53rd IEEE Conference on Decision and Control
December 15-17, 2014. Los Angeles, California, USA

A Formal Methods Approach to Pattern Synthesis in Reaction Diffusion Systems

Ebru Aydin Gol, Ezio Bartocci and Calin Belta

Abstract— **We propose a technique to detect and generate patterns in a network of locally interacting dynamical systems. Central to our approach is a novel spatial superposition logic, whose semantics is defined over the quad-tree of a partitioned image. We show that formulas in this logic can be efficiently learned from positive and negative examples of several types of patterns. We also demonstrate that pattern detection, which is implemented as a model checking algorithm, performs very well for test data sets different from the learning sets. We define a quantitative semantics for the logic and integrate the model checking algorithm with particle swarm optimization in a computational framework for synthesis of parameters leading to desired patterns in reaction-diffusion systems.**

I. INTRODUCTION

From the stripes of a zebra and the spots on a leopard to the filaments (Anabaena) [1], spirals, squares (Thiopedia rosea), and vortex (Paenibacillus) [2] formed by single-cell organisms, patterns can be found everywhere in nature. Pattern formation is at the very origin of morphogenesis and developmental biology, and it is at the core of technologies such as self-assembly, tissue engineering, and amorphous computing. Even though it received a lot of attention from diverse communities such as biology, computer science, and physics, this phenomenon is still not well understood.

Pattern recognition is usually formulated as a machine learning problem, in which patterns are characterized either statistically [3] or through a structural relationship among their features [4]. Despite its success in several application areas [5], pattern recognition still lacks a formal foundation. Can patterns be specified in a formal language with well-defined syntax and semantics? Can we develop algorithms for pattern detection from specification given in such a language? Given a collection of locally interacting agents, can we design parameter synthesis rules, control and interaction strategies guaranteeing the emergence of global patterns? In this paper, by drawing inspiration from model checking [6], we provide partial answers to these questions.

We address the following problem: *Given a network of locally interacting dynamical systems, and given sets of positive and negative examples of a desired pattern, find parameter values that guarantee the occurrence of the pattern in the network at steady state.* Our approach is based on a novel spatial superposition logic, called *Tree Spatial Superposition Logic* (TSSL), whose semantics is defined over quad-trees of partitioned images. The decision of whether a pattern exists in an image becomes a model checking problem. A pattern descriptor is a TSSL formula, and we employ machine-learning techniques to infer such a formula from the given positive and negative examples of the pattern. To synthesize parameters of the original networked system leading to a desired pattern, we use a particle swarm optimization (PSO) algorithm. The optimization fitness function is given by a measure of satisfaction induced by the quantitative semantics. We present examples showing that formulas in the proposed logic are good classifiers for some commonly encountered patterns. While the overall algorithm can be applied to any network of locally interacting systems, in this paper we focus on the Turing reaction-diffusion system [7], and show that pattern-producing parameters can be automatically generated with our method.

Ebru Aydin Gol (ebru@bu.edu) and Calin Belta (cbelta@bu.edu) are with Boston University. Ezio Bartocci (ezio.bartocci@tuwien.ac.at) is with Vienna University of Technology.

II. RELATED WORK

Pattern recognition is a well-established technique in machine learning. Given a data set and a set of classes, the goal is to assign each data to one class, or to provide a "most likely" matching of the data to the classes. The two main steps in pattern recognition are: (a) to extract distinctive features [8], [9] with relevant information from a set of input data representing the pattern of interest and (b) to build, using one of the several available machine learning techniques (see [10] for a detailed survey), an accurate classifier trained with the extracted features.

In this paper we establish an interesting connection between verification and pattern recognition. Both classical verification [11], [12] and pattern recognition techniques aim to verify (and possibly quantify) the emergence of a behavioral pattern. We propose logic formulas as pattern descriptors and verification techniques as pattern classifiers. The logical nature of such pattern descriptors allows to reason about the patterns and to infer interesting properties.

This paper is inspired by the original work on morphogenesis by Alan Turing [7], and is closely related to [13]. In the latter, the authors introduced a Linear Spatial Superposition Logic (LSSL), whose formulas were interpreted, as in this paper, over quad-tree partitions. The existence of a pattern in an image corresponded to the existence of a path in the corresponding tree from the root to the leaf corresponding to a representative point in the image. As a consequence, the method was shown to work for spirals, for which the center was chosen as the representative point. The tree logic proposed here is more general as it does not depend on the choice of such a point and captures the pattern "globally". For example, the patterns considered in this paper cannot be

978-1-4673-6090-6/14/$31.00 ©2014 IEEE

expressed in LSSL, because they rely on a tree representation rather than a path representation.

As opposed to [13], we define a quantitative semantics for the logic, and use the distance to satisfaction as a fitness function while searching for pattern-producing parameters. This quantitative semantics and the discounted model checking on a computational tree are inspired from [14], with the notable difference that we do not need a metric distance, but rather a measure of satisfiability. The main novelty of this paper, compared to the other pattern recognition approaches, is that we can quantify "how far" a system is from producing a desired pattern. This, which is possible due to the quantitative semantics of our logic, enables the use of optimization algorithms to search for pattern-producing parameters. Due to space limitations, the results in this paper are stated without proofs. The proofs and additional details can be found in [15].

III. PROBLEM FORMULATION

Notation. We use $\mathbb{R}$, $\mathbb{R}_+$, $\mathbb{N}$ and $\mathbb{N}_+$ to denote the set of real numbers, non-negative reals, integer numbers, and non-negative integers, respectively. For any $a,b,c \in \mathbb{R}$ and set $\mathcal{S} \subseteq \mathbb{R}$, $\mathcal{S}_{>c} := \{x \in \mathcal{S} \mid x > c\}$, and $\mathcal{S}_{[a,b]} := \{x \in \mathcal{S} \mid a \leq x \leq b\}$.

A reaction-diffusion system $\mathbf{S}$ is modeled as a spatially distributed and locally interacting $K \times K$ rectangular grid of identical systems, where each location $(i,j) \in \mathbb{N}_{[1,K]} \times \mathbb{N}_{[1,K]}$ corresponds to a system:

$$S_{i,j} : \frac{dx_{i,j}^{(n)}}{dt} = D_n(u_{i,j}^{(n)} - x_{i,j}^{(n)}) + f_n(\mathbf{x}_{i,j}, \mathbf{R}), \quad n = 1,\ldots,N, \tag{III.1}$$

where $\mathbf{x}_{i,j} = [x_{i,j}^{(1)},\ldots,x_{i,j}^{(N)}]$ is the state vector of system $S_{i,j}$, which captures the concentrations of all species of interest. $\mathbf{D}$ and $\mathbf{R}$ are the parameters of system $\mathbf{S}$. $\mathbf{D} = [D_1,\ldots,D_N] \in \mathbb{R}_+^N$ is the vector of diffusion coefficients. $\mathbf{R} \in \mathbb{R}^{P-N}$ is the vector of parameters that defines the local dynamics $f_n : \mathbb{R}_+^N \times \mathbb{R}^{P-N} \to \mathbb{R}$ for each of the species $n = 1,\ldots,N$. Note that the parameters and dynamics are the same for all systems $S_{i,j}, (i,j) \in \mathbb{N}_{[1,K]} \times \mathbb{N}_{[1,K]}$. The diffusion coefficient is strictly positive for diffusible species and it is 0 for non-diffusible species. Finally, $\mathbf{u}_{i,j} = [u_{i,j}^{(1)},\ldots,u_{i,j}^{(N)}]$ is the input of system $S_{i,j}$ from the neighboring systems:

$$u_{i,j}^{(n)} = \frac{1}{|\nu_{i,j}|} \sum_{v \in \nu_{i,j}} x_v^{(n)}, \text{ where}$$

$\nu_{i,j}$ denotes the set of indices of systems adjacent to $S_{i,j}$.

Given a parameter vector $\mathbf{p} = [D,R] \in \mathbb{R}^P$, we use $\mathbf{S}^{(\mathbf{p})}$ to denote an instantiation of a reaction-diffusion system. We use $\mathbf{x}(t) \in \mathbb{R}_+^{K \times K \times N}$ to denote the state of system $\mathbf{S}^{(\mathbf{p})}$ at time t, and $\mathbf{x}_{i,j}(t) \in \mathbb{R}_+^N$ to denote the state of system $S_{i,j}^{(\mathbf{p})}$ at time t. While the model captures the dynamics of concentrations of all species of interest, we assume that a subset $\{n_1,\ldots,n_o\} \subseteq \{1,\ldots,N\}$ of the species is observable through:

$$H : \mathbb{R}_+^{K \times K \times N} \to \mathbb{R}_{[0,b]}^{K \times K \times o} : \qquad \mathbf{y} = H(\mathbf{x}),$$

for some $b \in \mathbb{R}_+$.

We are interested in analyzing the observations generated by system (III.1) in steady state. Therefore, we focus on parameters that generate steady state behavior, which can be easily checked through a running average:

$$\sum_{i=1}^{K}\sum_{j=1}^{K}\sum_{n=1}^{N} \mid x_{i,j}^{(n)}(t) - x_{i,j}^{(n)} \mid < \varepsilon, \tag{III.2}$$

where $x_{i,j}^{(n)} = \int_{t-T}^{T} x_{i,j}^{(n)}(\tau)d\tau/T$ for some $T \leq t$. The system is said to be in steady state at time $\bar{t}$, if (III.2) holds for all $t \geq \bar{t}$. In the rest of the paper, we will simply call the observation of a trajectory at steady state as the *observation* of the trajectory, and denote it as $H(\mathbf{x}(\bar{t}))$.

Example 3.1: We consider a 32×32 reaction-diffusion system with two species (*i.e.* $K = 32$, $N = 2$):

$$\begin{aligned} \frac{dx_{i,j}^{(1)}}{dt} &= D_1\left(u_{i,j}^{(1)} - x_{i,j}^{(1)}\right) + R_1 x_{i,j}^{(1)} x_{i,j}^{(2)} - x_{i,j}^{(1)} + R_2, \\ \frac{dx_{i,j}^{(2)}}{dt} &= D_2\left(u_{i,j}^{(2)} - x_{i,j}^{(2)}\right) + R_3 x_{i,j}^{(1)} x_{i,j}^{(2)} + R_4. \end{aligned} \tag{III.3}$$

The system is inspired from Turing's reaction-diffusion system and is presented in [16] as a model of the skin pigments of an animal. At a cell (location (i,j)), the concentration of species 1, $x_{i,j}^{(1)}$, depends on the concentration of species 1 in this cell and in its neighbors (if $D_1 > 0$), and the concentration of species 2 in this cell only, *i.e.* $x_{i,j}^{(2)}$. Similarly, $x_{i,j}^{(2)}$ depends on the concentration of species 2 in this cell and in its neighbors (if $D_2 > 0$), and $x_{i,j}^{(1)}$ (if $R_3 \neq 0$). We assume that species 1 is observable through mapping $H : \mathbb{R}_+^{32\times32\times2} \to \mathbb{R}_{[0,1]}^{32\times32}$ given by:

$$\mathbf{y} = H(\mathbf{x}), \text{ where } y_{i,j} = \frac{\mathbf{x}_{i,j}^{(1)}}{\max_{m,n} \mathbf{x}_{m,n}^{(1)}}.$$

We simulate the system from random initial conditions with parameters $\mathbf{R} = [1,-12,-1,16]$, and different diffusion parameters $\mathbf{D}_1 = [5.6,24.5]$, $\mathbf{D}_2 = [0.2,20]$, and $\mathbf{D}_3 = [1.4,5.3]$. The observed concentrations of species 1 at different time points are shown in Figure 1. At time $t = 50$, all trajectories are in steady state. Note that, in all three cases, the spatial distribution of the steady state concentrations of species 1 has some regularity, *i.e.* it forms a "pattern". We will use *large spots* (LS), *fine patches* (FP), and *small spots* (SS) to refer to the patterns corresponding to $\mathbf{D}_1$, $\mathbf{D}_2$, and $\mathbf{D}_3$, respectively.

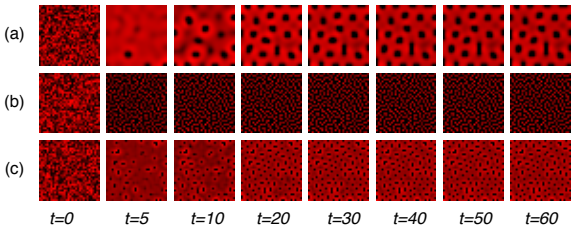

Fig. 1. Observations generated by system (III.3) with parameters $\mathbf{R}$ and (a) $\mathbf{D}_1$, (b) $\mathbf{D}_2$, and (c) $\mathbf{D}_3$ from Example 3.1 (the concentration of species 1 is represented with shades of red). The steady state observations produce (a) *large spots* (LS), (b) *fine patches* (FP), and (c) *small spots* (SS).

Problem 3.1: Given a reaction-diffusion system $\mathbf{S}$ (III.1), a finite set of initial conditions $\mathcal{X}_0 \subset \mathbb{R}^{K\times K\times N}$, ranges of the design parameters $\mathcal{P} = \mathcal{P}_1 \times \ldots \times \mathcal{P}_P$, $\mathcal{P}_i \subset \mathbb{R}, i = 1,\ldots,P$, a set of observations $\mathbf{Y}_+ = \{\mathbf{y}_i\}_{i=1,\ldots,N_+}$ that contain a desired pattern, a set of observations $\mathbf{Y}_- = \{\mathbf{y}_i\}_{i=1,\ldots,N_-}$ that do not contain the pattern, find parameters $\mathbf{p}^* \in \mathcal{P}$ such that the trajectories of system $\mathbf{S}^{(\mathbf{p}^*)}$ originating from $\mathcal{X}_0$ are guaranteed

to produce observations similar to the ones from $\mathbf{Y}_+$.

To solve Problem 3.1, we need to perform two steps: 1) design a mechanism that decides whether an observation contains a pattern, 2) develop a search algorithm over the state space of the design parameters to find $\mathbf{p}^*$.

The first step requires to define a pattern descriptor. To this goal, we develop a new spatial logic over spatial-superposition trees obtained from the observations, and treat the decision problem as a model checking problem. Then, finding a pattern descriptor reduces to finding a formula of the new logic that specifies the desired pattern. We employ machine-learning techniques to learn such a formula from the given sets of observations $\mathbf{Y}_+$ and $\mathbf{Y}_-$.

The second step is the synthesis of parameters $\mathbf{p}^*$ such that the observations produced by the corresponding reaction-diffusion system $\mathbf{S}^{(\mathbf{p}^*)}$ satisfy the formula learned in the first step. To this end, we introduce quantitative semantics for the new logic, which assigns a positive valuation only to the superposition-trees that satisfy the formula. This quantitative valuation is treated as a measure of satisfaction, and is used as the fitness function in a particle swarm optimization (PSO) algorithm. Finally, we propose a supervised, iterative procedure to find $\mathbf{p}^*$ that solves Problem 3.1. The procedure involves iterative applications of steps one and two, and an update of the set $\mathbf{Y}_-$ until a parameter set that solves Problem 3.1 is found, which is decided by the user.

IV. TREE SPATIAL SUPERPOSITION LOGIC

A. Quad-tree spatial representation

We represent the observations of a reaction-diffusion system as a matrix $\mathcal{A}_{k,k}$ of $2^k \times 2^k$ elements $a_{i,j}$ with $k \in \mathbb{N}_{>0}$. Each element corresponds to a small region in the space and is defined as a tuple $a_{i,j} = \langle a_{i,j}^{(1)}, \cdots, a_{i,j}^{(o)} \rangle$ of values representing the concentration of the observable species within an interval $a_{i,j}^{(c)} \in [0,b]$, with $b \in \mathbb{R}_+$. Given a matrix $\mathcal{A}_{k,k}$, we use $\mathcal{A}_{k,k}[i_s, i_e; j_s, j_e]$ to denote the sub-matrix formed by selecting the rows with indices from i_s to i_e and the columns with indices from j_s to j_e.

Definition 4.1: A quad-tree $Q = (V,R)$ is a quaternary tree [17] representation of $\mathcal{A}_{k,k}$ where each vertex $v \in V$ represents a sub-matrix of $\mathcal{A}_{k,k}$ and the relation $R \subset V \times V$ defines the four children of each node v that is not a leaf. A vertex v is a leaf when all the elements of the sub-matrix that it represents have the same values.

Figure 2 shows an example of a quadtree, where node v_0 represents the entire matrix; child v_1 represents the sub-matrix $\{1,\cdots,2^{k-1}\} \times \{1,\cdots,2^{k-1}\}$. In Figure 2, we label each edge in the quad-tree with the direction of the sub-matrix represented by the child: north west (NW), north east (NE), south west (SW), south east (SE).

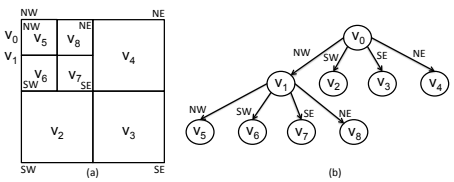

Fig. 2. Quad-tree representation (b) of a matrix (a).

Definition 4.2: We define the mean function $\mu_c : V \to [0,b]$ for sub-matrix $\mathcal{A}_{k,k}[i_s, i_e; j_s, j_e]$ represented by the vertex $v \in V$ of the quad-tree $Q = (V,R)$ as follows:

$$\mu_c(v) = \frac{1}{(i_e - i_s + 1)(j_e - j_s + 1)} \sum_{i,j \in \{i_s,\cdots,i_e\} \times \{j_s,\cdots,j_e\}} a_{i,j}^{(c)}$$

The function μ_c provides the expected value for an observable variable with index $c, 1 \le c \le o$ in a particular region of the space represented by the vertex v.

Algorithm BUILDINGQUADTRANSITIONSYSTEM

Input: Matrix $\mathcal{A}_{k,k}$ of $2^k \times 2^k$ of elements $a_{i,j} = \langle a_{i,j}^{(1)}, \cdots, a_{i,j}^{(o)} \rangle$, its quad-tree $Q = (V,R)$, the root $v_0 \in V$, and a labeling function $LQ : R \to \mathcal{D} = \{NW, NE, SE, SW\}$

Output: Quad Transition System $Q_{TS} = (S, s_1, \tau, \Sigma, [.], L)$

$\Sigma := \{m_1, \cdots, m_o\}$ ▷ Initialize the set of variables Σ of Q_{TS}.
$\tau = \emptyset$ ▷ Initialize the set τ of the transition relation τ of Q_{TS}.
$S := \{s_1\}$ ▷ Initialize the set of states S of Q_{TS}.
$TS := \{\langle s_1, \{v_0\}\rangle\}$
▷ Each tuple in TS contains a state in S and a set of vertices in V.
$LF := \{v \in V \mid \nexists t \in V : (v,t) \in R\}$ ▷ LF is the set of leaves of Q
$PLF := \{P_i \subseteq LF, 1 \le i \le n \mid P_i \neq \emptyset \wedge \forall v_a, v_b \in P_i, \forall v_c \in P_{j \neq i}, v_a \equiv v_b \wedge v_a \not\equiv v_c\}$
▷ PLF is a partition of LF with equivalent leaves.
for each $\hat{P} \in PLF$ **do**
▷ For each partition element, create a state s' with a self-loop and
▷ a transition to the state s_1 if $\hat{P}$ contains a child of v_0.
add new state s' to S and a tuple $\langle s', \hat{P}\rangle$ to TS
$\tau := \tau \cup \{(s',s')\} \cup \{(s,s') : \langle s, VS\rangle \in TS, \exists v \in VS, \exists v' \in \hat{P} : (v,v') \in R\}$
end for
$FS := \{v \in V | (v_0, v) \in R\} \backslash LF$
▷ explore the children of v_0 that are not leaves.
while $FS \neq \emptyset$ **do** ▷ FS contains the frontier vertices to be explored.
$LFS := \{v \in FS \mid \forall v' \in V : (v,v') \in R : \exists \langle s, VS\rangle \in TS \wedge v' \in VS\}$
$PLFS := \{P_{i \in I} \subseteq LFS \mid I \neq \emptyset, P_i \neq \emptyset, \forall v_a, v_b \in P_i, \forall v_c \in P_{j \neq i}, v_a \equiv v_b \wedge v_a \not\equiv v_c\}$
for each $\hat{P} \in PLFS$ **do**
add new state s' to S and a tuple $\langle s', \hat{P}\rangle$ to TS
$\tau := (\bigcup_{s : \langle s, VS\rangle \in TS : \exists v \in \hat{P}, \exists v' \in VS, (v,v') \in R} (s', s)) \cup \tau$
if $\exists v \in \hat{P} \wedge \exists \langle s, VS\rangle : \exists v' \in VS \wedge (v', v) \in R$ **then**
$\tau := \tau \cup \{(s,s')\}$
end if
end for
for each $\hat{v} \in FS \backslash LFS$ **do**
add new state s' to S and a tuple $\langle s', \{\hat{v}\}\rangle$ to TS
$\tau := (\bigcup_{s : \langle s, VS\rangle \in TS : \exists v' \in VS, (\hat{v}, v') \in R} (s', s)) \cup \tau$
if $\exists \langle s, VS\rangle : \exists v' \in VS \wedge (v', \hat{v}) \in R$ **then**
$\tau := \tau \cup \{(s,s')\}$
end if
end for
$FS := \{v \in V \mid \exists \bar{v} \in FS, (\bar{v}, v) \in R\} \backslash LF$
end while
define func $[.]$ as $[\bar{c}](\bar{s}) := \mu_{\bar{c}}(v_{\bar{s}})$, $\bar{c} \in \{1, \cdots, o\}$, $v_{\bar{s}} \in VS : \langle \bar{s}, VS\rangle \in TS$
define func L as $L(s,t) := (t = s)?\mathcal{D} : \bigcup_{\bar{v} \in \bar{VS}, \bar{v} \in \bar{VT} : \langle s, \bar{VS}\rangle, \langle t, \bar{VT}\rangle \in TS, (\bar{v}, \bar{v}) \in R} LQ(\bar{v}, \bar{v})$
return $S, s_1, \tau, \Sigma, [.], L$

Definition 4.3: Two vertices $v_a, v_b \in V$ are said to be equivalent when the mean function applied to the elements of the sub-matrices that they represent produce the same values:

$$v_a \equiv v_b \iff \mu_c(v_a) = \mu_c(v_b), \forall c, 1 \le c \le o$$

We use the mean of the concentration of the observable species as a spatial abstraction (superposition) of the observations in a particular region of the system, avoiding in this way to enumerate the observations of all locations.

Proposition 4.1: Given a vertex $v \in V$ of a quad-tree $Q = (V,R)$ and its four children $v_{NE}, v_{NW}, v_{SE}, v_{SW}$ the following

property holds:

$$\mu_c(v) = \frac{\mu_c(v_{NE}) + \mu_c(v_{NW}) + \mu_c(v_{SE}) + \mu_c(v_{SW})}{4}.$$

Proposition 4.2: The number of vertices needed for the quad-tree representation $Q = (V,R)$ of a matrix $\mathcal{A}_{k,k}$ is upper bounded by $\sum_{i=0}^{k} 2^{2i}$.

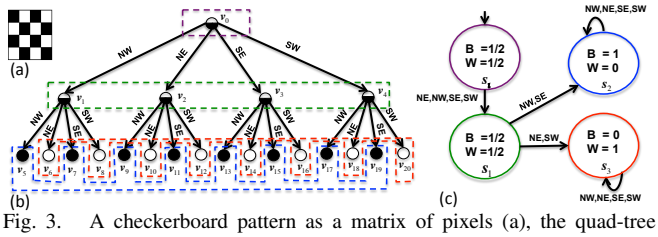

Fig. 3. A checkerboard pattern as a matrix of pixels (a), the quad-tree representation (b) and the derived quad transition system (c), where **B** and **W** denote black and white, respectively.

B. Quad Transition System

Definition 4.4: A *Quad Transition System* (QTS) is a tuple $Q_{TS} = (S, s_\iota, \tau, \Sigma, [.], L)$, where S is a finite set of states with $s_\iota \in S$ the initial state, $\tau \subseteq S \times S$ is the transition relation, Σ is a finite set of variables, $[.]$ is a function $[.] : S \to (\Sigma \to [0,b])$ that assigns to each state $s \in S$ and a variable $m \in \Sigma$ a rational value $[s](m)$ in $[0,b]$ with $b \in \mathbb{R}_+$, L is a labeling function for the transition $L : \tau \to 2^{\mathcal{D}}$ with $\mathcal{D} = \{NW, NE, SE, SW\}$ and with the property that $\forall (s,t),(s,t') \in \tau$, with $t \neq t'$ it holds that $L(s,t) \cap L(s,t') = \emptyset$, $\bigcup_{\forall t \in S:(s,t)\in\tau} L(s,t) = \mathcal{D}$. We require τ to be non-blocking and bounded-branching: $\forall s \in S, \exists t \in S : (s,t) \in \tau$ and $\forall s \in S$, if $T(s) = \{t : (s,t) \in \tau\}$ is the set of all successors of s, the cardinality of $|T(s)| \leq 4$.

The BUILDINGQUADTRANSITIONSYSTEM algorithm shows how to generate a QTS starting from a quad-tree representation $Q = (V,R)$ of a a matrix $\mathcal{A}_{k,k}$ and a labeling function $LQ : R \to \mathcal{D}$.

Proposition 4.3: A *quad transition system* (QTS) $Q_{TS} = (S, s_\iota, \tau, \Sigma, [.], L)$ generated by the BUILDINGQUADTRANSITIONSYSTEM algorithm has always a least fixed point, that is $\exists s \in S : (s,s) \in \tau$.

Definition 4.5 (Labeled paths): Given a set B of labels representing the spatial directions, a *labeled path* (lpath) of a QTS Q is an infinite sequence $\pi^B = s_0 s_1 s_2 \cdots$ of states such that $(s_i, s_{i+1}) \in \tau \wedge L(s_i, s_{i+1}) \cap B \neq \emptyset$, $\forall i \in \mathbb{N}$. Given a state s, we denote $LPaths^B(s)$ the set of all labeled paths starting in s, and with π_i^B the i-th element of a path $\pi^B \in LPaths^B(s)$. For example, in Figure 3, $LPaths^B(s_\iota) = \{s_\iota s_1 s_2 s_2 \cdots\}$ if $B = \{NW, SE\}$.

C. TSSL Syntax and Semantics

Definition 4.6 (TSSL syntax): The syntax of TSSL is defined as follows:

$$\varphi ::= \top \mid \bot \mid m \sim d \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \exists_B \bigcirc \varphi \mid \forall_B \bigcirc \varphi \mid \exists_B \varphi_1 \; \mathcal{U}_k \; \varphi_2 \mid \forall_B \varphi_1 \; \mathcal{U}_k \; \varphi_2$$

with $\sim \in \{\leq, \geq\}$, $d \in [0,b]$, $b \in \mathbb{R}_+$, $k \in \mathbb{N}_{>0}$, $B \subseteq \mathcal{D} : B \neq \emptyset$, and $m \in \Sigma$, with Σ the set of variables.

From this basic syntax one can derive other two temporal operators: the *exist eventually* operator $\exists_B F_k$, the *forall eventually* operator $\forall_B F_k$, the *exist globally* operator $\exists_B G_k$, and the *forall globally* operator $\forall_B G_k$ defined such that:

$$\exists_B F_k \varphi := \exists_B \top \; \mathcal{U}_k \; \varphi \qquad \exists_B G_k \varphi := \neg \forall_B F_k \neg\varphi.$$
$$\forall_B F_k \varphi := \forall_B \top \; \mathcal{U}_k \; \varphi \qquad \forall_B G_k \varphi := \neg \exists_B F_k \neg\varphi.$$

The TSSL logic resembles the classic CTL logic [18], with the main difference that the *next* and *until* are not temporal, but spatial operators meaning a change of resolution (or zoom in). The set B selects the spatial directions in which the operator is allowed to work and the parameter k limits the *until* to operate on a finite sequence of states. In the following we provide the TSSL qualitative semantics that, given a spatial model and a formula representing the pattern to detect, provides a yes/no answer.

Definition 4.7 (TSSL Qualitative Semantics): Let $Q = (S, s_\iota, \tau, \Sigma, [.], L)$ be a QTS, Then, Q *satisfies a TSSL formula* φ, written $Q \models \varphi$, if and only if $Q, s_\iota \models \varphi$, where:

$Q,s \models \top$	and	$Q,s \not\models \bot$
$Q,s \models m \sim d$	$\Leftrightarrow$	$[s](m) \sim d$
$Q,s \models \neg\varphi$	$\Leftrightarrow$	$Q,s \not\models \varphi$
$Q,s \models \varphi_1 \wedge \varphi_2$	$\Leftrightarrow$	$Q,s \models \varphi_1 \wedge Q,s \models \varphi_2$
$Q,s \models \exists_B \bigcirc \varphi$	$\Leftrightarrow$	$\exists s' : (s,s') \in \tau \wedge L(s,s') \cap B \neq \emptyset \wedge Q,s' \models \varphi$
$Q,s \models \forall_B \bigcirc \varphi$	$\Leftrightarrow$	$\forall s' : (s,s') \in \tau \wedge L(s,s') \cap B \neq \emptyset \wedge Q,s' \models \varphi$
$Q,s \models \exists_B \varphi_1 \; \mathcal{U}_k \; \varphi_2$	$\Leftrightarrow$	$\exists \pi^B \in LPaths^B(s) : \exists i, 0 < i \leq k :$ $(Q,\pi_i^B \models \varphi_2) \wedge (\forall j < i, (Q,\pi_j \models \varphi_1))$
$Q,s \models \forall_B \varphi_1 \; \mathcal{U}_k \; \varphi_2$	$\Leftrightarrow$	$\forall \pi^B \in LPaths^B(s) : \exists i, 0 < i \leq k :$ $(Q,\pi_i^B \models \varphi_2) \wedge (\forall j < i, (Q,\pi_j \models \varphi_1))$

Example 4.1: Checkerboard pattern. The checkerboard pattern from Fig 3 a) can be characterized with the following TSSL formula ($B^* = \{SW, NE, NW, SE\}$):

$$\forall_{B^*} \bigcirc (\forall_{B^*} \bigcirc ((\forall_{\{SW,NE\}} \bigcirc (m \geq 1)) \wedge (\forall_{\{NW,SE\}} \bigcirc (m \leq 0)))).$$

The "eventually" operator can be used to define all the possible checkerboards of different sizes less or equal than 4^2 as follows:

$$\forall_{B^*} F_2((\forall_{\{SW,NE\}} \bigcirc (m \geq 1)) \wedge (\forall_{\{NW,SE\}} \bigcirc (m \leq 0)))$$

The qualitative semantics is useful to check if a given spatial model violates or satisfies a pattern expressed in TSSL. However, it does not provide any information about how much the property is violated or satisfied. This information may be useful to guide a simulation-based parameter exploration for pattern generation. For this reason we equip our logic also with a quantitative valuation that provides a measure of satisfiability in the same spirit of [11]. Since the valuation of a TSSL formula with spatial operators requires to traverse and to compare regions of space at different resolution, we apply a discount factor of $\frac{1}{4}$ on the result each time a transition is taken in QTS.

Definition 4.8 (TSSL Quantitative Semantics): Let $Q = (S, s_\iota, \tau, \Sigma, [.], L)$ be a QTS. The quantitative valuation $[\![\varphi]\!] : S \to [-b, b]$ of a TSSL formula φ is defined as follows:

$$[\![\top]\!](s) = b \wedge [\![\bot]\!](s) = -b$$
$$[\![m \sim d]\!](s) = (\sim \text{is} \geq) \; ? \; ([m](s) - d) : (d - [m](s))$$
$$[\![\neg\varphi]\!](s) = -[\![\varphi]\!](s)$$
$$[\![\varphi_1 \wedge \varphi_2]\!](s) = \min([\![\varphi_1]\!](s), [\![\varphi_2]\!](s))$$
$$[\![\exists_B \bigcirc \varphi]\!](s) = \frac{1}{4} \max_{\pi^B \in LPaths^B(s)} [\![\varphi]\!](\pi_1^B)$$
$$[\![\forall_B \bigcirc \varphi]\!](s) = \frac{1}{4} \min_{\pi^B \in LPaths^B(s)} [\![\varphi]\!](\pi_1^B)$$
$$[\![\exists_B \varphi_1 \; \mathcal{U}_k \; \varphi_2]\!](s) = \sup_{\pi^B \in LPaths^B(s)} \{\min(\frac{1}{4^i} [\![\varphi_2]\!](\pi_i^B), \inf\{\frac{1}{4^j} [\![\varphi_1]\!](\pi_j^B) \mid j < i\}) \mid 0 < i \leq k\}\}$$
$$[\![\forall_B \varphi_1 \; \mathcal{U}_k \; \varphi_2]\!](s) = \inf_{\pi^B \in LPaths^B(s)} \{\min(\frac{1}{4^i} [\![\varphi_2]\!](\pi_i^B), \inf\{\frac{1}{4^j} [\![\varphi_1]\!](\pi_j^B) \mid j < i\}) \mid 0 < i \leq k\}$$

Theorem 4.1 (Soundness): Let $Q = (S, s_\iota, \tau, \Sigma, [.], L)$ be a QTS, $s \in S$ a state of Q, and φ a TSSL formula. Then, the following properties hold for the two semantics:

$$[\![\varphi]\!](s) > 0 \Longrightarrow Q, s \models \varphi$$

$$[\![\varphi]\!](s) < 0 \Longrightarrow Q, s \not\models \varphi$$

Remark 4.1: Theorem 4.1 provides the basis of the techniques for pattern generation discussed in the following sections. It is worth to note that, in the case $[\![\varphi]\!](s) = 0$, it is not possible to infer whether Q violates or satisfies a TSSL formula φ and only in this particular case we need to resort to the qualitative semantics for determining it.

V. TSSL PATTERN CLASSIFIERS

A QTS can be seen in the context of multi-resolution representation, since the nodes that appear at deeper levels provide information for higher resolutions. Therefore, a TSSL formula can effectively capture properties of an image. However, it is difficult to write a formula that describes a desired property, such as a pattern. Here, we propose to use machine-learning techniques to find such a formula from given sets of positive ($\mathbf{Y}_+$) and negative ($\mathbf{Y}_-$) examples.

We first define a labeled data set from the given data sets $\mathbf{Y}_+$ and $\mathbf{Y}_-$ as

$$\mathcal{L} = \{(Q_{\mathbf{y}}, +) \mid \mathbf{y} \in Y_+\} \cup \{(Q_{\mathbf{y}}, -) \mid \mathbf{y} \in Y_-\},$$

where $Q_{\mathbf{y}}$ is the QTS generated from $\mathbf{y}$. Then, we separate the data set $\mathcal{L}$ into disjoint training and testing sets $\mathcal{L}_L, \mathcal{L}_T$. We employ RIPPER [19], a rule based learner, to learn a classifier from $\mathcal{L}_L$, and then translate the classifier into a TSSL formula characterizing $+$. Each rule obtained from the learning algorithm is described as $r_i : C_i \Rightarrow \sim_i$, where C_i is a boolean formula over linear predicates over the variables of the states of a QTS, *e.g.* $[s](m) > d$, and $\sim_i$ takes values from the label set $\{+, -\}$. A linear predicate for a state $s \in S$ can be written as a TSSL formula via the QTS path from the root s_ι to s. Therefore, each C_i can be translated into an equivalent TSSL formula Φ_i. The classification rules are interpreted as nested if-else statements. Hence, a logically equivalent TSSL formula for the desired property is defined as follows:

$$\Phi_+ := \bigvee_{j \in R_+} \left(\Phi_j \wedge \bigwedge_{i=1,\ldots,j-1} \neg \Phi_i \right), \tag{V.4}$$

where R_+ is the set of indices of rules r_i with $\sim_i = +$, and Φ_i is the TSSL formula obtained from C_i.

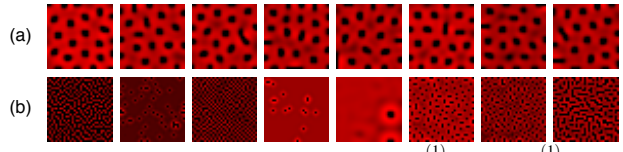

Fig. 4. Sample sets of images from the sets (a) $\mathbf{Y}_+^{(1)}$ and (b) $\mathbf{Y}_-^{(1)}$ for the LS pattern.

Example 5.1: LS pattern. For the LS pattern from Example 3.1, we generate a data set $\mathbf{Y}_+^{(1)}$ containing 8000 positive examples by simulating the reaction-diffusion system (III.3) from random initial conditions with parameters $\mathbf{R}$ and $\mathbf{D}_1$. Similarly, to generate the data set $\mathbf{Y}_-^{(1)}$ containing 8000 negative examples, we simulate system (III.3) from random initial conditions. However, in this case we use $\mathbf{R}$ and randomly choose the diffusion coefficients from $\mathbb{R}^2_{[0,30]}$. A sample set of images from the sets $\mathbf{Y}_+^{(1)}$ and $\mathbf{Y}_-^{(1)}$ is shown in Figure 4. We generate a labeled set $\mathcal{L}^{(1)}$ of QTS from these sets, and separate $\mathcal{L}^{(1)}$ into $\mathcal{L}_L^{(1)}, \mathcal{L}_T^{(1)}$. We use RIPPER algorithm implemented in Weka [20] to learn a classifier from $\mathcal{L}_L^{(1)}$. The learning step took 228.5*sec* on an iMac with a Intel Core i5 processor at 2.8GHz with 8GB of memory. The classifier consists of 24 rules. The first rule is

$$r_1 : (R \geq 0.59) \wedge (R \leq 0.70) \wedge (R.NW.NW.NW.SE \leq 0.75) \wedge (R.NW.NW.NW.NW \geq 0.45) \Rightarrow +,$$

where R denotes the root of a QTS, and the labels of the children are explained in Figure 2. Rule r_1 translates to the following TSSL formula:

$$\Phi_1 : (m \geq 0.59) \wedge (m \leq 0.70) \wedge (\exists_{NW} \bigcirc \exists_{NW} \bigcirc \exists_{NW} \bigcirc \exists_{SE} \bigcirc m \geq 0.75) \wedge (\exists_{NW} \bigcirc \exists_{NW} \bigcirc \exists_{NW} \bigcirc \exists_{NW} \bigcirc m \geq 0.45).$$

We define the TSSL formula $\Phi_+^{(1)}$ characterizing the pattern as in (V.4), and model check QTSs from $\mathcal{L}_T^{(1)}$ ($|\mathcal{L}_T^{(1)}| = 8000$) against $\Phi_+^{(1)}$, which yields a high prediction accuracy (96.11%) with 311 miss-classified QTSs.

FP and SS patterns. We follow the above explained steps to generate data sets $\mathbf{Y}_+^{(i)}, \mathbf{Y}_-^{(i)}$, generate labeled data sets $\mathcal{L}_L^{(i)}, \mathcal{L}_T^{(i)}$, and finally learn formulas $\Phi_+^{(i)}$ for the FP and SS patterns corresponding to diffusion coefficient vectors $\mathbf{D}_i$, $i = 2, 3$ from Example 3.1. The model checking of the QTSs from the corresponding test sets yields high prediction accuracies 98.01%, and 93.13% for $\Phi_+^{(2)}$, and $\Phi_+^{(3)}$, respectively.

VI. PARAMETER SYNTHESIS FOR PATTERN GENERATION

We slightly abuse the terminology and say that a trajectory $\mathbf{x}(t), t \geq 0$ of system $\mathbf{S}^{(\mathbf{p})}$ satisfies Φ if the QTS $Q = (S, s_\iota, \tau, \Sigma, [.], L)$ of the corresponding observation, $H(\mathbf{x}(\bar{t}))$, satisfies Φ, *i.e* $Q \models \Phi$, or $[\![\Phi]\!](s_\iota) > 0$. We define an induced quantitative valuation of a system $\mathbf{S}^{(\mathbf{p})}$ and a set of initial conditions $\mathcal{X}_0$ from a TSSL formula Φ as:

$$[\![\Phi]\!](\mathbf{S}^{(\mathbf{p})}) = \min_{x_0 \in \mathcal{X}_0} \{[\![\Phi]\!](s_\iota) \mid Q = (S, s_\iota, \tau, \Sigma, [.], L) \text{ is QTS of } H(\mathbf{x}(\bar{t}))\} \tag{VI.5}$$

The definition of the induced valuation of a system $\mathbf{S}^{(\mathbf{p})}$ implies that all trajectories of $\mathbf{S}^{(\mathbf{p})}$ originating from $\mathcal{X}_0$ satisfy Φ if $[\![\Phi]\!](\mathbf{S}^{(\mathbf{p})}) > 0$. Therefore, it is sufficient to find $\mathbf{p}$ that maximizes (VI.5). It is assumed that the ranges $\mathcal{P} = \mathcal{P}_1 \times \ldots \times \mathcal{P}_P$ of the design parameters are known. Therefore, the parameters maximizing (VI.5) can be found with a greedy search on a quantization of $\mathcal{P}$. However, the computation of $[\![\Phi]\!](\mathbf{S}^{(\mathbf{p})})$ for a given $\mathbf{p} \in \mathcal{P}$ is expensive, since it requires to perform the following steps for each $x_0 \in \mathcal{X}_0$: simulating the system $\mathbf{S}^{(\mathbf{p})}$ from x_0, generating QTS Q of the corresponding observation, and quantitative model checking of Q against Φ. Here, we use the particle swarm optimization (PSO) algorithm [21] over $\mathcal{P}$ with (VI.5) as the fitness function.

Example 6.1: LS pattern. We consider the reaction-diffusion system from Example 3.1 and the TSSL formula $\Phi_+^{(1)}$ corresponding to the LS pattern from Example 5.1. We assume that the parameters of the local dynamics are known, $\mathbf{R} = [1, -12, -1, 16]$, and the diffusion coefficients D_1 and D_2 are set as the design parameters with $\mathcal{P} = \mathbb{R}^2_{[0,30]}$. We implement PSO to find $\mathbf{p} \in \mathcal{P}$ maximizing the induced valuation (VI.5). The PSO computation was distributed on 16

processors at 2.1GHz on a cluster, and the running time was around 18 minutes. The optimized parameters are $D_1 = 2.25$ and $D_2 = 29.42$, and the valuation of the system is 0.0023. A set of observations obtained by simulating $\mathbf{S}^{([2.25,29.42])}$ is shown in Figure 6-(a). Note that, while all the observations have some spatial periodicity indicating the presence of a pattern, they are still different from the desired LS pattern.

FP and SS patterns. We apply the PSO algorithm on the same setting explained above to maximize the induced valuation (VI.5) for the TSSL formulas $\Phi_+^{(2)}$ (FP) and $\Phi_+^{(3)}$ (SS) from Example 5.1. The optimized parameters are $[0.083, 11.58]$ and $[1.75, 7.75]$ for $\Phi_+^{(2)}$ and $\Phi_+^{(3)}$, respectively. Sets of observations obtained by simulating systems $\mathbf{S}^{([0.083,11.58])}$ and $\mathbf{S}^{([1.75,7.75])}$ are shown in Figure 5. In contrast with the LS pattern, the observations are similar to the ones from the corresponding data sets *i.e.* $\mathbf{Y}_+^{(2)}$ and $\mathbf{Y}_+^{(3)}$.

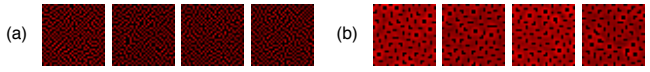

Fig. 5. Sample set of observations obtained by simulating (a) $\mathbf{S}^{([0.083,11.58])}$ and (b) $\mathbf{S}^{([1.75,7.75])}$.

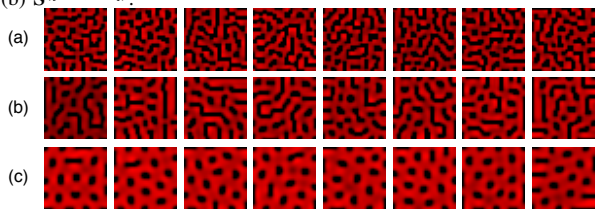

Fig. 6. Sample set of observations obtained by simulating (a) $\mathbf{S}^{([2.25,29.42])}$, (b) $\mathbf{S}^{([3.75,28.75])}$, and (c) $\mathbf{S}^{([6.25,29.42])}$.

As seen in Example 6.1, it is possible that simulations of the system corresponding to optimized parameters do not necessarily lead to desired patterns. This should not be unexpected, as the formula reflects the original training set of positive and negative examples, and was not "aware" that these new simulations are not good patterns. A natural extension of our method should allow to add the newly obtained simulations to the negative training set, and to reiterate the whole procedure as described below.

We start with the user defined sets of observations $\mathbf{Y}_+$ and $\mathbf{Y}_-$, and learn a TSSL formula Φ from the QTS representations of the observations (Section V). Then, in the optimization step, we find a set of parameters $\mathbf{p}$ that maximizes $\gamma = [\![\Phi]\!](\mathbf{S}^{(\mathbf{p})})$. If $\gamma < 0$, then we terminate the procedure as parameters producing observations similar to the ones from the set $\mathbf{Y}_+$ with respect to the TSSL formula Φ could not be found. If $\gamma \geq 0$, then the observations of system $\mathbf{S}^{(\mathbf{p})}$ satisfy Φ. Finally, the user inspects the observations generated from the reaction-diffusion system with the optimized set of parameters $\mathbf{S}^{(\mathbf{p})}$. If the observations are similar to the ones from the set $\mathbf{Y}_+$, then we find a solution. If, however, the user decides that the observations do not contain the pattern, then we add observations obtained from system $\mathbf{S}^{(\mathbf{p})}$ to $\mathbf{Y}_-$, and repeat the process, *i.e* learn a new formula, run the optimization until the user terminates the process or the optimization step fails ($\gamma < 0$).

Example 6.2: LS pattern. We apply the proposed design procedure to the system from Example 6.1. We decide that observations shown in Figure 6-(a) are not similar to the ones from the set $\mathbf{Y}_+^{(1)}$ shown in Figure 4-(a), and add 250 observations generated with the optimized parameters to $\mathbf{Y}_-^{(1)}$. In the second iteration, the optimized parameters are $D_1 = 3.75$ and $D_2 = 28.75$, and the observations obtained by simulating $\mathbf{S}^{([3.75,28.75])}$ are shown in Figure 6-(b). We continue by adding these to $\mathbf{Y}_-^{(1)}$. The parameters computed in the third iteration are $D_1 = 6.25$ and $D_2 = 29.42$. The observations obtained by simulating $\mathbf{S}^{([6.25,29.42])}$ are shown in Figure 6-(c). Although the optimized parameters are different from $\mathbf{D}_1$, which was used to generate $\mathbf{Y}_+^{(1)}$, the observations of $\mathbf{S}^{([6.25,29.42])}$ are similar to the ones from $\mathbf{Y}_+^{(1)}$.

REFERENCES

[1] J. Golden and H. Yoon, "Heterocyst formation in anabaena," *Curr Opin Microbiol.*, vol. 1, no. 6, pp. 623–629, 1998.

[2] R. Scherrer and V. Shull, "Structure, partial elemental composition, and size of thiopedia rosea cells and platelets," *Can J Microbiol.*, vol. 32, no. 7, pp. 607–610, 1986.

[3] A. Jain, R. Duin, and J. Mao, "Statistical pattern recognition: A review," *IEEE Transactions on pattern analysis and machine learning*, vol. 22, pp. 4–37, 2000.

[4] T. Pavlidis, *Structural Pattern Recognition*. Springer-Verlag, 1980.

[5] R. C. Veltkamp and M. Hagedoorn, "State-of-the-art in shape matching," Principles of Visual Information Retrieval, Tech. Rep., 1999.

[6] E. M. M. Clarke, D. Peled, and O. Grumberg, *Model checking*. MIT Press, 1999.

[7] A. M. Turing, "The chemical basis of morphogenesis," *Philosophical Transactions of the Royal Society of London*, vol. 327, pp. 37–72, 1952.

[8] B. Julesz, "Textons, the elements of texture perception, and their interactions," *Nature*, vol. 290, pp. 91–97, 1981.

[9] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. of CVPR 2005: the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, June 2005, pp. 886–893.

[10] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2002.

[11] A. Donzé, E. Fanchon, L. M. Gattepaille, O. Maler, and P. Tracqui, "Robustness analysis and behavior discrimination in enzymatic reaction networks," *PLoS One*, vol. 6, no. 9, p. e24246, 2011.

[12] G. E. Fainekos and G. J. Pappas, "Robustness of temporal logic specifications for continuous-time signals," *Theor. Comput. Sci.*, vol. 410, no. 42, pp. 4262–4291, 2009.

[13] R. Grosu, S. Smolka, F. Corradini, A. Wasilewska, E. Entcheva, and E. Bartocci, "Learning and detecting emergent behavior in networks of cardiac myocytes," *Communications of the ACM*, vol. 52, no. 3, pp. 97–105, 2009.

[14] L. de Alfaro, M. Faella, and M. Stoeling, "Linear and branching system metrics," *IEEE Transactions on Software Engineering*, vol. 35, no. 2, pp. 258–273, 2009.

[15] E. Aydin Gol, E. Bartocci, and C. Belta, "A formal methods approach to pattern synthesis in reaction diffusion systems," 2014, available at http://arxiv.org/.

[16] R. Collantes, "Algorithm alley. Dr. Dobb's journal," December 1996.

[17] R. Finkel and J. Bentley, "Quad trees a data structure for retrieval on composite keys," *Acta Informatica*, vol. 4, no. 1, pp. 1–9, 1974.

[18] E. M. Clarke and E. A. Emerson, "Design and synthesis of synchronization skeletons using branching-time temporal logic," in *Proc. of Logic of Programs Workshop*, ser. Lecture Notes in Computer Science, vol. 131, 1982, pp. 52–71.

[19] W. W. Cohen, "Fast effective rule induction," in *Proceedings of the Twelfth International Conference on Machine Learning*. Morgan Kaufmann, 1995, pp. 115–123.

[20] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: An update," *SIGKDD Explor. Newsl.*, vol. 11, no. 1, pp. 10–18, Nov. 2009.

[21] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, vol. 4, 1995, pp. 1942–1948.