Models of parallel computations

Sequential models such as the Turing machine or the RAM execute one transition or instruction per unit of time. In the models we introduce in this chapter, a polynomial number of transitions are followed per unit of time. We consider two models: boolean circuits and the PRAM (Parallel Random Access Machine), although there are many other possible models. Both circuits and PRAM assume synchronized elements which realize concurrent operations. Other models do not make this assumption and deal with distributed components.

In the case of circuits, the basic elements are gates and the elementary operations are the logical operations $\neg, \lor, \land$. In the case of PRAM, the basic elements are RAM processors and the elementary instructions are the those used by RAMs working in parallel while sharing a common memory.

We show that these two models are equivalent when the PRAM use a polynomial number of processors and the circuits use a notion of uniform computability. Parallel algorithms are particularly interesting when the time complexity is of the order $(\log n)^i$. The main parallel complexity classes are NC$^i$ and AC$^i$ which play an important role for both algorithmics and logic. A problem is NC$^i$-computable if it can be solved by a uniform family of circuits of polynomial size and depth $(\log n)^i$, i.e. of parallel time $(\log n)^i$. The class NC is the union of the classes NC$^i$.

In the first section we study the boolean circuits and in the second section we introduce the main parallel complexity classes. Some NC algorithms such as matrix multiplication and matrix inversion, which uses arithmetic circuits with addition, multiplication and division, are presented in the third section. The PRAM are studied in the fourth section as well as the equivalence with the circuits.

12.1. The boolean circuits

Let $\Omega = \{\neg, \lor, \land\}$ the set of boolean operators. In Chapter 1 we showed that a propositional formula $\psi(x_1, \ldots, x_n)$ built from the boolean variables $x_1, \ldots, x_n$ admits a binary tree decomposition where the leaves are labelled by the propositional variables $x_i$ and the nodes are labelled by the binary operators $\{\lor, \land\}$ or the unary operator $\{\neg\}$. A binary word $x$ can be considered as a valuation of the input variables $x_i \in \{0, 1\}$ and the binary tree represents the computations of a boolean circuit which realizes parallel computations because each node realizes a specific
elementary operation.

Classical models of computation compute inputs of arbitrary size. In the case of circuits, we consider families of circuits

\[ C = \{C_1, C_2, \ldots, C_n, \ldots\} \]

The circuit \( C_n \) deals with all inputs of size \( n \). Recall that the in-degree of a node \( v \) of a graph \( G \) is the number of edges \((u, v)\) adjacent to \( v \) and the out-degree of \( v \) is the number of edges \((v, w)\) adjacent to \( v \).

**Definition 12.1.** A boolean circuit \( C_n \) is a directed acyclic graph, where nodes of positive in-degree are labelled with elements of \( \Omega \), nodes of in-degree 0 are labelled by variables \( x_i \) where \( i = 1, \ldots, n \), called input variables. The nodes of out-degree 0 are labelled with variables \( y_1, y_2, \ldots, y_m \), called output variables. The in-degree is 2 for nodes labelled by \( \wedge \) or \( \lor \) and 1 for nodes labelled by \( \neg \).

We write, \( C_n^i(x_1, x_2, \ldots, x_n) = y_i \), to denote the function which on the input \( x = x_1, \ldots, x_n \) gives \( y_i \), the \( i \)th output of the circuit. With a decision problem we associate a family of circuits with a single output \( y_1 \) and interpret \( y_1 = 1 \) as accepting \( x \) and \( y_1 = 0 \) as rejecting \( x \). In a search problem, we associate a family of circuits which define a family of functions \( f(x) = y \) where \( y = y_1, \ldots, y_m \) if the circuit \( C_{|x|} \) has exactly \( m \) output variables.

**Example.** Consider the formulas \( \psi(x_1, x_2) \) and \( \psi'(x_1, x_2, x_3) \):

\[
((x_1 \land x_2) \lor (x_1 \lor x_2))
\]

\[
(x_1 \lor x_2) \land x_3
\]

With each these formulas we can associate a circuit with three inputs and one output. The pair \( (\psi(x_1, x_2), \psi'(x_1, x_2, x_3)) \) of formula describes the circuit of Figure 12.1 with three inputs and two outputs.

![Figure 12.1. A circuit with \( x_1, x_2, x_3 \) as inputs and \( y_1, y_2 \) as output on \( \{\land, \lor\} \).](image)

The main parameters of a circuit \( C \) are the depth and the size.