Regular Languages are $\mathcal{NC}$

Let $L$ be a regular language, and let $M$ be a DFA which accepts (actually, decides) $L$. Using $M$, we design an $\mathcal{NC}$ algorithm which decides $L$ in $O(\log n)$ time using $O(n)$ processors, where $n$ is the length of the input string $w$.

$M = (Q, \Sigma, \delta, q_0, F)$. Recall $Q$ is the set of states of $M$, $\Sigma$ is the input alphabet, $\delta : Q \times \Sigma \to Q$ is the transition function, $q_0 \in Q$ is the start state, and $F \subseteq Q$ is the set of final states. We extend the transition function to $\delta^* : Q \times \Sigma^* \to Q$ inductively: $\delta^*(q, \lambda) = q$, and $\delta^*(q, aw) = \delta^*(\delta(q, a), w)$ for any $a \in \Sigma$, $q \in Q$. If $w \in \Sigma^*$, then $w \in L$, i.e., is accepted by $M$, if $\delta^*(q_0, w) \in F$. Equivalently, we describe the transition function of $M$ by a function $\delta^*(\ , x) : Q \to Q$, for any $x \in \Sigma^*$; where $\delta^*(\ , x)(q) = \delta^*(q, x)$ for all $q \in Q$.

We now describe an $\mathcal{NC}$ algorithm $A$, which decides whether a given string is a member of $L$. To simplify our construction, we assume that the length of the input string is a power of 2, although it is a simple matter to generalize to arbitrary $n$: augment $\Sigma$ with a special “do nothing” symbol •, which we call a blank. Define $\delta(q, •) = q$ for any $q \in Q$. Let $w^*$ be the string obtained by padding the input string $w$ with just enough blanks to bring its length to a power of 2. For example, if $w = aabcababbcab$ we let $w^* = aabcababbcabca\cdots$. Let $n = 2^m = |w^*|$. Let $\mathcal{G}$ be the set of consisting of all subintervals obtained by breaking $w^*$ into $2^i$ pieces each of length $2^{m-i}$, for all $0 \leq i \leq m$. Thus $\mathcal{G}$ consists of all subintervals of length 1, $n/2$ subintervals of length 2, $n/4$ subintervals of length 4, and so forth; these will include 2 subintervals of length $n/2$ and one of length $n$, namely $w$ itself. The cardinality of $\mathcal{G}$ is $2n - 1$. Each member of $\mathcal{G}$ of length $2^i$, for $i > 0$, is the concatenation of two members of $\mathcal{G}$ of length $2^{i-1}$. We let $u_{i,j}$ be the $j^{th}$ member of $\mathcal{G}$ of length $2^i$, for $0 \leq i \leq m$ and $1 \leq j \leq 2^{m-i}$. That is, $u_{i,j}$ is the substring of $w^*$ of length $2^i$ ending at the $(2^i j)^{th}$ place of $w^*$. $A$ has $1 + m$ phases, which we number 0, 1, . . . , $m$. Phase $i$ of $A$ computes $\delta^*(\ , u_{i,j})$ for all $1 \leq j \leq 2^i$, takes $O(1)$ time and uses $2^{m-i}$ processors. The functions $\delta^*(\ , u_{i,j})$ for all $j$ can simply be read off the state diagram of $M$. For $i > 0$, $\delta^*(\ , u_{i,j})$ is simply the composition of the functions $\delta^*(\ , u_{i-1,2j-1})$ and $\delta^*(\ , u_{i-1,2j})$, for all $1 \leq j \leq 2^{m-i}$. For example, in Phase 1 of the example computation below, $\delta^*(\ , bc)$ is the composition of $\delta^*(\ , b)$ with $\delta^*(\ , c)$.
Example

Let \( M \) be given by the state diagram below. For simplicitly, we dispense with the clumsy “\( q_i \)” notation and write simply \( i \). Thus \( Q = \{0, 1, 2\} \), the start state is 0, and \( F = \{2\} \).

Let \( w = aabcabceca \). The sequential computation of \( M \) with input \( w \) takes 13 steps. Since \( 2 \in F \), \( w \) is accepted.

\[
0 \xrightarrow{a} 0 \xrightarrow{a} 0 \xrightarrow{b} 1 \xrightarrow{c} 1 \xrightarrow{a} 2 \xrightarrow{c} 1 \xrightarrow{b} 0 \xrightarrow{a} 0 \xrightarrow{b} 1 \xrightarrow{c} 1 \xrightarrow{c} 1 \xrightarrow{c} 1 \xrightarrow{a} 2
\]

Padding with blanks to obtain a length of 16, a power of 2, we let \( w^* = aabcabceca \cdots \). Execute \( A \) in five phases using 16 processors.

Phase 0:

\[
\begin{array}{cccccccccccccc}
\text{a} & \text{a} & \text{b} & \text{c} & \text{a} & \text{c} & \text{b} & \text{a} & \text{b} & \text{c} & \text{a} & \text{c} & \text{a} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 2 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{array}
\]

Phase 1:

\[
\begin{array}{cccccccccccccc}
\text{aa} & \text{bc} & \text{ac} & \text{ba} & \text{bc} & \text{cc} & \text{a} & \text{c} & \text{a} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 2 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{array}
\]

Phase 2:

\[
\begin{array}{cccccccccccccc}
\text{aabc} & \text{acba} & \text{becc} & \text{a} & \text{c} & \text{a} \\
0 & 0 & 0 & 0 & 0 & 0 \\
1 & 2 & 1 & 1 & 1 & 1
\end{array}
\]

Phase 3:

\[
\begin{array}{cccccccccccccc}
\text{aabcabca} & \text{becca} & \text{a} \\
0 & 0 & 0 \\
1 & 2 & 1
\end{array}
\]

Phase 4:

\[
\begin{array}{cccccccccccccc}
\text{aabcabceca} & \text{a} \\
0 & 0 & 0 \\
1 & 2 & 1
\end{array}
\]

The computation at Phase 4 tells us that \( \delta^*(0, aabcabceca \cdots) = 2 \), a final state. Thus \( aabcabceca \in L \).