
ELSEVIER Applied Mathematics and Computation 142 (2003) 521–543

APPLIED MATHEMATICS AND COMPUTATION

www.elsevier.com/locate/amc

Threshold agent networks: an approach to modelling and simulation

George Voutsadakis [1]

Physical Science Laboratory, New Mexico State University, Las Cruces, NM 88003, USA

Abstract

Threshold agent networks (TANs), a discretized version of threshold or neural networks, are proposed as alternative platforms to sequential dynamical systems for modelling computer simulations. It is argued that each model has its own advantages and disadvantages compared to the other and the choice on each occasion should depend on the particular characteristics of the application at hand. Some results on the expressive power and the limitations of TANs are presented. Finally, equivalence classes of TANs that are introduced based on characteristics of their state spaces are studied in detail and upper bounds are given on their cardinalities.

Keywords: Finite dynamical systems; Automata networks; Threshold networks; Neural networks; Threshold agent networks; Sequential dynamical systems; Computer simulations

1. Introduction

Although modelling and computer simulations of physical phenomena have a long history of existence and development, it was not but until very recently [1,2] that the question of modelling computer simulations per se was brought into the limelight. Sequential dynamical systems (SDSs) were introduced for

E-mail address: gvoutsad@psl.nmsu.edu (G. Voutsadakis).
URL: http://pigozzi.psl.nmsu.edu.
[1] This research was partially supported by a grant from the US Department of Defense.

0096-3003/02/$ - see front matter © 2002 Published by Elsevier Science Inc.
doi:10.1016/S0096-3003(02)00337-5

this purpose. A very eloquent description of the reasoning that led to the introduction of SDSs as the most appropriate model for computer simulations is provided in [7]. Such a model, it is argued, must consist of objects that have a state and the ability to interact with each other. Interaction is local in the sense that each object is able to interact only with its "neighbors" rather than with every other object taking part in the simulation. Neighborhood of the objects is determined by environmental considerations, i.e., by the space in which the simulation is supposed to be taking place. These chains of local interactions result in updating the states of the objects. The updates must occur in some specified order since updating the state of an object may affect the way the state of another object is updated and, therefore, the updating order is very relevant to the outcome of the simulation. This general framework is then specialized to a digital sequential machine with arbitrary boolean function computers. In this case the states become vectors of binary digits, one for each object, and the objects are residing in the space taking the form of vertices of a simple graph that determines via its adjacency relation the neighborhood relation of the objects. The local update functions are arbitrary boolean functions, one for each vertex, that depend only on the values of the vertex itself and of its adjacent vertices. Finally, the order in which the updates represented by these functions should be applied is given by a permutation of the vertices. Therefore, the final version of the model is formalized as follows.

Let $G = \langle V, E\rangle$ be a loop-free graph with vertex set $V = \{1, 2, \ldots, n\}$ and k the two element field with elements 0 and 1. For each $i \in V$, suppose that there is a function $F_i : k^n \to k^n$, that only changes the value of the ith position and only depends on the ith position and those positions j, such that $(j, i) \in E$. The F_is are the *local update functions*. Now let $\pi \in S_n$ be a permutation of V. π is called an *update schedule*. The functions F_i are composed in the order prescribed by π to obtain the function $F(G, \pi) = F_{\pi(n)} \circ F_{\pi(n-1)} \circ \cdots \circ F_{\pi(1)}$: $k^n \to k^n$. We call the function $F(G, \pi)$ the SDS determined by G, the local update functions F_i and the update schedule $\pi \in S_n$.

The transition from the higher level description of a computer simulation to a lower level, very close to the actual implementation of the simulation in a computer system, has introduced some peculiarities in the SDS model that may be natural in some contexts but may seem quite unnatural in others. It would have made perfect sense for many of the characteristics that are considered set in a certain way in this model to be set differently in another model. Some of these will now be pointed out. Our goal is to raise some doubts as to whether SDSs constitute "the" right model for computer simulations or, rather, "a" model that is very useful for modelling some types of simulations, while other models should be used in other types of simulations. In Section 2, an alternative model will be introduced. It will be argued that pluralism in selection is desirable as long as one has in his arsenal powerful tools in converting from one model to the other and in combining different models to construct bigger

ones or using different models to decompose in a structured way bigger and more complex simulations.

In defining the space of the simulation, i.e., the underlying graph of an SDS to be a simple undirected graph, the assumption is made implicitly that if one object affects another object in a simulation then the second object is necessarily in a position to affect the first. While this is a sound assumption from the topological viewpoint, it may fail otherwise for several reasons. So it seems very natural to relax this condition to avoid the enforced symmetry. The graph should, for many applications, be a digraph rather than an undirected graph. A second issue that arises is the type of the functions used as local update functions. The SDS model allows arbitrary boolean functions to be used as local update functions. This is consistent with the higher level point of view since at that level each of the objects will be able to compute any such function. On the other hand the final component in the global update function corresponding to that object will have a restricted form due to the spatial relations in the model and the form of the update schedule. If the opposite point of view is taken, staying closer to the actual implementation characteristics of the simulation in an electronic machine, it may be preferable to use boolean threshold functions as local update functions rather than arbitrary functions. This has, of course, the disadvantage of being somewhat restrictive. On the other hand, it has the advantage of narrowing the class of functions used to a more manageable one and, also, that the class of threshold functions has been in use in engineering and simulation science for quite a long time so that results from these fields may be readily pulled out and used in the present context. Finally, another point of contention is the sequentiality of the model. SDSs in this respect are taking again the lower, closer to the machine implementation level. So this assumption is sound as long as attention is restricted or, at least focused, on the actual implementation of the simulation in a sequential machine using a language that follows a sequential model of computation. This assumption, however, fails if a parallel machine or a distributed system is used for the simulation or a language that follows an inherently parallel prototype is used to program the simulation.

These are some of the theoretical reasons that led to the introduction of threshold agent networks (TANs) as an alternative platform for computer simulations to the SDS platform. No claim is made that TANs are more successful or more global than SDSs. It is strongly believed, however, that each model has its own advantages and disadvantages and each one is more appropriate than the other depending on the nature of the specific application as was argued above. Having more than one model is not necessarily a drawback. In fact, in [5] it is shown that there exist natural ways to transform one system into the other. Thus, under these circumstances the added flexibility in the initial choice relative to the application at hand is desirable and both models

may be combined should the need to build or decompose some larger model arise.

2. Threshold agent networks: the basics

Let $G = \langle V, E\rangle$ be a loop-free graph with vertex set $V = \{1, 2, \ldots, n\}$ and k the two element lattice with elements 0 and 1. For each $i \in V$, suppose that we are given a function $F_i : k^n \to k^n$, that only changes the value of the ith position and only depends on the ith position and those positions j, such that $(j, i) \in E$. Call the F_is the local update functions. Now let $\pi \in S_n$ be a permutation of V. π is called an update schedule. The functions F_i are composed in the order prescribed by π to obtain the function $F(G, \pi) = F_{\pi(n)} \circ F_{\pi(n-1)} \circ \cdots \circ F_{\pi(1)} : k^n \to k^n$. We call the function $F(G, \pi)$ the SDS determined by G, the local functions F_i and the update schedule $\pi \in S_n$. (See [2,7].)

A SDS is said to be *positive* if all local update functions are monotone, i.e., if, for all $1 \leqslant i \leqslant n$, and all $x_1, \ldots, x_n, y_1, \ldots, y_n \in k$, with $x_j \leqslant y_j$, $1 \leqslant j \leqslant n$,

$$F_i(x_1, \ldots, x_n) \leqslant F_i(y_1, \ldots, y_n).$$

It is well known that a boolean function is monotone if and only if it may be expressed as a term function for a term containing only the operation symbols 0, 1, $\wedge$ and $\vee$. Thus, an equivalent formulation of this definition would be that an SDS is positive if all local update functions are expressible as term functions in the language of boolean algebras without the negation symbol.

As an example consider the graph G with vertex set $V = \{1, 2, 3, 4\}$ that is depicted in Fig. 1.

Suppose that the local functions are given by

$$F_1 = x_2 \oplus x_4, \quad F_2 = x_1 \oplus x_3, \quad F_3 = x_2 \oplus x_4, \quad F_4 = x_1 \oplus x_3,$$

where by $\oplus$ is denoted the XOR binary operation on $\{0, 1\}$, given by the table

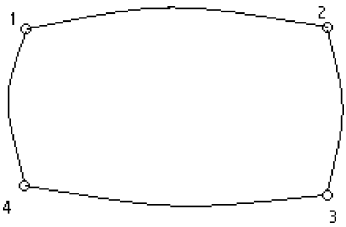

Fig. 1. An example of a SDS.

$\oplus$	0	1
0	0	1
1	1	0

Suppose also that the update schedule is $\pi = \mathrm{id}$, the identity permutation on V.

It is not difficult to see that the resulting SDS is given by

$$F(G,\pi)(x_1,x_2,x_3,x_4) = (x_2 \oplus x_4, x_2 \oplus x_3 \oplus x_4, x_2 \oplus x_3, x_3 \oplus x_4).$$

Then the following table describes two possible runs of the system, the first with initial condition $(x_1^0,x_2^0,x_3^0,x_4^0) = (1,0,0,0)$ and the second with initial condition $(x_1^0,x_2^0,x_3^0,x_4^0) = (1,1,0,0)$.

x_1	x_2	x_3	x_4	x_1	x_2	x_3	x_4
1	0	0	0	1	1	0	0
0	0	0	0	1	1	1	0
0	0	0	0	1	0	0	1
⋮				1	1	0	1
				0	0	1	1
				1	0	1	0
				0	1	1	1
				0	1	0	0
				1	1	1	0
				1	0	0	1
				⋮			

A TAN consists of a collection $A = \{A_1,\ldots,A_n\}$ of *agents*, where each agent A_i is formally an ordered pair $A_i = \langle k_i, P_i\rangle$, where k_i is an integer, the *threshold* of agent i, and $P_i \subseteq \{1,\ldots,n\}$ its *output set*. Agent i will be "active" at time j if at least k_i agents that have i in their output sets are active at time $j-1$. Note that if k_i is negative, then agent i will always be active at time j except if at least $-k_i$ agents that have i in their output sets are active at round $j-1$. As a consequence, negative thresholds may be used to model inhibitory as well as stimulatory behavior. P_i, $1 \leqslant i \leqslant n$, is the set of agents that will be affected by agent i at the end of each time step whenever agent i has been active at the end of the

previous time step. This dynamical behavior is formally expressed by the function $h : k^n \to k^n$ defined as follows: first, given a condition c that an n-tuple $\langle x_1, \ldots, x_n \rangle \in k^n$ may or may not satisfy, let $\chi_c : k^n \to k$ be the characteristic function of c, i.e.,

$$\chi_c(x_1, \ldots, x_n) = \begin{cases} 1, & \text{if } \langle x_1, \ldots, x_n \rangle \text{ satisfies } c, \\ 0, & \text{otherwise.} \end{cases}$$

Then define the functions $\chi_i : k^n \to k$, $1 \leqslant i \leqslant n$, by

$$\chi_i = \begin{cases} \chi_{\{|\{j : x_j = 1 \text{ and } i \in P_j\}| \geqslant k_i\}}, & \text{if } k_i \geqslant 0, \\ \chi_{\{|\{j : x_j = 1 \text{ and } i \in P_j\}| < -k_i\}}, & \text{if } k_i < 0. \end{cases}$$

Finally, set

$$h(x_1, x_2, \ldots, x_n) = (\chi_1(x_1, \ldots, x_n), \ldots, \chi_n(x_1, \ldots, x_n)),$$

h is called the *dynamics* of the TAN. The state of agent i at time step j is the ith coordinate of the jth *state vector*. The zeroth state vector is also referred to as an *initial condition*. The TAN on the agent set $A = \{A_i\}_{i \in I}$ will be denoted by A.

A TAN is said to be *positive* if all thresholds k_i, $1 \leqslant i \leqslant n$ are nonnegative, i.e., if, for all $1 \leqslant i \leqslant n$, $k_i \geqslant 0$.

As an example consider the following TAN with four agents. The second column of the table gives each agent's threshold and the third column gives each agent's output set.

Number	Threshold	Output Set
1	1	$\{2, 4\}$
2	-1	$\{1, 3\}$
3	1	$\{2, 4\}$
4	-1	$\{1, 3\}$

The sequence of state vectors for a run of this TAN with initial condition $(0, 0, 0, 0)$ is given in the following table

x_1	x_2	x_3	x_4
0	0	0	0
0	1	0	1
1	1	1	1
1	0	1	0
0	0	0	0
⋮			

The output sets may also be viewed as ordered lists of agent indices that collectively determine a binary relation E of "adjacency" between the agents:

$$(i,j) \in E \quad \text{if and only if } j \in P_i.$$

If this binary relation is converted to a graph representation, then a TAN with n agents may be equivalently defined as a pair consisting of a directed graph $G = \langle I, E\rangle$ on $|I| = n$ vertices together with a collection of functions $\{f_i\}_{i\in I}$, such that f_i depends only on $E_i = \{j \in I : (j,i) \in E\}$ and has the form

$$f_i(x_j : j \in E_i) = \begin{cases} T(\sum_{j\in E_i} x_j - k_i), & \text{if } k_i \geqslant 0, \\ T(-\sum_{j\in E_i} x_j - k_i), & \text{if } k_i < 0, \end{cases}$$

where $T : \mathbb{R} \to \mathbb{R}$ ($\mathbb{R}$ denotes the set of real numbers) is such that

$$T(u) = \begin{cases} 1, & \text{if } u \geqslant 0, \\ 0, & \text{otherwise.} \end{cases}$$

In this formulation, it is easy to see that TANs form a special subclass of threshold or neural networks, which, in turn, form a special subclass of finite automata networks [4].

The *state space* of a TAN $A = \{A_i\}_{1\leqslant i\leqslant n}$ with dynamics $h : k^n \to k^n$ is the directed graph $S_A = \langle V, E\rangle$ with vertices all the 2^n n-bit sequences (state vectors) and edges

$$(x,y) \in E \quad \text{if and only if } y = h(x), \quad \text{for all } x, y \in k^n.$$

Two state spaces S_A and S_B are *isomorphic*, denoted $S_A \cong S_B$, if they are isomorphic as directed graphs.

Let π be a permutation of $\{1,\ldots,n\}$. Given a binary sequence $x \in k^n$, denote by $\pi(x)$ the sequence $\langle x_{\pi(1)}, x_{\pi(2)}, \ldots, x_{\pi(n)}\rangle$. The *permutation* $\pi(S_A)$ *of a state space* $S_A = \langle k^n, E\rangle$ *by* π is the directed graph with vertices k^n and edge set $\{(\pi(x), \pi(y)) : (x,y) \in E\}$.

3. On the dynamics of threshold agent networks

In this section, some simple results pertaining to the expressive power of TANs will be formulated. By expressive power is meant the nature and the variety of the digraphs over k^n that may be produced as state spaces of TANs. Boolean circuits are used as an auxiliary intermediate step in transforming a given collection of boolean functions into a suitable TAN whose operation simulates in some sense the given functions.

First, it is obvious that all those digraphs that are state spaces of TANs must be graphs of functions, since each is the graph of the dynamics function of some TAN. That is, the following relation holds, for all state spaces $S = \langle k^n, E\rangle$,

$$(x,y) \in E \quad \text{and} \quad (x,z) \in E \quad \text{imply } y = z, \quad \text{for all } x,y,z \in k^n.$$

Using the pigeon-hole principle (or quoting well-known results on finite dynamical systems, for instance, Section 7.2 in [3]), the following proposition may be proved. It expresses the simple fact that, starting from an arbitrary state, any given TAN will eventually reach a uniquely determined finite limit cycle. This is true, more generally, for all finite dynamical systems.

Proposition 1. *For every TAN with agent set $A = \{A_i\}_{1 \leqslant i \leqslant n}$ and for every initial condition x^0, there exist a minimum nonnegative integer τ and a minimum positive integer T, such that the sequence of state vectors x^s, $s \geqslant \tau$, is periodic with period T.*

To illustrate Proposition 1, take the TAN that was described in Section 2. In that case, it is obvious that $\tau = 0$ and $T = 4$.

In the following theorem, it is shown that any functional relation between binary vectors of length n may be simulated as a periodic projection onto n coordinates of an appropriately constructed TAN with at least n agents.

Theorem 2. *Let v^s, $s \in \omega$, be a sequence of vectors $v^s = \langle v_1^s, \ldots, v_n^s \rangle \in k^n$, such that $v^{s_1} = v^{s_2}$ implies $v^{s_1+1} = v^{s_2+1}$, for all $s_1, s_2 \in \omega$. Then, there exist a TAN with agent set $A = \{A_i\}_{i \in I}$, a subset $J \subseteq I$, with $|J| = n$, and a positive integer d, such that*

$$\langle \langle x_j^{ds} : j \in J \rangle : s \in \omega \rangle = \langle v^s : s \in \omega \rangle,$$

where by x_i^t is denoted the state of agent A_i at time step t.

Proof. Take the first $2^n + 1$ vectors $v^0, \ldots, v^{2^n}$. Define functions $f_i : k^n \to k$, $1 \leqslant i \leqslant n$, by

$$f_i(x_1, \ldots, x_n) = \begin{cases} v_i^{j+1}, & \text{if } v^j = \langle x_1, \ldots, x_n \rangle, \text{ for some } 0 \leqslant j \leqslant 2^n, \\ 0, & \text{otherwise,} \end{cases}$$

f_i, $1 \leqslant i \leqslant n$ is well-defined because of the assumption made about the sequence v^s, $s \in \omega$.

Now construct the boolean circuit C that computes the f_is in terms of the inputs $x_1, \ldots, x_n$ and has uniform path length d. In other words, all paths from inputs to outputs of C contain exactly the same number of logical gates. This can always be done by introducing redundant propagators, if necessary. Propagators are 1-input-1-output gates that compute the identity function. This requirement is crucial, since only if it is satisfied, are the outputs of the circuit guaranteed to simulate correctly every d time steps the given vectors. Then construct the TAN $A(C)$ associated to this boolean circuit. (Rather than describing the details of this construction formally, we will give an illustration

of the construction following this proof.) The projection of the state vectors of this TAN on the set of agents corresponding to the input gates taken every d time steps will then give the original sequence of vectors v^s, $s \in \omega$. □

As an example, consider the following sequence of vectors in k^3

$$\begin{matrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ \vdots & & \end{matrix}$$

This sequence satisfies the requirement of Theorem 2. Thus, we may construct the following functions f_1, f_2, f_3: $k^3 \to k$:

x_1	x_2	x_3	f_1	f_2	f_3
0	0	0	0	0	1
0	0	1	0	1	1
0	1	1	0	1	0
0	1	0	0	0	0
1	0	0	0	0	0
1	0	1	0	0	0
1	1	0	0	0	0
1	1	1	0	0	0

We have

$$f_1(x_1,x_2,x_3) = 0, \quad f_2(x_1,x_2,x_3) = \overline{x_1} \wedge x_3, \quad f_3(x_1,x_2,x_3) = \overline{x_1} \wedge \overline{x_2}.$$

Therefore, the boolean circuit of path length, or depth, 3 of Fig. 2 computes f_1, f_2 and f_3. Notice the introduction of the propagators 4 and 7 to force uniform path length 3.

Now the following TAN $A(C)$ with 10 agents which are described by the following table is obtained from the circuit C. To each propagating gate in the input is assigned threshold 1, to each negation gate threshold −1, to each AND gate threshold equal to the number of inputs of that gate and to each OR gate (if there are any) threshold 1. Then the output set of each gate is taken to be the set of all those gates whose input is wired to the output of that gate. As mentioned above, it is crucial that all paths from inputs to outputs are of the

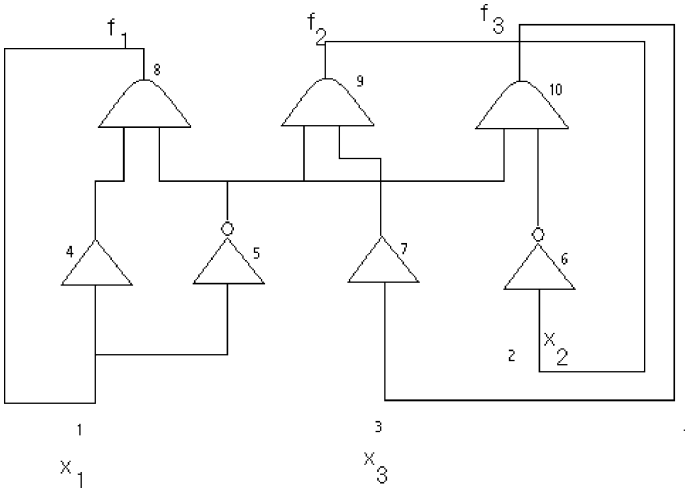

Fig. 2. A circuit C computing a given sequence of binary vectors.

same length in terms of intermediate gates so that the simulation of the circuit by the resulting TAN be performed correctly.

Agent	Threshold	Output Set
1	1	$\{4,5\}$
2	1	$\{6\}$
3	1	$\{7\}$
4	1	$\{8\}$
5	-1	$\{8,9,10\}$
6	-1	$\{10\}$
7	1	$\{9\}$
8	2	$\{1\}$
9	2	$\{2\}$
10	2	$\{3\}$

Running this TAN and taking projections of the state vectors onto the first three coordinates every three steps will yield the states that we started with. The following table gives the values of the run with initial condition $(0,0,\ldots,0)$.

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	0	0	0	0
0	0	0	0	1	1	0	0	0	1
0	0	1	0	1	1	0	0	0	1
0	0	1	0	1	1	1	0	0	1
0	0	1	0	1	1	1	0	1	1
0	1	1	0	1	1	1	0	1	1
0	1	1	0	1	0	1	0	1	1
0	1	1	0	1	0	1	0	1	0
0	1	0	0	1	0	1	0	1	0
0	1	0	0	1	0	0	0	1	0
0	1	0	0	1	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	0	0	1	1	0	0	0	0
⋮									

It is interesting to note that the dynamic behavior depicted by the sequence

0 0 0
0 0 1
0 1 1
0 1 0
0 0 0
⋮

is also achievable by the following TAN with just three agents.

Agent	Threshold	Output Set
1	1	$\emptyset$
2	1	$\{3\}$
3	-1	$\{2\}$

But, in general the following negative result is true.

Proposition 3. *There exists a sequence* $v^s = \langle v_1^s, v_2^s, v_3^s \rangle \in k^3$, $s \in \omega$, *satisfying the condition* $v^{s_1} = v^{s_2}$ *implies* $v^{s_1+1} = v^{s_2+1}$, *for all* $s_1, s_2 \in \omega$, *that is not the state vector sequence of any TAN with just three agents.*

Proof. Consider the sequence

$$\begin{array}{ccc} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \\ \vdots & & \end{array}$$

First, since the first state vector (after the initial condition) is $(0,0,1)$, agents 1 and 2 must have positive thresholds, since they would have been active otherwise, whereas agent 3 must have either 0 or a negative threshold. Now, from the second state vector $(1,1,0)$ we may conclude that both agents 1 and 2 have thresholds 1 and are in 3's output set and also that agent 3 must have threshold -1 and be in its own output set. But this cannot happen as, then, the fourth step $(1,0,1) \to (0,0,1)$ could not occur. (Agents 1 and 2, being in 3's output set and having thresholds 1, must be active at this step.) $\square$

Theorem 2 may be generalized to obtain the following refinement.

Proposition 4. *Let v^s, $0 \leqslant s < m$ be a finite sequence of vectors $v^s = \langle v_1^s, \ldots, v_n^s \rangle \in k^n$. Then, there exist a TAN with agent set $A = \{A_i\}_{i \in I}$, a subset $J \subseteq I$ and a positive integer d, such that*

$$\langle \langle x_j^{ds} : j \in J \rangle : 0 \leqslant s < m \rangle = \langle v^s : 0 \leqslant s < m \rangle.$$

Proof. First, append $\lceil \log_2 m \rceil$ bits to all vectors v^s, $s < m$, to produce the sequence of vectors

$$\begin{array}{lcl} u^0 & = & v^0 00 \ldots 0 \\ u^1 & = & v^1 00 \ldots 1 \\ \vdots & & \\ u^{m-1} & = & v^{m-1} b_{\lceil \log_2 m \rceil} \ldots b_0 \end{array}$$

where $b_{\lceil \log_2 m \rceil} \ldots b_0$ is the binary representation of $m-1$. Now periodically repeat these augmented vectors to produce an infinite sequence u^s, $s \in \omega$, of vectors of length $n + \lceil \log_2 m \rceil$. Note that this infinite sequence satisfies the condition $u^{s_1} = u^{s_2}$ implies $u^{s_1+1} = u^{s_2+1}$, for all $s_1, s_2 \in \omega$. Therefore, Theorem 2 may be applied to obtain a TAN with agents $A = \{A_i\}_{i \in I}$, a subset $K \subseteq I$ and a positive integer $d > 0$, such that $\langle \langle x_j^{ds} : j \in K \rangle : s \in \omega \rangle = \langle u^s : s \in \omega \rangle$. Then, it is clear that there exists a subset $J \subseteq K \subseteq I$, such that $\langle \langle x_j^{ds} : j \in J \rangle : 0 \leqslant s < m \rangle = \langle v^s : 0 \leqslant s < m \rangle$. $\square$

As an example, take the sequence of five 2-vectors

$$00, 01, 11, 00, 10.$$

Construct the sequence of five 5-vectors

$$00000, 01001, 11010, 00011, 10100.$$

Then pass to the infinite sequence

$$00000, 01001, 11010, 00011, 10100, 00000, 01001, 11010, 00011, 10100, \ldots$$

and follow the construction of the example following Theorem 2.

4. Time dependent SDSs and TANs

In this section time dependent SDSs (TDSDSs) and time dependent TANs (TDTANs) are introduced. These are generalized versions of SDSs and TANs, respectively, in which the graphs, local update functions, update schedules and thresholds and output sets, respectively, are allowed to vary with time. A special class of TDSDSs, called recursively TDSDSs, is then introduced and, using the boolean circuit construction, it is shown that the state spaces of SDSs in this class may be simulated by state spaces of (nontime dependent) TANs.

A TDSDS of order n is a triple

$$F = \langle \{G(t)\}_{t\in\omega}, \{F_i(t)\}_{1\leqslant i\leqslant n, t\in\omega}, \{\pi(t)\}_{t\in\omega}\rangle.$$

It consists, first, of a collection of simple graphs $\{G(t) = \langle V(t), E(t)\rangle\}_{t\in\omega}$ on the n vertices $\{1, 2, \ldots, n\}$. Second, of a collection of functions $\{F_i(t)\colon 1\leqslant i\leqslant n, t\in\omega\}$, with $F_i(t) : k^n \to k^n$, for all i and t, such that $F_i(t)$ only changes the ith variable and only depends on the ith variable and the variables j, with $(j, i)\in E(t)$. Denote this set by $N_i(t)$, i.e., $N_i(t) = \{i\}\cup\{1\leqslant j\leqslant n : (j, i)\in E(t)\}$. $F_i(t)$ is the ith *local update function* at time t. Finally, $\{\pi(t)\}_{t\in\omega}$ is a collection of permutations of the set $\{1, \ldots, n\}$, called the *update schedule* valid (or in effect) at time t. All local update functions at time t may be composed according to the update schedule that is in effect at time t to obtain a time varying function

$$F(t) = F_{\pi(t)(n)}(t)\circ F_{\pi(t)(n-1)}(t)\circ\cdots\circ F_{\pi(t)(1)}(t).$$

Alternatively, a TDSDS of order n may be viewed as a collection $F = \{F(t)\}_{t\in\omega}$ of SDSs $F(t) = \langle G(t), \{F_i(t)\}_{1\leqslant i\leqslant n}, \pi(t)\rangle$, all of them of order n.

A TDTAN of order n consists of a collection of n *time dependent agents* $A = \{A_i(t)\}_{1\leqslant i\leqslant n, t\in\omega}$. Each agent $A_i = \{A_i(t)\}_{t\in\omega}$ is formally a collection of pairs $A_i(t) = \langle k_i(t), P_i(t)\rangle$, where $k_i(t)$ is an integer, called the *threshold* of agent i at time t, and $P_i(t)\subseteq\{1, 2, \ldots, n\}$ is a subset of the index set, called the *output set* of agent i at time t. As the model runs in discrete time steps indexed by the natural numbers, agent A_i at time step t will be active at time t provided that he

belongs to the output sets of at least $k_i(t)$ agents that were active at time step $t-1$. Note that negative thresholds are also allowed in this case with the intended meaning that if $k_i(t) < 0$, agent A_i will always be active at time step t, unless he belongs to the output sets of at least $-k_i(t)$ agents that were active at time step $t-1$.

This dynamics is made explicit by the *dynamics function*

$$h_t(x_1, \ldots, x_n) = (\chi_1(t)(x_1, \ldots, x_n), \ldots, \chi_n(t)(x_1, \ldots, x_n)),$$

where, for all $1 \leqslant i \leqslant n$, $t \in \omega$,

$$\chi_i(t) = \begin{cases} \chi_{\{|\{j : x_j(t)=1 \text{ and } i \in P_j(t)\}| \geqslant k_i(t)\}}, & \text{if } k_i(t) \geqslant 0, \\ \chi_{\{|\{j : x_j(t)=1 \text{ and } i \in P_j(t)\}| < -k_i(t)\}}, & \text{if } k_i(t) < 0. \end{cases}$$

The TDTAN with set of agents A and dynamics function h_t will be denoted by A. Time dependence will be clear from context and, thus, no confusion will hopefully arise.

An alternative description of a TDTAN A would be as a collection $A = \{A(t)\}_{t \in \omega}$ of TANs indexed by the natural numbers (and thought of as discrete time steps).

A TDSDS $F = \{F_t\}_{t \in \omega}$ is said to be *recursively time dependent* if there exists a positive integer w, called the *window* of F, such that, for every $t \in \omega$, F_t depends only on the sets $\{F_{t-i} : 1 \leqslant i \leqslant w\}$ and $\{\vec{x}(t-i) : 1 \leqslant i \leqslant w\}$.

The following theorem is an easy consequence of the fact that, given a boolean function on finitely many variables, a boolean circuit may be constructed that computes the function and the possibility of constructing a TAN whose sequence of state vectors, restricted appropriately in time and space, simulates the operation of the boolean circuit, as was shown in the example following Theorem 2.

Theorem 5. *For every recursive TDSDS F on n vertices and any window w, there exist a TAN with agent set $A = \{A_i\}_{i \in I}$, a subset $J \subseteq I$, with $|J| = n$, and a positive integer d, such that, for every initial condition v^0 for F, there exists an initial condition $\langle x_i^0 : i \in I \rangle$, such that*

$$\langle \langle x_j^{ds} : j \in J \rangle : s \in \omega \rangle = \langle v^s : s \in \omega \rangle,$$

where, as before, by x_i^t is denoted the state of agent A_i at time step t.

Proof. Let F be a recursively TDSDS. First construct the boolean function B that expresses the recursive dependence of the graph, the local update functions, the update schedule and the state at time step t on the graphs, the local update functions, the update schedules and the states at time steps $t-1, \ldots, t-w$. Then construct the boolean circuit $C(B)$ that simulates B. Care must be taken as before so that all paths from inputs to outputs in that boolean

circuit have uniform length d. Then use the construction in Section 3 to construct the TAN $A(C(B))$ that simulates the boolean circuit $C(B)$. □

Theorem 5, combined with results from [5], may be used to prove the following.

Corollary 6. *For every recursively TDSDS F on n vertices and window w, there exist an SDS F' on m vertices, a subset $J \subseteq \{1, \ldots, m\}$ and a positive integer d, such that, for every initial condition x^0 for F, there exists an initial condition x'^0 for F', such that*

$$\langle \langle x_j'^{ds} : j \in J \rangle : s \in \omega \rangle = \langle x^s : s \in \omega \rangle.$$

5. Equivalent and strongly equivalent TANs

In this section notions of equivalence between TANs are introduced based on characteristics of their state spaces. Then Pólya's theory of counting is invoked to count the number of equivalence classes into which these equivalence relations partition the class of all TANs of a specified order. The reader that is not familiar with Pólya's theory of counting is referred to [6] but, for the sake of completeness, a self-contained introduction is also presented in Appendix A.

5.1. Equivalence and strong equivalence

Two TANs A and B will be said to be *equivalent* if they have isomorphic state spaces, i.e., if $S_A \cong S_B$.

A very interesting notion intermediate between equivalence and isomorphism is the notion of strong equivalence. (By isomorphism here is meant an isomorphism in the category of TANs. The reader is referred to [8] for a detailed categorical treatment of the networks studied in this paper.) The notion of a permutation of a state space, that was introduced at the end of Section 2, is used to describe strong equivalence.

Let A be a TAN with set of agents $\{A_i\}_{1 \leqslant i \leqslant n}$ and state space S_A. In addition, let π be a permutation of $\{1, \ldots, n\}$. Recall that, by $\pi(S_A)$ is denoted the graph obtained from S_A by permuting the digits of each node according to π. The fact that, for every permutation π on $\{1, \ldots, n\}$, $\pi(S_A) \cong S_A$ justifies the following definition of strong equivalence between TANs.

Two TANs A and B with sets of agents $A = \{A_i\}_{1 \leqslant i \leqslant n}$ and $B = \{B_i\}_{1 \leqslant i \leqslant m}$, respectively, such that $A_i = \langle k_i, P_i \rangle$, $1 \leqslant i \leqslant n$, and $B_i = \langle l_i, Q_i \rangle$, $1 \leqslant i \leqslant m$, are said to be *strongly equivalent* if $m = n$ and there exists a permutation π of $\{1, \ldots, n\}$, such that $\pi(S_A) = S_B$.

To see that strong equivalence is strictly stronger than equivalence consider the following two TANs

Agent	Threshold	Output Set	Agent	Threshold	Output Set
1	1	{2}	1	−1	{2}
2	1	{1}	2	−1	{1}

It is not difficult to check from their state spaces (see Fig. 3) that they are equivalent but not strongly equivalent.

The following proposition may be easily proven by recalling that an isomorphism of TANs [8] provides a permutation of the vertices that preserves the input–output relations, i.e., satisfies exactly the strong equivalence specification on the state spaces of the TANs.

Proposition 7. *Two isomorphic TANs are strongly equivalent.*

5.2. On the number of strong equivalence classes

In this section Pólya's theory of counting is applied to obtain an upper bound on the number of strong equivalence classes of TANs over a fixed digraph in terms of the number n of agents in the network (or nodes in the digraph) and the elements and number of automorphisms in the automorphism group of the digraph. It is also shown that, for every n, this upper bound is tight for the case of TANs with n agents whose digraph is a cycle.

Let A be a TAN with set of agents $\{A_i\}_{1\leqslant i\leqslant n}$, $A_i = \langle k_i, P_i\rangle$, $1\leqslant i\leqslant n$, and dynamics h. Define the *labeled digraph of* A as the labeled digraph $G(A) = \langle V, E, l\rangle$ with set of nodes the set $\{1,\ldots,n\}$, set of edges the set of all

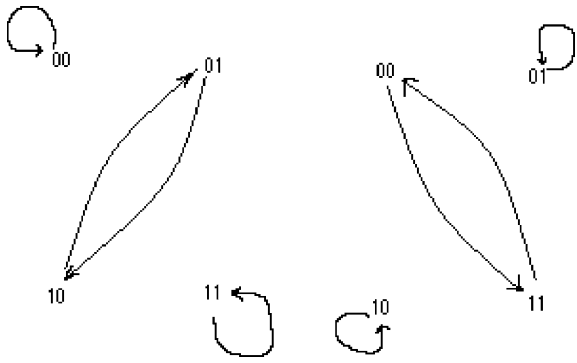

Fig. 3. The two state spaces of the TANs.

$(i,j) \in V \times V$, such that $j \in P_i$, and labels $l(i) = k_i$, for all $i \in V$. The *digraph of* A is then the graph $\langle V, E\rangle$.

Suppose that G is the digraph of A. Let $\mathrm{Aut}(G)$ denote the automorphism group of G and let d denote the maximum indegree of G, i.e.,

$$d = \max_{v \in V} \mathrm{indeg}(v).$$

Note that all negative thresholds in A that are less than $-d$ can be replaced by 0 without affecting the dynamics of the TAN. Also note that all thresholds in A that are greater than d may be replaced by $d+1$ without affecting the dynamics of the TAN either. Thus, to compute strong equivalence classes we need only consider thresholds in the range $\{-d, -d+1, \ldots, 0, 1, \ldots, d, d+1\}$. It is, furthermore, a straightforward observation that the number of strongly inequivalent TANs with digraph G is at most equal to the number of the different (up to permutation of the vertices inducing isomorphisms) labelings of the digraph G with integers in the range $-d, \ldots, d+1$. These are counted by Pólya's formula and they are $P_{\mathrm{Aut}(G)}(2d+2, 2d+2, \ldots)$, where $P_{\mathrm{Aut}(G)}$ denotes the cycle index of the automorphism group of the digraph G.

Hence we have

Theorem 8. *Let $G = \langle V, E\rangle$ be a directed graph with $d = \max_{v \in V} \mathrm{indeg}(v)$. There are at most $P_{\mathrm{Aut}(G)}(2d+2, 2d+2, \ldots)$ TANs with digraph G that are not strongly equivalent, where $P_{\mathrm{Aut}(G)}$ denotes the cycle index of the automorphism group of the digraph G.*

As an example, consider the digraph with two vertices 1, 2 and the directed edges $(1,2)$ and $(2,1)$, hence $d = 1$. The automorphism group of this digraph consists of the permutations

$$\begin{pmatrix} 1 & 2 \\ 1 & 2 \end{pmatrix}, \quad \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}.$$

Thus, $P_{\mathrm{Aut}(G)}(x_1, x_2) = \frac{1}{2}(x_1^2 + x_2)$. Therefore the upper bound on the strongly nonequivalent TANs on the digraph G is $P_{\mathrm{Aut}(G)}(4,4) = 10$. In fact, all 10 possible TANs on this digraph are strongly nonequivalent as shown in Fig. 4.

The idea just employed to obtain the bound on the number of strong equivalence classes of TANs may be used in conjunction with the generalization of Pólya's Theorem 14 of Appendix A to obtain an upper bound on the equivalence classes of TANs. Here, apart from the symmetry group of the vertices of the graph G of the TAN, another factor that contributes to having equivalent networks is taken into account. Note that if the numbers $-d, -d+1, \ldots, -1, 0, 1, \ldots, d, d+1$ are exchanged with the numbers $d, d-1, \ldots, 1, d+1, -1, \ldots, -d, 0$, respectively, then, in the state space domain two isomorphic state spaces will be obtained by flipping 0's and 1's in the

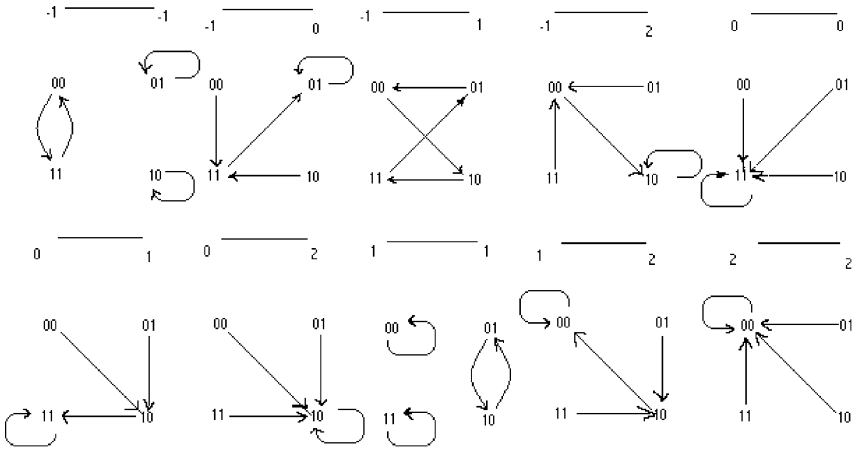

Fig. 4. Strongly nonequivalent TANs on a digraph with 2 vertices.

corresponding sequences. Thus, if the generalization of Pólya's theorem is applied to both the graph G with its group $\mathrm{Aut}(G)$ of automorphisms and to the set of colors $\{-d,\ldots,d+1\}$ with the group H of symmetries containing the identity and the permutation

$$\begin{pmatrix} -d & -d+1 & \cdots & -1 & 0 & 1 & \cdots & d & d+1 \\ d & d-1 & \cdots & 1 & d+1 & -1 & \cdots & -d & 0 \end{pmatrix}$$

of order 2, the following bound on the number of equivalence classes of TANs on G is obtained:

Theorem 9. *Let $G=\langle V,E\rangle$ be a directed graph with $d=\max_{v\in V}\mathrm{indeg}(v)$. There are at most*

$$P_{\mathrm{Aut}(G)}\left(\frac{\partial}{\partial z_1},\frac{\partial}{\partial z_2},\ldots\right)\exp((2d+2)(z_1+z_2+\cdots))\big|_{z_1=z_2=\cdots=0},$$

TANs with digraph G that are not equivalent, where $P_{\mathrm{Aut}(G)}$ denotes the cycle index of the automorphism group of the digraph G.

5.3. Sharpness of the strong equivalence bound

The task of showing that the bound established in Theorem 8 is sharp is taken up in this section. More precisely, let $G=\langle V,E\rangle$ be a directed graph with d being the maximum indegree of the vertices in V, $\mathrm{Aut}(G)$ the automorphism group of the digraph G and denote, as before, by $P_{\mathrm{Aut}(G)}$ the cycle index of the group $\mathrm{Aut}(G)$. By Theorem 8, there are at most $P_{\mathrm{Aut}(G)}(2d+2,2d+2,\ldots)$

TANs that are mutually strongly inequivalent. This is exactly the number of different labelings of the vertices of G with labels chosen from the set $\{-d, -d+1, \ldots, d+1\}$ that correspond to meaningful assignments of thresholds to the agents in the TAN. It is now shown that this bound is sharp for the TANs with underlying digraph the cyclic graph with n vertices, for all n. This is accomplished by showing that, if two TANs A and B are strongly equivalent, then there must exist a permutation of the vertices, such that, up to this permutation, the corresponding vertices must have the same output functions, i.e., must possess the same thresholds.

To this aim, let A and B be the two TANs with sets of agents $A = \{A_i\}_{1 \leqslant i \leqslant n}$ and $B = \{B_i\}_{1 \leqslant i \leqslant n}$, with $A_i = \langle k_i, P_i \rangle$ and $B_i = \langle l_i, P_i \rangle$, $1 \leqslant i \leqslant n$, where $P_i = \{i-1, i+1\}$, $2 \leqslant i \leqslant n-1$, and $P_1 = \{2, n\}$, $P_n = \{1, n-1\}$. Let $h^A : k^n \to k^n$ and $h^B : k^n \to k^n$ be the corresponding dynamics. Suppose that A and B are strongly equivalent. Thus, there exists a permutation $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$, such that, for all $1 \leqslant i \leqslant n$,

$$h_i^A(x_1, \ldots, x_n) = h_{\pi(i)}^B(x_{\pi(1)}, \ldots, x_{\pi(n)}).$$

But h_i^A and $h_{\pi(i)}^B$ only depend on x_{i-1}, x_{i+1} and $x_{\pi(i-1)}, x_{\pi(i+1)}$, respectively, whence, by slightly abusing notation,

$$h_i^A(x_{i-1}, x_{i+1}) = h_{\pi(i)}^B(x_{\pi(i-1)}, x_{\pi(i+1)}).$$

Since these functions are the same, agent i in TAN A must have exactly the same threshold as agent $\pi(i)$ in TAN B and also π must carry $i-1$ and $i+1$ to the vertices $\pi(i)-1, \pi(i)+1$, respectively, or vice versa, i.e., it must be an automorphism of the cycle C_n. Thus, the two strongly equivalent TANs A and B are in the same equivalence class with respect to automorphisms and labeling of the underlying digraph of the model, as counted by Theorem 8. This proves

Theorem 10. *Let C_n be the cycle with n vertices. There are exactly*

$$P_{\mathrm{Aut}(C_n)}(6, 6, \ldots),$$

TANs with digraph C_n that are not strongly equivalent, where $P_{\mathrm{Aut}(C_n)}$ denotes the cycle index of the automorphism group of the digraph C_n. Thus, the bound on the number of strong equivalence classes of TANs provided by Theorem 8 *is sharp for some digraphs with n vertices, for all n.*

Acknowledgements

The author wishes to thank Reinhard Laubenbacher for his encouragement and support. Also thanks to Bodo Pareigis for several useful discussions.

Finally, thanks to the two anonymous referees whose detailed comments and suggestions led to improvements in the presentation.

Appendix A. Pólya's theory of counting

In this appendix we briefly review the main elements of Pólya's theory of counting, that is used heavily in Section 5. The basic theory is introduced in the first subsection and a generalization is presented in the second. For more details and an excellent exposition, [6, Chapter 5], is strongly recommended.

A.1. Basic theory

Let D and R be the domain and codomain, respectively, of a set of $|R|^{|D|}$ functions. Suppose that a *weight* is assigned to each of the elements in R. If $r \in R$, denote by $w(r)$ its weight, which could be either a symbol or a number. The *store enumerator* of R is the sum of the weights of the elements in R:

$$\text{Store enumerator} = \sum_{r \in R} w(r).$$

The *weight of a function* $f : D \to R$, denoted $W(f)$, is the product of the weights of the images of the elements in D under f:

$$W(f) = \prod_{d \in D} w(f(d)).$$

The *inventory of a set of functions* is the sum of their weights:

$$\text{Inventory of a set of functions} = \sum_{\text{all } f \text{ in the set}} W(f).$$

Given a permutation group G on the domain D, the *equivalence relation* $\sim_G \subseteq R^D \times R^D$ on R^D *induced by* G is the relation defined by

$$f_1 \sim_G f_2 \quad \text{iff } f_1(d) = f_2(\pi(d)), \text{ for all } d \in D, \text{ for some } \pi \in G.$$

It is not hard to check that two functions belonging to the same equivalence class of $\sim_G$ have the same weight. This weight is said to be the *weight of the pattern* (*equivalence class*). The *inventory of a set of patterns* is the sum of the weights of the patterns in the set.

Let π be a permutation with b_1 cycles of length 1, b_2 cycles of length 2, …, b_k cycles of length k, and so on. The monomial $x_1^{b_1} x_2^{b_2} \cdots x_k^{b_k} \cdots$ will be called the *cycle structure representation* of π. The *cycle index* P_G of the permutation group G is the sum of the cycle structure representations of the permutations in G divided by the number of permutations in G:

$$P_G(x_1,\ldots,x_k,\ldots)=\frac{1}{|G|}\sum_{\pi\in G}x_1^{b_1}x_2^{b_2}\cdots x_k^{b_k}\cdots$$

Theorem 11 (Pólya). *The inventory of the equivalence classes of functions from domain D to codomain R is*

$$P_G\left(\sum_{r\in R}w(r),\sum_{r\in R}w(r)^2,\ldots,\sum_{r\in R}w(r)^k,\ldots\right).$$

Corollary 12. *The number of equivalence classes of functions from D* to *R is*

$$P_G(|R|,|R|,\ldots,|R|,\ldots).$$

As an example we illustrate the method for counting the number of ways of painting the four faces a, b, c and d of the pyramid of Fig. 5 with two colors of paints x and y.

We set $D=\{a,b,c,d\}$, $R=\{x,y\}$ and $w(x)=x$, $w(y)=y$. The permutation group for the pyramid is

$$G=\left\{\begin{pmatrix}a&b&c&d\\a&b&c&d\end{pmatrix},\begin{pmatrix}a&b&c&d\\b&c&a&d\end{pmatrix},\begin{pmatrix}a&b&c&d\\c&a&b&d\end{pmatrix}\right\}.$$

The first permutation has cycle structure representation x_1^4, and the second and the third x_1x_3. Thus, the cycle index of the group G is

$$P_G(x_1,x_2,x_3)=\tfrac{1}{3}(x_1^4+2x_1x_3)$$

and, therefore, the pattern inventory is

$$\tfrac{1}{3}[(x+y)^4+2(x+y)(x^3+y^3)]=x^4+y^4+2x^3y+2x^2y^2+2xy^3.$$

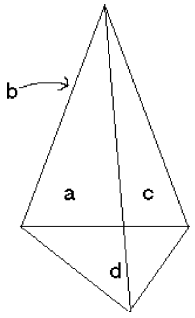

Fig. 5. The pyramid of the example on Pólya's theory.

Hence, there are eight distinct ways of painting the four faces of the pyramid with the colors x and y.

A.2. Generalization of Pólya's theory of counting

Pólya's theorem may be generalized in the following direction. In addition to the group G of permutations on the domain D we also have a group H of permutations on the codomain R. Two functions f_1 and f_2 are now related via the relation $\sim_{G,H} \subseteq R^D \times R^D$ if there exists a $\pi \in G$ and a $\tau \in H$, such that, for all $d \in D$,

$$\tau(f_1(d)) = f_2(\pi(d)).$$

Then, the following theorems hold [6, Theorems 5–7].

Theorem 13. *The number of equivalence classes of functions from D to R is given by*

$$\frac{1}{|G|}\frac{1}{|H|}\sum_{\pi\in G,\tau\in H}\psi(\pi,\tau),$$

where $\psi(\pi,\tau)$ is the number of functions f, such that $\tau(f(d)) = f(\pi(d))$, for all $d \in D$.

Theorem 14. *The number of equivalence classes of functions from D to R is the value of the expression*

$$P_G\left(\frac{\partial}{\partial z_1},\frac{\partial}{\partial z_2},\frac{\partial}{\partial z_3},\dots\right)\times P_H(\exp(z_1+z_2+z_3+\cdots),\\ \exp(2(z_2+z_4+z_6+\cdots)),\exp(3(z_3+z_6+z_9+\cdots)),\dots),$$

evaluated at $z_1 = z_2 = z_3 = \cdots = 0$.

As an example we compute the number of ways to distribute five objects two of which are indistinguishable to four boxes two of which are indistinguishable. In this case G contains the identity permutation and the permutation that interchanges the two indistinguishable objects and H contains the identity permutation and the permutation that interchanges the two indistinguishable boxes. Thus,

$$P_G = \tfrac{1}{2}(x_1^5 + x_1^3x_2) \quad \text{and} \quad P_H = \tfrac{1}{2}(x_1^4 + x_1^2x_2).$$

Therefore the number of distinct patterns is

$$\frac{1}{4}\left(\frac{\partial^5}{\partial z_1^5}+\frac{\partial^3}{\partial z_1^3}\frac{\partial}{\partial z_2}\right)[\exp(4(z_1+z_2))+\exp(2(z_1+z_2))\exp(2z_2)]|_{z_1=z_2=0}$$
$$=\frac{1}{4}(4^5+2^5+4^3\times 4+2^3\times 4)=336.$$

References

[1] C.L. Barrett, C.M. Reidys, Elements of a theory of simulation. I. Sequential CA over random graphs, Applied Mathematics and Computation 98 (1999) 241–259.
[2] C.L. Barrett, H.S. Mortveit, C.M. Reidys, Elements of a theory of simulation. II. Sequential dynamical systems, Applied Mathematics and Computation 107 (1999) 121–136.
[3] G. Birkhoff, T.C. Bartree, Modern Applied Algebra, McGraw-Hill, New York, 1970.
[4] E. Goles, S. Martínez, Neural and automata networks dynamical behavior and applications, in: Mathematics and its Applications, Kluwer Academic Publishers, Dordrecht, 1990.
[5] R. Laubenbacher, B. Pareigis, G. Voutsadakis, Categorical ties between TANs and SDSs (preprint).
[6] C.L. Liu, Introduction to Combinatorial Mathematics, McGraw-Hill, New York, 1968.
[7] H. Mortveit, Sequential dynamical systems, Ph.D. Thesis, Department of Mathematical Sciences, Norwegian University of Science and Technology, April 2000.
[8] G. Voutsadakis, A categorical approach to threshold agent networks (preprint).