Theoretical Computer Science ••• (••••) •••–•••

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Computational complexity of finite asynchronous cellular automata

Alberto Dennunzio [a], Enrico Formenti [b], Luca Manzoni [a], Giancarlo Mauri [a,*], Antonio E. Porreca [a]

[a] Dipartimento di Informatica, Sistemistica e Comunicazione, Università degli Studi di Milano-Bicocca, Viale Sarca 336/14, 20126 Milano, Italy
[b] Université Nice Sophia Antipolis, CNRS, I3S, UMR 7271, 06900 Sophia Antipolis, France

ARTICLE INFO

Article history:
Received 8 March 2015
Received in revised form 12 November 2015
Accepted 4 December 2015
Available online xxxx

Keywords:
Natural computing
Asynchronous cellular automata
Computational complexity

ABSTRACT

Cellular Automata (CA) are a well-established bio-inspired model of computation that has been successfully applied in several domains. In the recent years the importance of modelling real systems more accurately has sparkled a new interest in the study of asynchronous CA (ACA). When using an ACA for modelling real systems, it is important to determine the fidelity of the model, in particular with regards to the existence (or absence) of certain dynamical behaviors.

This paper is concerned with two big classes of problems: reachability and preimage existence. For each class, both an existential and a universal version are considered. The following results are proved. Reachability is **PSPACE**-complete, its resource bounded version is **NP**-complete (existential form) or **coNP**-complete (universal form). The preimage problem is dimension sensitive in the sense that it is **NL**-complete (both existential and universal form) for one-dimensional ACA while it is **NP**-complete (existential version) or $\Pi_2^{\mathbf{P}}$-complete (universal version) for higher dimension.

1. Introduction

Cellular Automata (CA) are a well-known and widely used formal model, consisting of a set of state automata arranged on a finite or infinite lattice. A configuration is a snapshot of all the states of the automata. Each automaton updates its state by using a local rule that depends only on the states of the automaton and of its neighbours. The local rule is uniform, i.e., the same for all the automata, and it is applied synchronously, that is, all the automata update their state at the same time. We refer the reader to [15,9,1,8,7,2] for an up-to-date overview and recent results on classical CA.

The interest in studying asynchronous CA (ACA) has recently increased because of the necessity of modelling real-life systems. Indeed, in nature few processes are really synchronous, and asynchrony is the common situation. Inspired by the necessity of modelling such processes, many different variations of the classical CA have been defined. We recall, for example, α-asynchronous CA [12], in which a cell is updated at each step with a probability α, fully asynchronous CA [11,17] (fully-ACA) in which only one cell is updated at each time step, and mACA, an "umbrella" model, that captures the behaviours of many other asynchronous CA models [6,5].

* Corresponding author.
E-mail addresses: dennunzio@disco.unimib.it (A. Dennunzio), formenti@unice.fr (E. Formenti), luca.manzoni@disco.unimib.it (L. Manzoni), mauri@disco.unimib.it (G. Mauri), porreca@disco.unimib.it (A.E. Porreca).

http://dx.doi.org/10.1016/j.tcs.2015.12.003
0304-3975/

Cellular automata have been widely used as formal models of complex systems because of their wide variety of dynamical behaviours. Most of these dynamical behaviours are undecidable in the general case [10,16,3]. However, in the case of finite configurations, they turn out to be decidable. It is therefore interesting to understand from a computationally complexity point of view how hard is the decision problem for a given dynamical behavior. In [20], Sutner classifies the complexity of most of these behaviours. In particular, he shows that the reachability problem is **PSPACE**-complete and the existence of a preimage is **NL**-complete for one-dimensional CA while it is **NP**-complete for higher dimensions.

This paper follows the same trend for both the ACA and the fully ACA models. In any asynchronous setting, of course, one has to cope with the cell updating sequence of the automaton, which strongly interacts with the dynamical behaviour and might influence the computational complexity. Indeed, for any decision problem, two versions are considered: a universal form (i.e., the property is considered w.r.t. to all possible updates) and an existential one (i.e., the property is satisfied by at least one update sequence).

The main problem addressed is reachability, i.e., to establish if a configuration y is in the orbit of another configuration x. Similarly to what happens for classical CA, the reachability problem for ACA is **PSPACE**-complete. However, when a restriction is assumed on the number of time steps in the automaton evolution to reach a configuration y from a configuration x, there is a difference between the synchronous and asynchronous cases. In the former, the problem is **P**-complete, while in the latter it is either **NP** (existential form) or **coNP**-complete (universal form). These results highlight the fact that the order in which the cells are updated can be used to "extract" non-determinism.

Our results rely on a general simulation of non-deterministic Turing machine (TM) inspired by the simulation of a deterministic TM presented in [4]. While the idea of using asynchronous CA to simulate other systems – even other CA – is not new (see, for example [21]), here the asynchrony is not an additional challenge that needs to be solved, but an essential feature of the simulation that is used to perform non-deterministic choices.

The other main class of problems addressed in this paper is linked to the preimage existence. Remark that this problem has additional challenges for ACA and fully-ACA. Indeed, a preimage may exist for a certain choice of cells to update but not for another. The universal and the existential form of the preimage existence problem for one-dimensional fully ACA are in **L**, while the corresponding ones for fully ACA are **NL**-complete.

The same problems turn out to be intractable for ACA in dimension two or higher: the existence of an update sequence for which there exists a preimage is an **NP**-complete problem, while asking whether a preimage exists for all updates is $\mathbf{coNP}^{\mathbf{NP}}$-complete. The last case is particularly interesting since it shows an increase in complexity with respect to the synchronous case, which is **NP**-complete [20].

The paper is structured as follows: Section 2 recalls the necessary notions on CA, ACA, fully-ACA, and Turing machines. Section 3 contains a way to simulate a non-deterministic Turing machine with only a polynomial slowdown by means of ACA and fully-ACA. The complexity of the reachability and reachability in polynomial time problems are then studied and the obtained results are compared with the ones known for classical CA. The preimage problems are the core of Section 4. Finally, a brief summary of the paper and some directions for future research are given in Section 5.

2. Basic notions

In this section we briefly recall the basic notions on CA, ACA, and fully ACA.

For $a, b \in \mathbb{N}$ with $a \leq b$, denote by $[a, b]$ the set of integers $\{a, a+1, \ldots, b\}$. Let A be a finite alphabet. A (finite) *CA configuration* of length $n \in \mathbb{N}$ is a function $c : [1, n] \to A$, i.e., $c \in A^n$. For any configuration c and any position (*cell*) $i \in [1, n]$ we denote by c_i the element $c(i)$ in that position.

Definition 1. A *one-dimensional CA* is a triple $\mathcal{C} = (A, r, \lambda)$, where A is a finite *alphabet*, $r \in \mathbb{N}$ is the *radius*, and $\lambda : A^{2r+1} \to A$ is the *local rule* of the CA.

A finite CA of size $n \in \mathbb{N}$ is a CA with configurations of length n. The local rule is applied synchronously to all positions of any configuration $c \in A^n$, giving rise to a *global function* $\Lambda : A^n \to A^n$ defined as follows:

$$\forall i \in [1, n], \qquad \Lambda(c)_i = \lambda(c_{i-r}, \ldots, c_i, \ldots, c_{i+r})$$

where *periodic boundary conditions* are used, i.e., $c_j = c_{(j \mod n)+1}$ for positions $j \notin [1, n]$. In this paper, we only deal with finite ACA and fully ACA with periodic boundary conditions. However, all the results also hold in the case of fixed boundary conditions.

The definition of CA can be extended to $d \in \mathbb{N}$ dimensions.

Definition 2. Let $d \in \mathbb{N}$. A d-dimensional CA is a triple $\mathcal{C} = (A, r, \lambda)$, where A is a finite alphabet, $r \in \mathbb{N}$ is the radius, and $\lambda : A^{(2r+1)^d} \to A$ is the local rule of the CA.

The definition of CA dynamics and global function immediately generalize to the any dimension.

We are now ready to define ACA and fully-ACA. A finite and instantiated *Asynchronous CA* (ACA) of size n (in any dimension) is a quadruple $(A, r, \lambda, \upsilon)$ where the first three elements define a CA and $\upsilon = (\upsilon_t)_{t \in \mathbb{N} \setminus \{0\}}$ with $\upsilon_t \subseteq [1, n]$ is called

Please cite this article in press as: A. Dennunzio et al., Computational complexity of finite asynchronous cellular automata, Theoret. Comput. Sci. (2016), http://dx.doi.org/10.1016/j.tcs.2015.12.003