Regular Languages are in Nick's Class

We give an $\mathcal{NC}$ algorithm which decides the mambership problem for a regular language, proving that the class of regular languages is a subclass of Nick's Class.

Logical Matrices

A logical matrix is a matrix whose entries are of Boolean type. We write 1 for true and 0 for false. Matrix addition and multiplication is defined in the usual manner for logical matrices, except that disjunction replaces addition and conjunction replaces multiplication.

For example, $\begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}$

Transition Matrices

Let $L \subseteq \Sigma^*$ be a regular language over Σ, and let $M = (\Sigma, Q, F, q_0, \delta)$ be an NFA which accepts L. Let $Q = \{q_i : 0 \leq i < k\}$. For any $a \in \Sigma$ we define the *transition matrix* T_a to be the $k \times k$ logical matrix where

$$T_a[i,j] = \begin{cases} 1 \text{ if } q_j \in \delta(a, q_i) \\ 0 \text{ otherwise} \end{cases} \quad \text{for all } \ 0 \leq i, j < k$$

We extend this definition by mapping concatenation to matrix multiplication, *i.e.*, $T_{uv} = T_u T_v$, and T_λ is the identity matrix.

Algorithnm $\mathcal{A}$

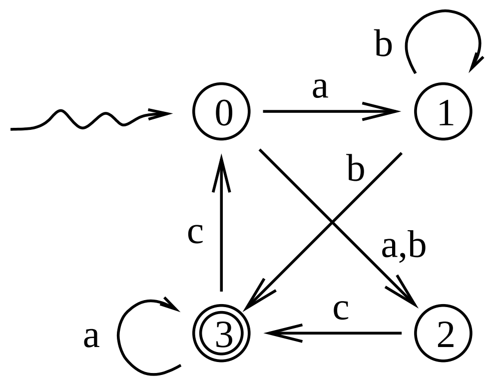
$\mathcal{A}$ reads a string $w \in \Sigma^*$ and returns 1 if and only if $w \in L$. Without loss of generality, $|w|$ is a power of 2. Let $m = \log_2 n$. For any $0 \leq p \leq m$, w is the concatenation of 2^{m-p} substrings of length 2^p. Let $\mathcal{S}$ be the set of all substrings thus obtained. $\mathcal{A}$ is as follows:

```
                Algorithm 𝒜
Compute the transition matrix T_a for all a ∈ Σ
For all p from 1 to m
  For all u ∈ 𝒮 of length 2^p
    Let u = xy where |x| = |y| = 2^(p−1)
    T_u = T_x T_y
If (T_w[0, f] for some f ∈ F) return 1
Else return 0
```

Since we are taking the sizes of Σ and Q to be $O(1)$, all T_a for $a \in \Sigma$ can be computed in $O(1)$ time by one processor. The remainder of the algorithm consists of $n-1$ matrix multiplications which can be done in $O(\log n)$ time with $O(n)$ processors. Thus $\mathcal{A} \in \mathcal{NC}$.

Example

Let $\Sigma = \{a, b, c\}$ and $L = L(M)$, where M is the followinhg NFA. Let $w = acacabba$.



We compute transition matrices of elementary strings, then copy to the 8 leaves of our computation tree. Each matrix in rows 2–4 is the product of the two above it. Then $w \in L$ since $T_w[0,3] = 1$ and $q_3 \in F$.

$$\underset{T_\lambda}{\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}} \quad \underset{T_a}{\begin{bmatrix}0&1&1&0\\0&0&0&0\\0&0&0&0\\0&0&0&1\end{bmatrix}} \quad \underset{T_b}{\begin{bmatrix}0&0&1&0\\0&1&0&1\\0&0&0&0\\0&0&0&0\end{bmatrix}} \quad \underset{T_c}{\begin{bmatrix}0&0&0&0\\0&0&0&0\\0&0&0&1\\1&0&0&0\end{bmatrix}}$$

$$\underset{T_a}{\begin{bmatrix}0110\\0000\\0000\\0001\end{bmatrix}} \underset{T_c}{\begin{bmatrix}0000\\0000\\0001\\1000\end{bmatrix}} \underset{T_a}{\begin{bmatrix}0110\\0000\\0000\\0001\end{bmatrix}} \underset{T_c}{\begin{bmatrix}0000\\0000\\0001\\1000\end{bmatrix}} \underset{T_a}{\begin{bmatrix}0110\\0000\\0000\\0001\end{bmatrix}} \underset{T_b}{\begin{bmatrix}0010\\0101\\0000\\0000\end{bmatrix}} \underset{T_b}{\begin{bmatrix}0010\\0101\\0000\\0000\end{bmatrix}} \underset{T_a}{\begin{bmatrix}0110\\0000\\0000\\0001\end{bmatrix}}$$

$$\underset{T_{ac}}{\begin{bmatrix}0001\\0000\\0000\\1000\end{bmatrix}} \quad \underset{T_{ac}}{\begin{bmatrix}0001\\0000\\0000\\1000\end{bmatrix}} \quad \underset{T_{ab}}{\begin{bmatrix}0101\\0000\\0000\\0000\end{bmatrix}} \quad \underset{T_{ba}}{\begin{bmatrix}0000\\0001\\0000\\0000\end{bmatrix}}$$

$$\underset{T_{acac}}{\begin{bmatrix}1000\\0000\\0000\\0001\end{bmatrix}} \qquad \underset{T_{abba}}{\begin{bmatrix}0001\\0000\\0000\\0000\end{bmatrix}}$$

$$\underset{T_{acacabba} = T_w}{\begin{bmatrix}0001\\0000\\0000\\0000\end{bmatrix}}$$