously, then, when $i = 0$ in $uv^{[i]}w$ we have $|uw| < n$. (If $|uw| \geq n$ then you did <u>not</u> choose x that is the shortest string with $|x| \geq n$.)

- Does it seem like  violates the above theorem ?

It does NOT, but why ? clue : The double circle.

- The above theorem also shows that the problem of determining whether $L(\mathcal{M})$ is empty is solvable ? All we have to do is to run $\mathcal{M}$ on all strings x with $|x| < n$.

- THEOREM : Let $\mathcal{M}$ be a dfa with n states. Then $L(\mathcal{M})$ is infinite iff <u>$L(\mathcal{M})$ contains a string x such that $n \leq |x| < 2n$.</u>

Proof : Given an x with $n \leq |x| < 2n$, by the Pumping Lemma, L is infinite.
For the converse, let L be infinite. Let x be the shortest string with $|x| \geq 2n$. We write $x = x_1 x_2$ where $|x_1| = n$ and $|x_2| \geq n$. As in the proof of Pumping Lemma, $x_1 = uvw$ with $v \neq 0$ such that $\delta^*(q_1, uw) = \delta^*(q_1, x_1)$. Implying $uwx_2 \in L$. But $|uwx_2| < 2n$. If not, x was not shortest $\geq 2n$.