\(N\mathcal{C}\) and \(\mathcal{P}\)-Completeness

Nick’s Class

\(N\mathcal{C}\), or Nick’s Class, is named after Nick Pippenger, currently on the faculty of Harvey Mudd College. A language is \(N\mathcal{C}\) 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 \(N\mathcal{C}\). For example, the 0/1 version of the shortest path problem is in \(N\mathcal{C}\), and every context-free language is in the class \(N\mathcal{C}\). Whether \(N\mathcal{C} = \mathcal{P}\) 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.

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 \(!x_i\) for \(j < i\) (! means ‘not’)
   \item \(x_j + x_k\) for \(j < i\) and \(k < i\) (+ means ‘or’)
   \item \(x_j \ast x_k\) for \(j < i\) and \(k < i\) (\(\ast\) means ‘and’)
   \end{enumerate}

3. The answer to an instance of the 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 \(N\mathcal{C} = \mathcal{P}\). Can we reduce every DP problem to the CVP, using a Nick’s Class reduction?

Problems

1. Given an alphabet \(\Sigma\), an instance of the symbol find problem over \(\Sigma\) consists of a string \(w\) over \(\Sigma\) together with a single symbol \(x \in \Sigma\). The instance is true if and only if some symbol of the string \(w\) is equal to \(x\). Prove that the symbol find problem over \(\Sigma\) is in Nick’s Class.
2. Prove that every regular language is in Nick’s Class.

Dynamic Programming

The CVP is clearly a dynamic programming problem. Can every dynamic programming problem be reduced to the CVP by an \(N\mathcal{C}\) function?

To clear our thoughts, we give a formal definition of a DP problem. We have a directed acyclic graph \((G, E)\). Each vertex \(x\) of \(G\) has a label consisting of a variable \(v_x\) as well as a polynomial time algorithm \(A_x\) which computes \(v_x\) from \(\{v_y : (y, x) \in E\}\).