A Uniform Universal CREW PRAM

Bruno Martin *
LIP-IMAG, URA CNRS n° 1398
Ecole Normale Supérieure de Lyon,
46 Allée d'Italie, 69364 Lyon Cedex 07, France.

Abstract. The universality of the Parallel Random-Access Machines is usually defined by simulating universal Turing machines or boolean networks. These definitions are well-suited if we are interested in evaluating the complexity of algorithms but it is not as good if we want to deal with computability. We propose in this paper another definition for the universality of the Parallel Random-Access Machines based on cellular automata and we discuss the advantages and the drawbacks of this simulation. We prove that there exists a Concurrent-Read Exclusive-Write Parallel Random-Access Machine which is capable of simulating any given cellular automaton in constant time. We then derive to the definition of complexity classes for the Parallel Random-Access Machines and for cellular automata.

Introduction

The field of parallel computation is going through a period of unrest. While most theoretical computer science is busy designing and evaluating algorithms on Parallel Random-Access Machines, one of the first problem to be solved is maybe the universality for the models of parallel computation. This kind of problem seems to be quite new in this area of the research in theoretical computer science devoted to parallelism. Very few papers speak about the computational power of the parallel machines. They are often known to be universal by simulating the most classical models of computation such as Turing machines or boolean networks. These two models have both drawbacks when applied to parallel machines. The first one implies a sequentialization of the parallel machine and the second one introduces the difficult notion of uniformity. Let us briefly recall the notion of uniformity; a circuit is defined for bounded inputs. So, to speak of problem resolution (that is, computations are defined for any size of the inputs) we need the definition of the circuits capable of solving a problem for any input size. We get then a family of circuits \{C_n\} for solving a given problem. A common definition of the uniformity is the logspace uniformity which means that the description of the \(n^{th}\) circuit \(C_n\) can be generated by a Turing machine using \(O(\log n)\) workspace. Then, the complexity of a problem solvable by circuits must use the definition of the uniformity.

In the present paper, we introduce a proper uniform intrinsically parallel model of computation, the cellular automata model known to be near the single program

* This work was supported by Esprit Basic Research Action "Algebraic and Syntactic Methods In Computer Science" and by the Programme de Recherches Coordonnées Mathématiques-Informatique
many data models of parallel machines. We will show that there exists a concurrent-read exclusive-write parallel random-access machine which simulates in constant-time and with little space any given unidimensional cellular automaton. Thus, we prove directly the equivalence between the PRAM model and the cellular automata model and precise in which sense the two models can be compared.

From this point, it is possible to apply well-known results of the theory of computability to find out complexity measures and complexity classes relative to these models. We will discuss the PLINEAR class, which seems well-suited for the cellular automata model.

1 Definitions

1.1 Parallel Random-Access Machines

The concept of *Parallel Random-Access Machine* (PRAM for short) has been introduced to include the processing of huge data. In this model, the time is proved equivalent, within a polynomial, to the space of the Turing Machine [2]. This machine is capable to wake up a large (possibly infinite) number of processors, all operating both on a private memory and on a global shared memory. However, this machine is not restricted to execute the same instruction in all active processors at each unit of time. For this model, a memory word can have a constant size, be logarithmic in the size of the used memory, or be unlimited. A *uniform cost* Parallel Random-Access Machine is a PRAM for which every instruction takes a unit of time and in a *logarithmic-cost* one, instruction time is the sum of the sizes of the values involved and the size of the address of the operand, all in bits. More formally, a Parallel Random-Access Machine consists in an unbounded set of processors $p_0, p_1, p_2 \ldots$ and an unbounded global memory $x_0, x_1, x_2, \ldots$ and a finite program. Each processor $p_i$ has an unbounded local memory $y_0, y_1, y_2, \ldots$. Register $y_0$ is called the accumulator of the processor. Each processor has a program counter and a flag indicating whether the processor is running or not. A program consists of possibly labeled instructions chosen from the following list:

\[
\begin{align*}
y_i &:= \text{constant;} \\
y_i &:= y_i + y_k; \\
y_i &:= \lfloor y_i/2 \rfloor; \\
y_i &:= y_{y_i}; \\
\text{accept;} \\
\text{goto } m \text{ if } y_i > 0; \\
y_i &:= x_{y_i}; \\
x_{y_i} &:= y_i; \\
\text{fork } m;
\end{align*}
\]

The parallelism is achieved by the "fork $m$" instruction. When a processor $p_i$ executes a fork instruction, it selects the first inactive processor $p_j$, clears the local memory of $p_j$ and fetches the accumulator of $p_i$ onto the accumulator of $p_j$. Then, $p_j$ starts its computation in the label $m$ of the program. The communications between