Theoretical Computer Science 573 (2015) 16–25

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Dichotomy results for fixed point counting in boolean dynamical systems ☆

Christopher M. Homan [a], Sven Kosub [b,*]

[a] Department of Computer Science, Rochester Institute of Technology, Rochester, NY 14623, USA
[b] Department of Computer & Information Science, University of Konstanz, Box 67, D-78457 Konstanz, Germany

ARTICLE INFO

Article history:
Received 16 September 2013
Received in revised form 20 January 2015
Accepted 22 January 2015
Available online 26 January 2015
Communicated by M. Kiwi

Keywords:
Discrete dynamical systems
Fixed point
Algorithms and complexity

ABSTRACT

We present dichotomy theorems regarding the computational complexity of counting fixed points in boolean (discrete) dynamical systems, i.e., finite discrete dynamical systems over the domain $\{0,1\}$. For a class $\mathcal{F}$ of boolean functions and a class $\mathcal{G}$ of graphs, an $(\mathcal{F},\mathcal{G})$-system is a boolean dynamical system with local transitions functions lying in $\mathcal{F}$ and graphs in $\mathcal{G}$. We show that, if local transition functions are given by lookup tables, then the following complexity classification holds: Let $\mathcal{F}$ be a class of boolean functions closed under superposition and let $\mathcal{G}$ be a graph class closed under taking minors. If $\mathcal{F}$ contains all min-functions, all max-functions, or all self-dual and monotone functions, and $\mathcal{G}$ contains all planar graphs, then it is #P-complete to compute the number of fixed points in an $(\mathcal{F},\mathcal{G})$-system; otherwise it is computable in polynomial time. We also prove a dichotomy theorem for the case that local transition functions are given by formulas (over logical bases). This theorem has a significantly more complicated structure than the theorem for lookup tables. A corresponding theorem for boolean circuits coincides with the theorem for formulas.

1. Introduction

Efforts to understand the behavior of complex systems have led to various models for finite discrete dynamical systems, including (finite) cellular automata (see, e.g., [33,34]), discrete recurrent Hopfield networks (see, e.g., [17,1]), and concurrent and communicating finite state machines (see, e.g., [21,24]). A fairly general class of systems was introduced in [6]. There, a finite discrete dynamical system (over a finite domain $\mathcal{D}$) is defined as: (a) a finite undirected graph, where vertices correspond to variables and edges correspond to an interdependence between the two connected variables, (b) for each vertex v, a local transition function that maps tuples of values (belonging to $\mathcal{D}$) of v and v's neighbors to values of v, and (c) an update schedule that governs which variables are allowed to update their values in which time steps. Formal definitions can be found in Section 2.

A central goal in the study of dynamical systems is to classify them according to how easy it is to predict their behavior. In a finite, discrete setting, a certain behavioral pattern is considered predictable if it can be decided in polynomial time whether a given system will show the pattern [10]. Although the pattern reachability problem is, in general, an intractable

☆ A preliminary version of this paper [19] was presented at the 10th Italian Conference on Theoretical Computer Science (ICTCS'07).

* Corresponding author. Tel.: +49 7531 88 5263; fax: +49 7531 88 3577.
E-mail addresses: cmh@cs.rit.edu (C.M. Homan), Sven.Kosub@uni-konstanz.de (S. Kosub).

http://dx.doi.org/10.1016/j.tcs.2015.01.040
0304-3975/

problem, i.e., at least NP-hard (see, e.g., [15,27,4]), many tractable classes of patterns and systems have been identified. However, there is still a serious demand for exhaustive characterizations of *islands of predictability*.

A fundamental behavioral pattern is the *fixed point* (a.k.a., homogeneous state, or equilibrium). A value assignment to the variables of a system is a fixed point if the values assigned to the variables are left unchanged after the system updates them. Note that fixed points are invariant under changes of the update regime. In this sense, they can be seen as a particularly robust behavior. A series of recent papers has been devoted to the identification of finite systems with tractable/intractable fixed-point analyses [5,30,28,29,18]. Precise boundaries are known for which systems finding fixed points can be done in polynomial time. For the fixed-point counting problem this is far less so.

Contributions of the paper. We prove dichotomy theorems on the computational complexity of counting fixed points in boolean (discrete) dynamical systems, i.e., finite discrete dynamical systems over the domain $\{0, 1\}$. For a class $\mathcal{F}$ of boolean functions and a class $\mathcal{G}$ of graphs, an $(\mathcal{F}, \mathcal{G})$-system is a boolean dynamical system with local transition functions lying in $\mathcal{F}$ and a graph lying in $\mathcal{G}$. Following [18], Post classes (a.k.a., clones) and forbidden-minor classes are used to classify $(\mathcal{F}, \mathcal{G})$-systems. In Section 4 we state the following theorem (Theorem 9): Let $\mathcal{F}$ be a class of boolean function closed under superposition and let $\mathcal{G}$ be a minor-closed graph class. If $\mathcal{F}$ contains all min-functions, all max-functions, or all self-dual and monotone functions, and $\mathcal{G}$ contains all planar graphs, then it is #P-complete to compute the number of the fixed points in an $(\mathcal{F}, \mathcal{G})$-system; otherwise it is computable in polynomial time. Here, the local transition functions are supposed to be given by lookup tables. In addition, we prove a dichotomy theorem (Theorem 19) for the case that local transition functions are given by formulas (over logical bases). Moreover, the corresponding theorem for boolean circuits coincides with the theorem for formulas. The theorem has a significantly more complicated structure than for lookup tables.

Related work. There is a series of work regarding the complexity of certain computational problems for finite discrete dynamical systems (see, e.g., [15,27,5,2,3,30,28,4] and the references therein). The problem of counting fixed points of boolean dynamical systems has been studied in [30,28,29]. To summarize: counting the number of fixed points is in general #P-complete. So is counting the number of fixed points for boolean dynamical systems with monotone local transition functions over planar bipartite graphs or over uniformly sparse graphs. We note that all system classes considered here are based on formula or circuit representations. That is, if they fit into our scheme at all, then the intractability results fall into the scope of Theorem 19 (and are covered there). Detailed studies of computational problems related to fixed-point existence have been reported in [5,18]. In [18], a complete classification of the fixed-point existence problem with respect to the analysis framework we use in this paper was shown.

2. The dynamical systems framework

In this section we present a formal framework for dynamical systems. A fairly general approach is motivated by the theoretical study of simulations. The following is based on [8,6,7,18].

The underlying network structure of a dynamical system is given by an undirected graph $G = (V, E)$ without multi-edges and loops. We suppose that the set V of vertices is ordered. So, without loss of generality, we assume $V = \{1, 2, \ldots, n\}$. For any vertex set $U \subseteq V$, let $N_G(U)$ denote the neighbors of U in G, i.e.,

$$N_G(U) =_{\text{def}} \big\{ j \mid j \notin U \text{ and there is an } i \in U \text{ such that } \{i, j\} \in E \big\}.$$

If $U = \{i\}$ for some vertex i, then we use $N_G(i)$ as a shorthand for $N_G(\{i\})$. The degree d_i of a vertex i is the number of its neighbors, i.e., $d_i =_{\text{def}} \|N_G(i)\|$.

A *dynamical system S over a domain $\mathcal{D}$* is a pair (G, F) where $G = (V, E)$ is an undirected graph (the *network*) and $F = \{f_i \mid i \in V\}$ is a set of *local transition functions* $f_i : \mathcal{D}^{d_i+1} \to \mathcal{D}$. The intuition of the definition is that each vertex i corresponds to an active element (entity, agent, actor etc.) which is always in some state x_i and which is capable to change its state, if necessary. The domain of S formalizes the set of possible states of all vertices of the network, i.e., for all $i \in V$, it always holds that $x_i \in \mathcal{D}$. A vector $\vec{x} = (x_i)_{i \in V}$ such that $x_i \in \mathcal{D}$ for all $i \in V$ is called a *configuration of S*. The local transition function f_i for some vertex i describes how i changes its state depending on the states of its neighbors $N_G(i)$ in the network and its own state.

We are particularly interested in dynamical systems operating on a discrete time-scale. A *discrete dynamical system* $\mathcal{S} = (S, \alpha)$ consists of a dynamical system S and a mapping $\alpha : \{1, \ldots, T\} \to \mathcal{P}(V)$, where V is the set of vertices of the network of S and $T \in \mathbb{N}$. The mapping α is called the *update schedule* and specifies which state updates are realized at certain time-steps: for $t \in \{1, \ldots, T\}$, $\alpha(t)$ specifies those vertices that simultaneously update their states in step t.

A discrete dynamical system $\mathcal{S} = (S, \alpha)$ over domain $\mathcal{D}$ induces a global map $\mathbf{F}_{\mathcal{S}} : \mathcal{D}^n \to \mathcal{D}^n$ where n is the number of vertices of S. For each vertex $i \in V$, define an *activity function* φ_i for a set $U \subseteq V$ and $\vec{x} = (x_1, \ldots, x_n) \in \mathcal{D}^n$ by

$$\varphi_i[U](\vec{x}) =_{\text{def}} \begin{cases} f_i(x_{i_1}, \ldots, x_{i_{d_i+1}}) & \text{if } i \in U \\ x_i & \text{if } i \notin U \end{cases}$$

where $\{i_1, i_2, \ldots, i_{d_i+1}\} = \{i\} \cup N_G(i)$. For a set $U \subseteq V$, define the *global transition function* $\mathbf{F}_S[U] : \mathcal{D}^n \to \mathcal{D}^n$ for all $\vec{x} \in \mathcal{D}^n$ by

$$\mathbf{F}_S[U](\vec{x}) =_{\text{def}} \big(\varphi_1[U](\vec{x}), \ldots, \varphi_n[U](\vec{x})\big).$$