CELLULAR AUTOMATA AND SPATIAL MACHINES

BRUNO MARTIN

Definitions

We first recall briefly the definition of cellular automata we use in the sequel. A *cellular automaton* is a bi-infinite array of identical cells indexed by $\mathbb{Z}$, the set of integers. Each cell is a finite state machine $C = (Q, \delta)$ where Q denotes a finite set of states and δ the local transition function, a mapping $\delta : Q^3 \to Q$ which gives the next state of the cells in parallel.

The model of spatial machines has been introduced in [4] in order to modelize more properly realistic parallel machines. According to [4], spatial machines capture some basic assumptions as to physical computing machines: some minimal physical volume is required for any entity which participates in the computation, such as a memory bit, a processor and a message bit travelling through space, and a message bit cannot travel faster than the speed of light. The model is supposed to be more realistic than a general cellular automaton since the number of processors is limited to be finite although they can move and send messages in an infinite space.

Informally, a spatial machine is a finite set of finite processors moving and exchanging messages in 3 dimensional euclidian space. Each processor is like a Turing machine's unit of control where the tape has been supressed. They communicate through sequences of bits moving in the 3 dimensional space. Several distinctions about the model have been made according to the number of processors and their mobility. Let us denote by *stationnary* a spatial machine whose processors are all required to be immobile and by *sequential* a two processors spatial machine, one mobile and one immobile (see [3, 4] for further details). It has been proved in [4] that stationnary spatial machines are not more powerful than finite automata and that sequential spatial machines are as powerful as Turing machines.

1. Simulations of cellular automata by spatial machines

In [10] we proved:

Theorem 1. *There exists a sequential spatial machine capable to simulate any k-states totalistic cellular automaton. Furthermore, if the computation-time of the cellular automaton is $T(n)$, the corresponding computation-time of the spatial machine will be $(T^2(n) + nT(n) + O(1))(\lceil \log_2 k \rceil + O(1))$.*

This result corresponds to a sequential simulation of a given totalistic[1] cellular automaton. We propose an alternative parallel simulation of cellular automata which improves the time-complexity.

Theorem 2. *There exists a spatial machine capable to simulate any k-states totalistic cellular automaton. Furthermore, if the computation-time of the cellular automaton is $T(n)$, the corresponding computation-time of the spatial machine will be $T(n)O(k\lceil \log_2 k \rceil)$.*

One might think that this time could be easily decreased to $T(n)O(\lceil \log_2 k \rceil)$ but it is not the case since in our simulation we do not encode a cellular automaton's transition function as the transition function of the spatial machine's processors. This simulation permits then to simulate directly any cellular automaton on a spatial machine given the description of the cellular automaton and its input.

2. Applications to universality

Since cellular automata are computation universal [1, 9], we can deduce the universality of spatial machines from the previous simulations. Let us make the following distinctions on the way universality is obtained. First remind that a universal machine U is *computation-universal* if it can compute any partial recursive function. This formal definition authorizes us several ways to prove the universality of a machine model.

The first one is the *universality by simulation* where machine U from model $\mathcal{M}$ can simulate directly the behavior of any given machine A from model $\mathcal{M}'$ on any input. U receives as an input the program of machine A, a reasonable encoding of the input x of A and simulates each

[1] See [1] for a definition of totalistic cellular automata

step of the computation of A on input x. Consider for instance the constant-time simulation of a cellular automaton by a PRAM [8] or by a (parallel) spatial machine. If both U and A are in the same model, we call this kind of universality *intrinsic* (see [1, 9] for two intrinsic universal cellular automata).

The second is the *universality by hereditary*. In this case, machine U from model $\mathcal{M}$ simulates a universal machine U' from model $\mathcal{M}'$ with U' a machine universal by simulation. This is the way universality of a model is usually carried out. Consider for example the simulation of a universal Turing machine by a spatial machine [4], the simulation of a universal cellular automaton by a sequential spatial machine or the simulations of a universal Turing machine by cellular automata [5, 7, 11].

The last one is the *universality by construction* where given the description of machine A from model $\mathcal{M}$ and its input x, a *constructor* C from model $\mathcal{M}$ constructs explicitely this machine. This is the way universality of von Neumann machine is proved (see [2] for further explanation).

3. Decidability and complexity results

The simulation of a Turing machine by a sequential spatial machine provides easily some decidability and complexity results.

The following problem is undecidable :

INSTANCE : Let S be a spatial machine, P a processor and q a state of P.

QUESTION : Does state q appear in the computation of S?

It sufficies to prove the result for a sequential spatial machine since any p processors spatial machine can be simulated by a sequential one [3] and by reduction from the halting problem for Turing machines.

The *Generic Machine Simulation Problem* for spatial machines is P-complete, which is an easy corollary of GMSP for Turing machines. The proof is similar to the one of the P-completeness of GMSP for cellular automata referred as problem A.12.3 in [6].

4. Conclusion

As a consequence of our simulations, we can construct two universal spatial machines (one by simulation and the other by hereditary). Furthermore, the spatial machines are restricted to the framework of Turing-computability since they do not form a massively parallel model of computation but they seem to modelise more properly some features of feasible parallel computers.

References

[1] J. Albert and K. Culik II. A simple universal cellular automaton and its one-way and totalistic version. *Complex Systems*, 1:1–16, 1987.

[2] A. Arbib. From universal Turing machines to self-reproduction in *The universal Turing machine*, pages 177–189. Oxford Science Publications, 1988.

[3] Y. Feldman and E. Shapiro. Spatial machines: Towards a more realistic approach to parallel computation. Technical Report CS88-05, The Weizmann Institute of Science, May 1988.

[4] Y. Feldman and E. Shapiro. Spatial machines, a more realistic approach to parallel computation. *Comm. of the ACM*, 35(10):61–73, 1992.

[5] D. Gordon. On the computational power of totalistic cellular automata. *Math. System Theory*, 20:43–52, 1987.

[6] R. Greenlaw, H.J. Hoover, and W.L. Ruzzo. *Limits to parallel computation*. Oxford University Press, 1995.

[7] K. Lindgren and M. G. Nordhal. Universal computation in simple one-dimensional cellular automata. *Complex Systems*, 4:299–318, 1990.

[8] B. Martin. Efficient unidimensional universal cellular automaton. In *MFCS'92*, number 629 in Lecture Notes in Computer Science, pages 374–382. Springer Verlag, 1992.

[9] B. Martin. A universal cellular automaton in quasi-linear time and its S-m-n form. *Theoretical Computer Science*, 123:199–237, 1994.

[10] B. Martin. Machines spatiales universelles. *Technique et Science Informatiques*, 14(5):551–566, 1995.

[11] A.R. Smith III. Simple computation-universal cellular spaces. *J. ACM*, 18:339–353, 1971.

(Bruno Martin) Laboratoire d'informatique, Signaux et Systèmes de Sophia-Antipolis, I3S-C.N.R.S. - U.R.A. 1376, 650 Route des Colles, B.P. 145, F-06903 SOPHIA-ANTIPOLIS Cedex

E-mail address: bmartin@unice.fr