\section*{$\mathcal{NC}$ and $\mathcal{P}$-Completeness}

\subsection*{Nick’s Class}

$\mathcal{NC}$, or Nick’s Class, is named after Nick Pippenger, currently on the faculty of Harvey Mudd College. A language is $\mathcal{NC}$ if its membership problem can be solved by a parallel program using polynomially many processors in polylogarithmic time.

Many of the problems that you are familiar with are in the class $\mathcal{NC}$. For example, the 0/1 version of the shortest path problem is in $\mathcal{NC}$, and every context-free language is in the class $\mathcal{NC}$. Whether $\mathcal{NC} = \mathcal{P}$ is an open question of enormous importance.

We say that a $\mathcal{P}$-time language (problem) is $\mathcal{P}$-complete if every $\mathcal{P}$-time language can be reduced to it by a function which can be computed in polylogarithmic time by polynomially many processors.

\section*{The Circuit Value Problem, or the Boolean Circuit Problem}

We now give a $\mathcal{P}$-complete problem, namely the circuit value problem. An instance of the Circuit Value Problem is a sequence of $n$ Boolean assignments.

1. The left side of the $i$th assignment is the Boolean variable $x_i$.
2. The right side of the $i$th assignment is one of the following.
   
   \begin{enumerate}
   \item 0 (false)
   \item 1 (true)
   \item $x_j$ for $j < i$
   \item $!x_j$ for $j < i$ (! means ‘not’)
   \item $x_j + x_k$ for $j < i$ and $k < i$ (+ means ‘or’)
   \item $x_j \cdot x_k$ for $j < i$ and $k < i$ (* means ‘and’)
   \end{enumerate}

3. The answer to an instance of CVP is the value of $x_n$.

Trivially, CVP is in $\mathcal{P}$. Simply execute the $n$ statements in order. In fact, the CVP is a Dynamic Programming problem. It is known that CVP is $\mathcal{P}$-complete, which implies that if it is in Nick’s Class, then $\mathcal{NC} = \mathcal{P}$. Can we reduce every DP problem to the CVP, using a Nick’s Class reduction?

\section*{Boolean Dynamic Programming}

We define a Boolean dynamic program $\mathcal{P}$ to be a dynamic program for which the answer to every subproblem is a Boolean value. The \emph{value} of $\mathcal{P}$ is defined to be the value of $x_n$, the last subproblem. Let $x_i$ be the value of the $i$th subproblem of a Boolean dynamic program. Then there is a Boolean function of several variables $F_i$ such that $x_i = F_i(x_1, x_2, \ldots x_{i-1})$. (Note that $F_1$ is a function of no variables, hence is either constant 0 or constant 1.)

We define the \emph{reachback} of a Boolean dynamic program to be the maximum integer $d$ such that, for all $i$, $x_i$ depends only on $\{x_j : j \geq i - d\}$.\footnote{The reachback is an upper bound on the \emph{width} of $\mathcal{P}$, as defined in the file dpNC01.}

We define a \emph{Boolean Dynamic Programming Language} to be any set of Boolean dynamic programs whose values are all \emph{true}. For example, CVP is a Boolean dynamic programming language.
Theorem 1 Given constants $K, K_2$, let $DP[K, K_2]$ be the set of all Boolean dynamic programming problems whose values are true and whose reachback is bounded by $K \log_2 n + K_2$, where $n$ is the length of the problem. Then $DP[K, K_2]$ is $NC$.

Theorem 2 The class of regular languages is $NC$.

Proof: Let $L$ be a regular language. Without loss of generality, $L$ is over the Boolean alphabet $\Sigma = \{0, 1\}$. Let $M$ be a DFA which decides $L$, and whose states are $Q = \{q_0, q_1, \ldots, q_{m-1}\}$.

We define a reduction $R$ from $L$ to $DP[0, 2m]$. For $w \in \Sigma^*$ let $\ell = |w|$. $R(w)$ is the Boolean dynamic program $P$ whose subproblems are $x_i$ for $0 \leq i \leq n = \ell m$. We now define each subproblem. If $i < n$, let $k = i \% m$ and let $t = i/m$, integer truncated division. Then $x_i$ is true if and only if the computation of $M$ after $t$ steps is in state $q_k$. We define $x_n$ to be true if and only if the computation is in a final state after $\ell$ steps. Clearly $x_0$ is true, and the reachback of $P$ cannot exceed $2m$, since the state of $M$ at step $t$ can be computed from the state of $M$ at step $t - 1$, and $x_n$ is true if and only if $w \in L$. $R$ can be computed by $\ell$ processors in $O(m^2)$ time, hence the reduction is $NC$. □