A Formal Methods Approach to Pattern Recognition and Synthesis in Reaction Diffusion Networks

Ezio Bartocci, Ebru Aydin Gol, *Member, IEEE*, Iman Haghighi, *Student Member, IEEE*, and Calin Belta, *Senior Member, IEEE*

***Abstract*—We introduce a formal framework for specifying, detecting, and generating spatial patterns in reaction diffusion networks. Our approach is based on a novel spatial superposition logic, whose semantics is defined over the quad-tree representation of a partitioned image. We demonstrate how to use rule-based classifiers to efficiently learn spatial superposition logic formulas for several types of patterns from positive and negative examples. We implement pattern detection as a model-checking algorithm and we show that it achieves very good results on test data sets which are different from the training sets. We provide a quantitative semantics for our logic and we develop computational framework where our quantitative model-checking algorithm works in synergy with a particle swarm optimization technique to synthesize the parameters leading to the formation of desired patterns in reaction diffusion networks.**

***Index Terms*—Formal verification and synthesis, pattern recognition and formation, reaction diffusion networks.**

I. Introduction

SPATIAL pattern formation is central to the understanding of how complex organisms develop and how self-organization arises out of locally interacting dynamical systems. Examples of spatial patterns are ubiquitous in nature: from the stripes of a zebra and the spots on a leopard to the filaments (Anabaena) [1], squares (Thiopedia rosea), and vortex (Paenibacillus) [2] formed by single-cell organisms.

Pattern formation is not only at the very origin of morphogenesis and developmental biology, but it is also at the core of technologies, such as self-assembly, tissue engineering, and amorphous computing. Even though the study of spatial patterns has kindled the interest of several communities, such as biology, computer science, and physics, the mechanisms responsible for their formation are not yet well understood.

Manuscript received January 4, 2016; revised June 9, 2016; accepted August 25, 2016. Date of publication September 13, 2016; date of current version March 16, 2018. This work was supported in part by ONR under Grant ONR N00014-14-1-0554, in part by the National Science Foundation under Grant CBET-0939511, in part by the Austrian FFG project HARMONIA (no. 845631), in part by the Austrian FWF-funded (no. S 11405-N23) SHiNE project, and in part by the EU ICT COST Action IC1402 on Runtime Verification beyond Monitoring (ARVI). Recommended by Associate Editor Rafael Fierro.

E. Bartocci is with the Institute of Computer Engineering, Vienna University of Technology, Vienna, Austria (e-mail: ezio.bartocci@tuwien.ac.at).

E. Aydin Gol is with the Department of Computer Engineering, Middle East Technical University, Ankara Turkey (e-mail: ebrugol@metu.edu.tr).

I. Haghighi and C. Belta are with the Division of Systems Engineering, Boston University, Boston, MA, USA (e-mail: haghighi@bu.edu; cbelta@bu.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCNS.2016.2609138

Pattern recognition is usually considered as a branch of machine learning [3] where patterns have a statistical characterization [4] or they are described through a structural relationship among their features [5]. Even though pattern recognition has been successful in several application areas [6], it still lacks of a formal foundation and a suitable high-level specification language that can be used to specify patterns in a concise and intuitive way and to reason about them in a systematic way.

In particular, we are interested in the following questions. Can patterns be specified in a formal language with well-defined syntax and semantics? Can we develop algorithms for pattern detection from specifications given in such a language? Given a large collection of locally interacting agents, can we design parameter synthesis rules, control and interaction strategies guaranteeing the emergence of global patterns? In this paper, our goal is to provide some preliminary answers to such questions by drawing inspiration from the field of computer aided verification and model checking [7], [8].

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 leverages on a novel spatial superposition logic, called *tree spatial superposition logic* (TSSL), whose semantics are defined over quad trees of partitioned images. In our setting, a pattern descriptor is a TSSL formula and detecting the existence of a pattern in an image is a model-checking problem. We can either manually specify the pattern using the TSSL syntax or we can employ machine-learning techniques using rule-based classifiers to infer such a formula from given sets of positive and negative examples. We also develop a computational framework where our model-checking algorithm works in synergy with a particle swarm optimization technique to synthesize the parameters leading to patterns of interest in reaction diffusion systems.

The optimization fitness function is given by a measure of satisfaction induced by the quantitative semantics that we define for the logic. The positive and negative signs of this measure are sound w.r.t. the satisfaction or violation of the formula, while the absolute value represents "how far" an image is from a desired pattern. We provide examples demonstrating that TSSL formulas can encode, for some commonly encountered patterns, very good classifiers. Furthermore, we compared TSSL formulas with traditional linear classifiers, and in all of the examples, the classification rate of the TSSL formula was the highest (more

than 95%). In the examples, we focus on the Turing reaction diffusion network [9], and show that pattern-producing parameters can be automatically generated with our method. However, the overall computational approach can, in principle, be applied to any network of locally interacting systems.

The rest of this paper is organized as follows. In Section II, we discuss related work. In Section III, we formulate the problem and outline our approach. We define the syntax and semantics of TSSL in Section IV. A machine-learning technique to learn TSSL formulas from positive and negative examples of desired patterns is developed in Section V. The solution to the pattern generation problem is presented in Section VI as a supervised, iterative procedure that integrates quantitative model checking and optimization. We conclude with final remarks and directions for future work in Section IX.

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: 1) to extract distinctive features [10]–[13] with relevant information from a set of input data representing the pattern of interest and 2) to build, using one of the several available machine-learning techniques [14], an accurate classifier trained with the extracted features. The descriptor chosen in the *feature extraction phase* depends on the application domain and the specific problem.

This work is related to pattern recognition in *computer vision*, where these descriptors may assume different forms. Feature descriptors, such as *textons* [10] and *histograms of oriented gradients* (HoG) [11], are concerned with statistical information of color distributions of intensity gradients and edge directions. The *scale-invariant feature transform* (SIFT), proposed by Lowe in [13], is based on the appearance of an object at particular interest points, and is invariant to image scale and rotation. The *shape context* [12] is another feature descriptor intended to describe the shape of an object by the points of its contours and the surrounding context.

In this paper, we establish a connection between verification and pattern recognition. Both classical verification [15]–[19] 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 descriptors allows to reason about patterns and to infer interesting properties, such as spatial periodicity and self-similar (fractal) texture. Furthermore, combining different pattern descriptors using modal and logical operators is quite intuitive.

This paper is inspired by the original work on morphogenesis by Turing [9], and is closely related to [20]. In the latter reference, the authors introduced a linear spatial superposition logic (LSSL), whose formulas were interpreted over quad-tree image 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 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 expressed in LSSL, because they rely on a tree representation rather than a path representation.

As opposed to [20], we also define a quantitative semantics for the logic, which can be seen as a "distance" to satisfaction given an image and a formula. We use this distance as a fitness function in an optimization problem to search for pattern-producing parameters in a system. This quantitative semantics and the discounted model checking on a computational tree are inspired from [21], with the notable difference that we do not need a metric distance, but rather a measure of satisfiability. Such measures have also been used in [15]–[19]. While such measures exist for classical classifiers such as support vector machines (SVM) [3], Fisher linear discriminants (FLD) [3], and Kozinec's hyperplane [22]in the form of the distance from an image to the classifying hyperplane in the feature space, we show (through numerical experiments) that the measure induced by the quantitative semantics of TSSL is better suited for optimization algorithms.

This paper is also related to the vast literature on consensus protocols (see [23]–[25]). As in these works, here we consider a network of locally interacting dynamical systems, and we are interested in achieving a desired, emergent global behavior. However, as opposed to most works in this area, the global behavior we consider is a spatially distributed pattern, rather than an agreement on some quantity. Moreover, rather than showing that some global behavior emerges from given local interactions, we design a top-down approach in which we prescribe the global behavior and then synthesize the local dynamics achieving it.

Part of the material from this paper appeared in the Proceedings of the IEEE Conference on Decision and Control (CDC) 2014 [26], where most of the theoretical results were presented without proofs. In addition to the technical details, this paper includes:

1) a notion of max distance between two quad transition systems (see Definition 9) in Section IV;
2) a theorem on the correctness of the TSSL qualitative semantics w.r.t. the quantitative semantics given two quad transition systems with a given max distance (Theorem 2) in Section IV;
3) a comparison of the classification and quantification capabilities offered using TSSL w.r.t. the traditional linear classifiers in Section VII;
4) a new version of TSSL with basic propositions expressing constraints over higher statistical moments and an example on the improved effectiveness for pattern synthesis in Section VIII.

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 $c \in \mathbb{R}$ and set $\mathcal{S} \subseteq \mathbb{R}$, $\mathcal{S}_{>c} := \{x \in \mathcal{S} \mid x > c\}$, and for any $a, b \in \mathbb{R}$, $\mathcal{S}_{[a,b]} := \{x \in \mathcal{S} \mid a \leq x \leq b\}$.

A reaction diffusion network $\mathbf{S}$ 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

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

where $\mathbf{x}_{i,j} = [x_{i,j}^{(1)}, \ldots, x_{i,j}^{(N)}]$ is the N-dimensional state vector of system $S_{i,j}$, which captures the concentrations of all species of interest. Diffusion coefficients $\mathbf{D} = [D_1, \ldots, D_N] \in \mathbb{R}_+^N$ and reaction constants $\mathbf{R} \in \mathbb{R}^{P-N}$ are the parameters of a system $\mathbf{S}$. The local dynamics $f_n : \mathbb{R}_+^N \times \mathbb{R}^{P-N} \to \mathbb{R}$ are defined by $\mathbf{R}$ 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 nondiffusible 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

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

$\nu_{i,j}$ 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 network. 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 $b \in \mathbb{R}_+$. For example, a subset of the genes in a gene network is tagged with fluorescent reporters. The relative concentrations of the corresponding proteins can be inferred by using fluorescence microscopy.

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

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

where $x_{i,j}^{(n)} = \int_{t-T}^{t} x_{i,j}^{(n)}(\tau) d\tau / T$ for a sufficiently large $T \leq t$. The system is said to be in steady state at time $\bar{t}$, if (2) holds for all $t \geq \bar{t}$. In the rest of this 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 1: We consider a 32×32 reaction diffusion network with two species (i.e., $K = 32$, $N = 2$)

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

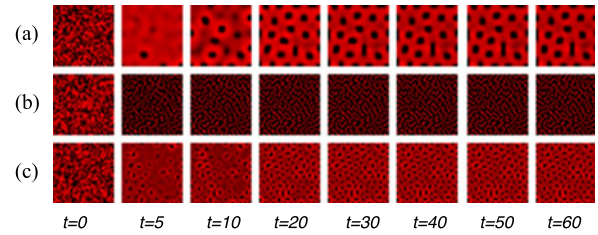

Fig. 1. Observations generated by system (3) with parameters $\mathbf{R}$ and a) $\mathbf{D}_1$; b) $\mathbf{D}_2$; and c) $\mathbf{D}_3$ from Example 1 (the concentration of species 1 is represented with shades of red). The steady-state observations produce: 1) *large spots* (LS); 2) *fine patches* (FP); and 3) *small spots* (SS).

The system is inspired from Turing's reaction diffusion system, which is presented in [27] 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, that is, $x_{i,j}^{(2)}$. 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 the mapping $H : \mathbb{R}_+^{32 \times 32 \times 2} \to \mathbb{R}_{[0,1]}^{32 \times 32}$

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

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 Fig. 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, that is, 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.

In this paper, we consider the following problem:

Problem 1: Given a reaction diffusion network $\mathbf{S}$ as defined in (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 steady-state observations $\mathbf{Y}_+ = \{\mathbf{y}_i\}_{i=1,\ldots,N_+}$ that contain a desired pattern, a set of steady-state 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 that contain the desired pattern.

To solve Problem 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 the definition of a pattern descriptor. With 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. The new logic and the superposition trees are explained in Section IV. Then, finding a pattern descriptor reduces to finding a formula

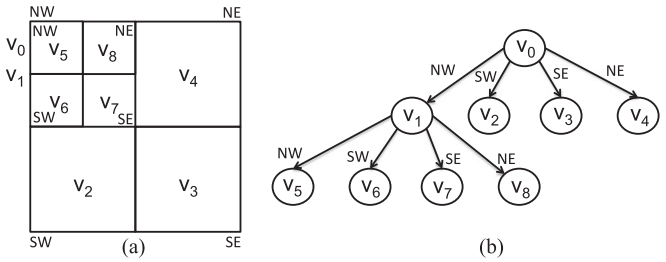

Fig. 2. Quad-tree representation (b) of a matrix (a).

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 network $\mathbf{S}^{(\mathbf{p}^*)}$ 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. The choice of PSO is motivated by its inherent distributed nature, and its ability to operate on irregular search spaces, that is, it does not require a differentiable fitness function. Finally, we propose a supervised, iterative procedure to find $\mathbf{p}^*$ that solves Problem 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 1 is found, such that the corresponding steady-state observations match the desired patterns defined by the user.

IV. Tree Spatial Superposition Logic

A. Quad-Tree Spatial Representation

We represent the observations of a reaction diffusion network 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 submatrix 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 1: A quad-tree $Q = (V, R)$ is a quaternary tree [28] representation of $\mathcal{A}_{k,k}$ where each vertex $v \in V$ represents a submatrix 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 elements of the submatrix that it represents have the same values.

Fig. 2 shows an example of a quadtree, where node v_0 represents the entire matrix; child v_1 represents the submatrix $\{1, \cdots, 2^{k-1}\} \times \{1, \cdots, 2^{k-1}\}$; child v_7 represents the submatrix $\{2^{k-2}+1, \cdots, 2^{k-1}\} \times \{2^{k-2}+1, \cdots, 2^{k-1}\}$; etc. In Fig. 2, we also label each edge in the quad tree with the direction of the submatrix represented by the child: north west (NW), north east (NE), south west (SW), south east (SE).

Algorithm: Building Quad Transition System.

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 $\mathcal{Q}_{TS} = (S, s_\iota, \tau, \Sigma, [.], L)$

$\Sigma := \{m_1, \cdots, m_o\}$ ▷ Initialize the set of variables Σ of $\mathcal{Q}_{TS}$.
$\tau = \emptyset$ ▷ Initialize the set τ of the transition relation τ of $\mathcal{Q}_{TS}$.
$S := \{s_\iota\}$ ▷ Initialize the set of states S of $\mathcal{Q}_{TS}$.
$TS := \{\langle s_\iota, \{v_0\}\rangle\}$
▷ Each tuple in TS contains a state in S and a set of vertices in V.
$LF := \{v \in V | \nexists t \in V : (v, t) \in R\}$ ▷ LF is the set of leaves of Q
$PLF := \{P_i \subseteq LF, 1 \le i \le n | P_i \neq \emptyset \wedge \forall v_a, v_b \in P_i, \forall v_c \in P_{j \neq i}, v_a \equiv v_b \wedge v_a \not\equiv v_c\}$
▷ 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_ι 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.
$LFS := \{v \in FS \mid \forall v' \in V : (v, v') \in R : \exists \langle s, VS\rangle \in TS \wedge v' \in VS\}$
$PLFS := \{P_{i \in I} \subseteq LFS | I \neq \emptyset, P_i \neq \emptyset, \forall v_a, v_b \in P_i, \forall v_c \in P_{j \neq i}, v_a \equiv v_b \wedge v_a \not\equiv v_c\}$
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 $[\bar{s}](m_{\bar{c}}) := \mu_{\bar{c}}(v_{\bar{s}}), \bar{c} \in \{1, \cdots, o\}, v_{\bar{s}} \in VS : \langle \bar{s}, VS\rangle \in TS$
define func L as $L(s, t) := (t = s)?\mathcal{D} : \bigcup_{\tilde{v} \in \tilde{VS}, \bar{v} \in \tilde{VT} : \langle s, \tilde{VS}\rangle, \langle t, \tilde{VT}\rangle \in TS, (\tilde{v}, \bar{v}) \in R} LQ(\tilde{v}, \bar{v})$
return $S, s_\iota, \tau, \Sigma, [.], L$

Definition 2: We define the mean function $\mu_c : V \to [0, b]$ for submatrix $\mathcal{A}_{k,k}[i_s, i_e; j_s, j_e]$ 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 is the sample mean and an estimation for the expected value for an observable variable with index $c, 1 \leq c \leq o$ in a particular region of the space represented by the vertex v.

Definition 3: Two vertices $v_a, v_b \in V$ are said to be equivalent when the mean function applied to the elements of the submatrices that they represent produce the same values

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

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 enumeration of the observations of all locations. This approach is inspired by [20] and [29], where the authors aim to combat the state-explosion problem that would stem otherwise.

Proposition 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:

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

Proof: The proof can be easily derived by expanding the terms of Definition 2. ■

Proposition 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 $\sum_{i=0}^{k} 2^{2i}$.

Proof: The proof follows from the fact that the worst case scenario is when all of the elements have different values. In this case, the cardinality of the set V is equal to the cardinality of a full and complete quaternary tree. For example, to represent the matrix $\mathcal{A}_{3,3}$, it would require a max number of vertices $|V| \leq 1 + 4 + 16 + 64 = 85$. ■

B. Quad Transition System

We now introduce the notion of quad transition system that extends the classical quad-tree structure, allowing for a more compact exploration for model checking.

Definition 4: A *Quad transition system* (QTS) is a tuple $\mathcal{Q}_{TS} = (S, s_\iota, \tau, \Sigma, [.], L)$, where

1) S is a finite set of states with $s_\iota \in S$ the initial state;
2) $\tau \subseteq S \times S$ is the transition relation. We require τ to be nonblocking and bounded branching:
   a) $\forall s \in S, \exists t \in S : (s, t) \in \tau$;
   b) $\forall s \in S$, if $T(s) = \{t : (s, t) \in \tau\}$ is the set of all successors of s, then the cardinality $|T(s)| \leq 4$;
3) Σ is a finite set of variables;
4) $[.]$ 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 $b \in \mathbb{R}_+$;
5) 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 $L(s, t) \cap L(s, t') = \emptyset, \bigcup_{\forall t \in S : (s,t) \in \tau} L(s, t) = \mathcal{D}$.

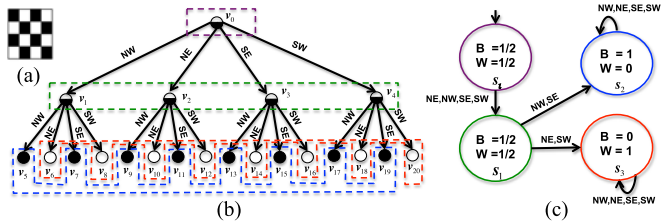

Fig. 3. (a) A checkerboard pattern as a matrix of pixels, (b) the quad-tree representation, and (c) the derived quad transition system, where **B** and **W** denote black and white, respectively.

The BUILDINGQUADTRANSITIONSYSTEM algorithm shows how to generate a QTS starting from a quad-tree representation $Q = (V, R)$ of a matrix $\mathcal{A}_{k,k}$ and a labeling function $LQ : R \to \mathcal{D}$. After an initialization phase (line 1–4), the algorithm starts to partition the set of equivalent leaves (line 5–6). Then, for each element in the partition, it creates a QTS state with a self-loop transition (line 7–10) and a transition from the initial state if the element represents a root's child node in the quad tree. Then, it explores all nonleaf quad-tree nodes in a breadth-first fashion and adds new states and transitions to QTS accordingly (line 12–30). Equivalent quad-tree nodes are represented only by a single state in the QTS. The resulting QTS is more compact than the initial quad tree.

Proposition 3: The transition relation of the QTS $\mathcal{Q}_{TS} = (S, s_\iota, \tau, \Sigma, [.], L)$ generated by the BUILDINGQUADTRANSITIONSYSTEM algorithm always has a least fixed point, that is $\exists s \in S : T(s) = \{s\}$.

Proof: This property holds because the algorithm BUILDINGQUADTRANSITIONSYSTEM generates one (if the quad tree has only one vertex) or more (if the quad tree has multiple leaves) states with only a self-loop transition. ■

Definition 5 (Labeled Paths): Given a set B of labels representing the spatial directions, a *labeled path* (lpath) of a QTS $\mathcal{Q}$ 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 ith element of a path $\pi^B \in LPaths^B(s)$. For example, in Fig. 3, $LPaths^{\{NW,SE\}}(s_\iota) = \{s_\iota s_1 s_2 s_2 \cdots\}$.

C. TSSL Syntax and Semantics

Definition 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

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

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

TSSL resembles the classic CTL logic [30], with the main difference being 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* (like-bounded until in-bounded model checking [31]) to operate on a finite sequence of states. In the following text, 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 7 (TSSL Qualitative Semantics): Let $\mathcal{Q} = (S, s_\iota, \tau, \Sigma, [.], L)$ be a QTS, then $\mathcal{Q}$ *satisfies a TSSL formula* φ, written $\mathcal{Q} \models \varphi$, if and only if $\mathcal{Q}, s_\iota \models \varphi$, where un. eqn. shown at the bottom of the page.

Example 2: Checkerboard Pattern: The checkerboard pattern from Fig. 3(a) can be characterized with the following TSSL formula ($B^* = \{SW, NE, NW, SE\}$):

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

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

$$\forall_{B^*} F_1((\forall_{\{SW,NE\}} \bigcirc (m \geq 1)) \wedge (\forall_{\{NW,SE\}} \bigcirc (m \leq 0)))$$

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 [17]. Since the valuation of a TSSL formula with spatial operators requires traversing and comparing 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. We choose this value to reflect that each node represents a partition of the space that is $\frac{1}{4}$ smaller than its predecessor. In the following text, we provide the definition of the TSSL quantitative semantics necessary to measure the satisfaction of a TSSL specification over a given QTS. We show that the sign of this measure indicates either the fulfilment (positive sign) or the violation (negative sign) of a given specification. We then provide a notion of distance between QTSs, showing the relation between this distance and the TSSL qualitative and quantitative semantics.

Definition 8 (TSSL Quantitative Semantics): Let $\mathcal{Q} = (S, s_\iota, \tau, \Sigma, [.], L)$ be a QTS. The quantitative valuation $[\![\varphi]\!] : S \to [-b, b]$ of a TSSL formula φ is defined as follows:

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

Theorem 1 (Soundness): Let $\mathcal{Q} = (S, s_\iota, \tau, \Sigma, [.], L)$ be a QTS, $s \in S$ a state of $\mathcal{Q}$, and φ a TSSL formula. Then, the following properties hold for the two semantics:

$$[\![\varphi]\!](s) > 0 \Longrightarrow \mathcal{Q}, s \models \varphi \qquad [\![\varphi]\!](s) < 0 \Longrightarrow \mathcal{Q}, s \not\models \varphi$$

Proof: The proof can be derived by structural induction on the operational semantics. ■

Remark 1: Theorem 1 provides the basis of the techniques for the parameter synthesis discussed in the following sections. $[\![\varphi]\!](s)$ enables the process of quantitative valuation of a TSSL formula φ over a QTS by performing the recursive computation presented in Definition. 8. The computational cost is linear in the QTS size and polynomial in the length of the formula. It is worth noting 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.

We now introduce a notion of distance between two given QTSs. This measure quantifies, by recursively exploring the corresponding pair of nodes of two QTSs, the max absolute difference between the evaluation of the variables in the pair of nodes discounted by a factor $1/4^k$. The term k is the recursion level of the explored pair of nodes. A higher level leads to smaller partitions of the space that the pair of nodes represent. Consequently, their max absolute difference is less important.

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

Since the nodes correspond to partitions of the space, the max distance computes the overall worst discrepancy between corresponding partitions of the space.

Definition 9 (QTS Max Distance): The max distance of two QTSs $Q^{(1)} = (S^{(1)}, s_l^{(1)}, \tau^{(1)}, \Sigma, [.]^{(1)}, L^{(1)})$ and $Q^{(2)} = (S^{(2)}, s_l^{(2)}, \tau^{(2)}, \Sigma, [.]^{(2)}, L^{(2)})$ is defined as

$$d_\infty(Q^{(1)}, Q^{(2)}) = n_\infty(s_l^{(1)}, s_l^{(2)}, 0)$$

where $n_\infty : \mathcal{S} \times \mathcal{S} \times \mathbb{N} \to [0, b]$ is the max distance between states of different QTSs such that

$$n_\infty(s^{(1)}, s^{(2)}, k) = \begin{cases} \frac{1}{4^k} \max\limits_{[m \in \Sigma]} |[s^{(1)}]^{(1)}(m) - [s^{(2)}]^{(2)}(m)| \\ \text{if } (s^{(1)}, s^{(1)}) \in \tau^{(1)} \wedge (s^{(2)}, s^{(2)}) \in \tau^{(2)} \\ \max\limits_{[(\tilde{s}^{(1)}, \tilde{s}^{(2)}) \in S^*]} n_\infty(\tilde{s}^{(1)}, \tilde{s}^{(2)}, k+1) \text{otherwise} \\ S^* = \{(\tilde{s}^{(1)}, \tilde{s}^{(2)}) | \tilde{s}^{(1)} \in S^{(1)}, \tilde{s}^{(2)} \in S^{(2)} \wedge \\ L^{(1)}(s^{(1)}, \tilde{s}^{(1)}) \cap L^{(2)}(s^{(2)}, \tilde{s}^{(2)}) \neq \emptyset\} \end{cases}$$

It is worth noting that if two pictures are the same, but they have a different number of pixels, then their QTS representations will be equivalent and their max difference will be zero.

We now introduce a second theorem, showing the correctness of the qualitative semantics w.r.t. the quantitative semantics. According to this theorem, if the max distance between two QTSs is less than the quantitative valuation of a TSSL formula φ over the first QTS satisfying φ, then we also know that the other QTS satisfies φ.

Theorem 2 (Correctness): Given a TSSL formula φ and two QTSs $Q^{(1)} = (S^{(1)}, s_l^{(1)}, \tau^{(1)}, \Sigma, [.]^{(1)}, L^{(1)})$ and $Q^{(2)} = (S^{(2)}, s_l^{(2)}, \tau^{(2)}, \Sigma, [.]^{(2)}, L^{(2)})$ and two states $s^{(1)} \in S^{(1)}$ and $s^{(2)} \in S^{(2)}$. If $Q^{(1)}, s^{(1)}$ satisfies the formula φ and the max distance $n_\infty(s^{(1)}, s^{(2)}, 0)$ is less than the quantitative evaluation $[\![\varphi]\!](s^{(1)})$ of φ over $Q^{(1)}$ then also $Q^{(2)}, s^{(2)}$ satisfies the same formula φ. Formally

$$Q^{(1)}, s^{(1)} \models \varphi \wedge n_\infty(s^{(1)}, s^{(2)}, 0) < [\![\varphi]\!](s^{(1)}) \Rightarrow Q^{(2)}, s^{(2)} \models \varphi$$

Proof: (Sketch) We can distinguish the following cases:

case $\varphi := \top$**:** in this case, the theorem is true following the definition of the qualitative semantics (see Definition 7) for which $Q^{(1)}, s^{(2)}$ and $Q^{(2)}, s^{(2)}$ satisfy $\top$.

case $\varphi := m \geq d$**:**

In this case, we have:

a) $n_\infty(s^{(1)}, s^{(2)}, 0) < [\![\varphi = m \geq d]\!](s^{(1)})$ (see hypothesis);
b) $[\![m \geq d]\!](s^{(1)}) = [s^{(1)}]^{(1)}(m) - d$ (Def. 8);
c) $[s^{(1)}]^{(1)}(m) - n_\infty(s^{(1)}, s^{(2)}, 0) - d > 0$ [from a) and b)];
d) $|[s^{(1)}]^{(1)}(m) - [s^{(2)}]^{(2)}(m)| \leq n_\infty(s^{(1)}, s^{(2)}, 0)$ (Def. 9); if we substitute $n_\infty(s^{(1)}, s^{(2)}, 0)$ with $|[s^{(1)}]^{(1)}(m) - [s^{(2)}]^{(2)}(m)|$ in c), given d), we can safely obtain
e) $[s^{(1)}]^{(1)}(m) - |[s^{(1)}]^{(1)}(m) - [s^{(2)}]^{(2)}(m)| - d > 0$. Using the property of the absolute difference, we have:
f) $[s^{(2)}]^{(2)}(m) \geq [s^{(1)}]^{(1)}(m) - |[s^{(1)}]^{(1)}(m) - [s^{(2)}]^{(2)}(m)|$.

If we substitute $[s^{(2)}]^{(2)}(m)$ with $[s^{(1)}]^{(1)}(m) - |[s^{(1)}]^{(1)}(m) - [s^{(2)}]^{(2)}(m)|$ in e), given f), we have $[s^{(2)}]^{(2)}(m) - d > 0$.

Finally, using the Theorem 1, we obtain the following: $[s^{(2)}]^{(2)}(m) - d > 0 \Rightarrow [\![m \geq d]\!](s^{(2)}) > 0 \Rightarrow Q^{(2)}, s \models \varphi$

case $\varphi := m \leq d$**:**

In this case, we have

g) $n_\infty(s^{(1)}, s^{(2)}, 0) < [\![\varphi = m \leq d]\!](s^{(1)})$ (see hypothesis);
h) $[\![m \leq d]\!](s^{(1)}) = d - [s^{(1)}]^{(1)}(m)$ (Def. 8);
i) $d - [s^{(1)}]^{(1)}(m) - n_\infty(s^{(1)}, s^{(2)}, 0) > 0$ [from g) and h)];
j) $|[s^{(1)}]^{(1)}(m) - [s^{(2)}]^{(2)}(m)| \leq n_\infty(s^{(1)}, s^{(2)}, 0)$ (Def. 9). If we substitute $n_\infty(s^{(1)}, s^{(2)}, 0)$ with $|[s^{(1)}]^{(1)}(m) - [s^{(2)}]^{(2)}(m)|$ in i), given j), we can safely obtain
k) $d - [s^{(1)}]^{(1)}(m) - |[s^{(1)}]^{(1)}(m) - [s^{(2)}]^{(2)}(m)| > 0$. Using the property of the absolute difference, we have
l) $-[s^{(2)}]^{(2)}(m) \geq -[s^{(1)}]^{(1)}(m) - |[s^{(1)}]^{(1)}(m) - [s^{(2)}]^{(2)}(m)|$.

If we substitute $-[s^{(2)}]^{(2)}(m)$ with $-[s^{(1)}]^{(1)}(m) - |[s^{(1)}]^{(1)}(m) - [s^{(2)}]^{(2)}(m)|$ in k), given l), we have $d - [s^{(2)}]^{(2)}(m) > 0$.

Finally, using Theorem 1 we obtain the following: $d - [s^{(2)}]^{(2)}(m) > 0 \Rightarrow [\![m \leq d]\!](s^{(2)}) > 0 \Rightarrow Q^{(2)}, s \models \varphi$

all of the other cases:

If $Q^{(1)}, s^{(1)} \models \varphi$, then we have

$$[\![\varphi]\!](s^{(1)}) = \begin{cases} (1) : \frac{1}{4^j} b : j \in \mathbb{N} \\ (2) : \frac{1}{4^j}([s^{(1)}]^{(1)}(m^{(1)}) - d) : j \in \mathbb{N} \end{cases}$$

Situation (1) may occur when one of the subformulae of φ is $\top$ and the proof is equivalent to the case of $\varphi := \top$. Situation 2) can be proved in a similar way as the case $\varphi := m \sim d$. ■

Proposition 4: Given a TSSL formula φ and two QTSs $Q^{(1)} = (S^{(1)}, s_l^{(1)}, \tau^{(1)}, \Sigma, [.]^{(1)}, L^{(1)})$ and $Q^{(2)} = (S^{(2)}, s_l^{(2)}, \tau^{(2)}, \Sigma, [.]^{(2)}, L^{(2)})$ then

$$Q^{(1)} \models \varphi \wedge d_\infty(Q^{(1)}, Q^{(2)}) < [\![\varphi]\!](s_l^{(1)}) \Rightarrow Q^{(2)} \models \varphi.$$

Proof: This is a special case of Theorem 2 where $s^{(1)}$ and $s^{(2)}$ are the initial states $s_l^{(1)}$, $s_l^{(2)}$ of $Q^{(1)}$ and $Q^{(2)}$, respectively. ■

Remark 2: The correctness theorem implies that the higher the quantitative valuation of a TSSL formula is with respect to a QTS, the harder it is to violate the formula by perturbing the QTS since the maximum distance between the perturbation and the original QTS must be at least equal to the quantitative valuation. In other words, a higher positive quantitative valuation means a more robust satisfaction of a formula under QTS perturbations. This is why the quantitative valuation of a TSSL formula is also called its robustness degree.

V. TSSL Pattern Classifiers

A QTS can be seen in the context of multiresolution 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,

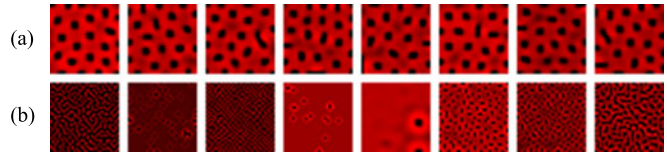

Fig. 4. Sample sets of images from the sets (a) $\mathbf{Y}_{+}^{(1)}$ and (b) $\mathbf{Y}_{-}^{(1)}$ for the LS pattern.

such as a pattern. Here, we propose using 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

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

where $\mathcal{Q}_{\mathbf{y}}$ is the QTS generated from $\mathbf{y}$. Then, we separate the data set $\mathcal{L}$ into disjoint training and testing sets $\mathcal{L}_L, \mathcal{L}_T$. In machine learning, the training set is used to learn a classifier for a target class (for example, $+$), and the testing set is used to measure the accuracy of the classifier. We employ RIPPER [32] as a rule-based learner, to learn a classifier from $\mathcal{L}_L$, and then translate the classifier into a TSSL formula characterizing $+$. The classifier is composed of a set of rules. Each rule 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, for example, $[s](m) > d$, 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 as a state s is uniquely represented using the existential ($\exists$) and next ($\bigcirc$) operators along the path from 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, an equivalent TSSL formula for the desired property is defined as follows:

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

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.

Example 3: LS Pattern: For the LS pattern from Example 1, we generate a data set $\mathbf{Y}_{+}^{(1)}$ containing 8000 positive examples by simulating the reaction diffusion system (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 (3) from random initial conditions. However, in this case, we use $\mathbf{R}$ and randomly choose the diffusion coefficients from $\mathbb{R}^2_{[0,30]}$. As stated before, we only consider the observation of a system in steady state; for this reason, simulated trajectories that do not reach steady state in 60 time units are discarded. A sample set of images from the sets $\mathbf{Y}_{+}^{(1)}$ and $\mathbf{Y}_{-}^{(1)}$ is shown in Fig. 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 the RIPPER algorithm implemented in Weka [33] 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 2.8 GHz with 8 GB of memory. The classifier consists of 24 rules. The first rule is

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

R denotes the root of a QTS, and the labels of the children are shown in Fig. 2, and $+$ indicates the presence of the pattern.

Rule r_1 translates to the following TSSL formula:

$$\Phi_1 : (m \geq 0.59) \wedge (m \leq 0.70) \wedge (\exists_{NW} \bigcirc \exists_{NW} \bigcirc \exists_{NW} \bigcirc \exists_{SE} \bigcirc m \leq 0.75) \wedge (\exists_{NW} \bigcirc \exists_{NW} \bigcirc \exists_{NW} \bigcirc \exists_{NW} \bigcirc m \geq 0.45). \tag{5}$$

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

FP and SS Patterns: We follow the above 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 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

In this section, we present the solution to Problem 1that is, a framework to synthesize parameters $\mathbf{p} \in \mathcal{P}$ of a reaction diffusion network $\mathbf{S}$ (1) such that the observations of system $\mathbf{S}^{(\mathbf{p})}$ satisfy a given TSSL formula Φ. First, we show that the parameters of a reaction diffusion system that produce trajectories satisfying the TSSL formula can be found by optimizing quantitative model-checking results. Second, we include the optimization in a supervised iterative procedure for parameter synthesis.

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 $\mathcal{Q} = (S, s_\iota, \tau, \Sigma, [.], L)$ of the corresponding observation $H(\mathbf{x}(\bar{t}))$ satisfies Φ, that is, $\mathcal{Q} \models \Phi$, 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

$$[\![\Phi]\!](\mathbf{S}^{(\mathbf{p})}) = \min_{x_0 \in \mathcal{X}_0} \{[\![\Phi]\!](s_\iota) \mid \mathcal{Q} = (S, s_\iota, \tau, \Sigma, [.], L) \text{ is QTS of } H(\mathbf{x}(\bar{t})), \mathbf{x}(0) = x_0\} \tag{6}$$

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 (6). 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 (6) 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 performing the following

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

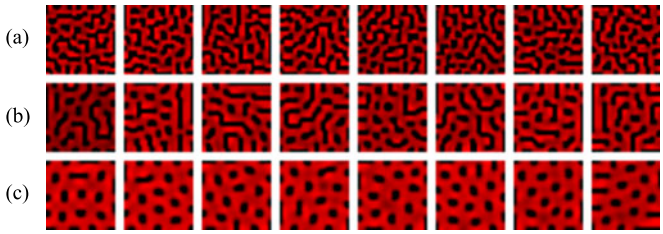

Fig. 6. Sample set of observations obtained by simulating (a) $\mathbf{S}^{([2.25,29.42])}$, (b) $\mathbf{S}^{([3.75,28.75])}$, and (c) $\mathbf{S}^{([6.25,29.42])}$.

steps for each $x_0 \in \mathcal{X}_0$: simulating the system $\mathbf{S}^{(\mathbf{p})}$ from x_0, generating QTS $\mathcal{Q}$ of the corresponding observation, and quantitative model checking of $\mathcal{Q}$ against Φ. Here, we use the PSO algorithm [34] over $\mathcal{P}$ with (6) as the fitness function. The choice of PSO is motivated by its inherent distributed nature, and its ability to operate on irregular search spaces. In particular, PSO does not require a differentiable fitness function.

Example 4: LS Pattern: We consider the reaction diffusion network from Example 1 and the TSSL formula $\Phi_+^{(1)}$ corresponding to the LS pattern from Example 3. 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 (6). The PSO computation was distributed on 16 processors at 2.1 GHz 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 Fig. 6(a). Note that while all of 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 also apply the PSO algorithm on the same setting explained before to maximize the induced valuation (6) for the TSSL formulas $\Phi_+^{(2)}$ (FP pattern) and $\Phi_+^{(3)}$ (SS pattern) from Example 3. The optimized parameters are $[0.083, 11.58]$ and $[1.75, 7.75]$ for $\Phi_+^{(2)}$ and $\Phi_+^{(3)}$, respectively. The sets of observations obtained by simulating systems $\mathbf{S}^{([0.083,11.58])}$ and $\mathbf{S}^{([1.75,7.75])}$ are shown in Fig. 5. In contrast with the LS pattern, the observations are similar to the ones from the corresponding data sets, that is, $\mathbf{Y}_+^{(2)}$ and $\mathbf{Y}_+^{(3)}$.

Remark 3: In this paper, we consider the observations generated from a given set of initial conditions $\mathcal{X}_0$. However, the initial condition can be set as a design parameter and optimized in PSO over a given domain $\mathbb{R}^{K\times K\times N}_{[a,b]}$.

As seen in Example 4, 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 do not produce good patterns. A natural extension of our method should enable adding the newly obtained simulations to the negative training set, and to reiterate the whole procedure. This approach is summarized in the INTERACTIVEDESIGN algorithm.

Algorithm: InteractiveDesign.

Input: Parametric reaction diffusion system $\mathbf{S}$, ranges of parameters $\mathcal{P}$, a set of initial states $\mathcal{X}_0$, sets of observations $\mathbf{Y}_+$ and $\mathbf{Y}_-$

Output: Optimized parameters $\mathbf{p}$, the corresponding valuation γ (no solution if $\gamma < 0$)

1: **while** $True$ **do**
2: $\Phi = Learning(\mathbf{Y}_+, \mathbf{Y}_-)$
3: $\{\mathbf{p}, \gamma\} = Optimization(\mathbf{S}, \mathcal{X}_0, \Phi)$ $\triangleright$ γ is the induced valuation of $\mathbf{S}^{(\mathbf{p})}$
4: **if** $\gamma < 0$ **then return** $\mathbf{p}, \gamma$
5: **end if**
6: UserQuery: Show observations of trajectories of $\mathbf{S}^{(\mathbf{p})}$ originating from $\mathcal{X}_0$.
7: **if** User approves **then return** $\mathbf{p}, \gamma$
8: **else**
9: $\mathbf{Y}_- = \mathbf{Y}_- \cup \{H(x(\bar{t})) \mid x(t), t \geq 0,$ is generated by $\mathbf{S}^{(\mathbf{p})}, x(0) \in \mathcal{X}_0\}$.
10: **end if**
11: **end while**

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 algorithm 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, that is, learn a new formula, run the optimization until the user terminates the process or the optimization step fails ($\gamma < 0$).

Example 5: LS Pattern: We apply theINTERACTIVEDESIGN algorithm to the system from Example 4. A sample set of observations obtained in the first iteration is shown in Fig. 6(a). We decide that these observations are not similar to the ones from the set $\mathbf{Y}_+^{(1)}$ shown in Fig. 4(a), and add these 250 observations generated with the optimized parameters to $\mathbf{Y}_-^{(1)}$ (line 9). 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 Fig. 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 Fig. 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 the set $\mathbf{Y}_+^{(1)}$ and we terminate the algorithm.

Remark 4: As mentioned earlier, the model-checking procedure (computation of quantitative valuation) is very efficient (linear in the size of the system and polynomial in the length of the formula). For instance, computing the quantitative valuation for the TSSL formula corresponding to the LS pattern against a 32 by 32 image takes about 0:5 s on an iMac with an Intel Core i5 processor at 2.8 GHz with 8 GB of memory. However, a large number of unknown parameters would be problematic for the developed parameter synthesis framework since such a system requires a very large swarm population and a great number of iterations for PSO independent of the fitness function (e.g., quantitative model checking).

VII. Comparison: TSSL and Linear Classifiers

In this section, we provide a comparison between TSSL and well-known learning algorithms based on linear classifiers.

A. Linear Classifiers

Assume we have m data points $(x^i, y^i) : i = 1, 2, \ldots, m$, where $x^i \in \mathbb{R}^d$ is a vector containing d features that correspond to the ith example in the training set and $y^i \in \{-1, +1\}$ is the class label associated with x^i. A *linear classifier* is a function of the form $h(x) = \mathrm{sgn}(w^T x + b)$, where $w \in \mathbb{R}^d$ is the normal vector corresponding to the hyperplane $\{x \in \mathbb{R}^d : w^T x + b = 0\}$ and $b \in \mathbb{R}$ is the hyperplane's bias, and sgn is the signum indicator function. The Euclidean distance between a point in the feature space and the hyperplane is called the *geometric margin*, $\gamma(x) = \frac{w^T x + b}{||w||}$. Notice that the geometric margin of a point can be viewed as a distance to pattern satisfaction, since the class prediction of a testing point with a higher geometric margin is stronger.

The goal of the learning problem is to find w and b such that $h(x)$ correctly classifies the training data points. Several algorithms have been proposed in the machine-learning literature to learn a classifying hyperplane for a given data set. In this paper, we use three such algorithms and compare the results with TSSL:

1) Support vector machines (SVMs) [3]: A hyperplane is chosen such that the minimum margin among all data points in the training set is maximized. It is shown in [3] that an SVM can be learned when the data are not linearly separable by solving a quadratic programming problem (Soft Margin Method). Furthermore, SVM can be kernelized, that is, kernel functions can be used to map the original data points to a higher dimensional space where the data are linearly separable, which results in a nonlinear classifier in the original feature space.
2) Fisher linear discriminant (FLD) [3]: A hyperplane is obtained by maximizing the between-class variance while minimizing the within-class variances.
3) Kozinec's algorithm [22]: A separating hyperplane is learned in an iterative procedure that applies corrections to classify each point in the training set.

TABLE I
Classification Rates of TSSL (Learned by RIPPER) Compared to Linear Classifiers (the Classification Rates are Computed for a Testing Set Consisting of 8000 Examples)

	Correct classification rates		
Classifier	LS	FP	SS
TSSL(RIPPER)	96.7%	96.1%	95.6%
SVM hyperplane	94.5%	91.7%	95.3%
FLD	96.5%	93.9%	92.8%
Kozinec's hyperplane	95.2%	89.1%	92.4%

B. Classification Rate

In this section, we compare the effectiveness of TSSL classifiers and the linear classifiers described before. We created three distinct training sets for the LS, SS, and FP Turing patterns using the procedure discussed in Section V. Each set consists of 4000 positive and 4000 negative examples. Eight-thousand other images were generated to test the results.

We considered two types of features for the linear classifiers. First, we simply considered the normalized concentrations of species 1 in each cell of the grid (i.e., the feature vector is 1024-D for our 32×32 grid). Second, we used histograms of oriented gradients, which were created according to the methodology presented in [11]. For each type of linear classifier and for each type of feature, we learned the classifier, tested it against the testing set, and kept the one with the best class-action rate to compare it with TSSL. Table I shows the results of this comparison.

Remark 5: Learning an SVM requires determining particular design parameters (e.g., proper kernel functions and their parameters, the so-called parameter C in the soft margin method, proper features). These parameters need to be fine-tuned using techniques, such as cross-validation, in order to learn an effective classifier. This is a difficult and time-consuming process for a large number of data points and features. On the other hand, TSSL works effectively without the need for tuning any parameters as shown in Table I.

C. Distance to Pattern

A very important feature of TSSL is its quantitative semantics, which can be used as a measure of "distance to satisfaction". One can use this measure to compare two patterns and determine which one is a "better". Furthermore, this metric is used as fitness function in particle swarm optimization (Section VI) to synthesize system parameters. It is interesting to note that for linear classifiers as described before, one can also view the (e.g., Euclidean) distance between a data point and the classifier hyperplane (geometrical margin) as distance to satisfaction. In this section, we hypothesize that the distance given by TSSL

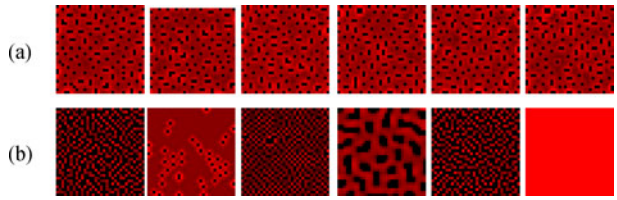

Fig. 7. Examples of images used as a training set to compare TSSL with SVM. (a) Positive examples (SS). (b) Negative examples.

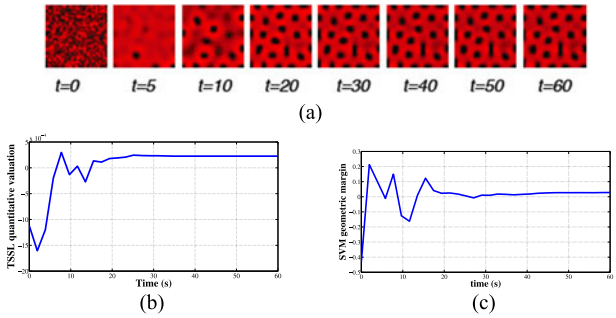

Fig. 8. (a) Formation of LS in steady state. (b) TSSL quantitative valuation with respect to $\Phi_{+}^{(1)}$ [see (5)] at various time steps. (c) SVM geometric margin.

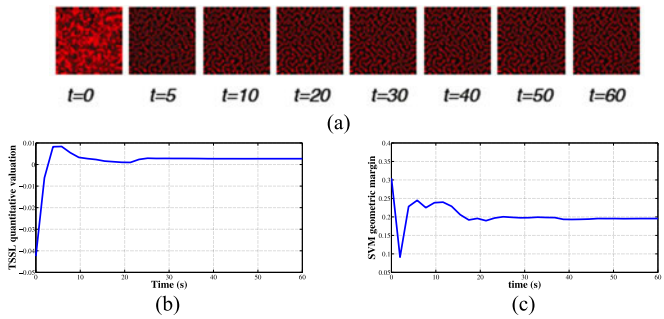

Fig. 9. (a) Formation of FP in steady state. (b) TSSL quantitative valuation with respect to $\Phi_{+}^{(2)}$ at various time steps. (c) SVM geometric margin.

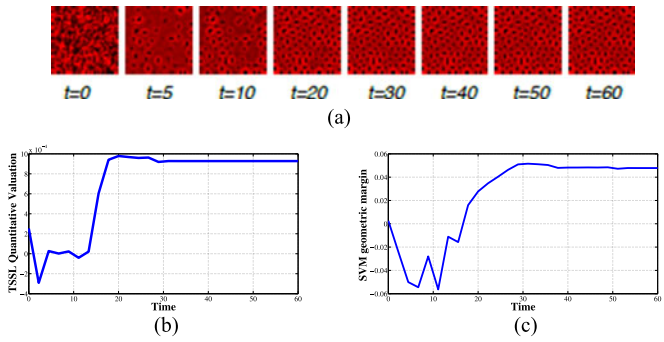

Fig. 10. (a) Formation of SS in the steady state. (b) TSSL quantitative valuation with respect to Φ_{+}^{3} at various time steps. (c) SVM geometric margin.

is more meaningful and more useful for optimization-based pattern synthesis than that given by linear classifiers.

Figs. 8, 9, and 10 show a comparison between the TSSL and SVM metrics. Each figure shows the evolution of the metric over time for each of the three considered patterns: LS, FP, and SS. In all three cases, the TSSL quantitative valuation is better behaved. Indeed, the TSSL curves have fewer local optima (e.g., Figs. 9 and 10), and reach global maxima at steady state (e.g., Figs. 8 and 9).

VIII. Higher-Order Statistics in TSSL

In previous sections, TSSL formulae have been learned using the first moments as features in nodes of the quad trees. In other words, we have assumed that $\Sigma = \{m\}$ in the definition of quad transition systems where m denotes mean values. In this section, we study the effect of adding higher moments to the set of variables Σ. In particular, we added variance of concentrations of specie 1 to the set of variables in the QTS and investigated how it improves the results.

Assume that the set of variables in the QTS is $\Sigma = \{m, v\}$ where m represents mean values and v represents variance, respectively. We repeated the procedure presented in Sections V and VI and observed that:

1) **Improvement in pattern recognition:** Adding variance significantly reduces the length and number of RIPPER classification rules and, thus, the TSSL formula that represents a given pattern will be much shorter. Although the enhancement in prediction accuracy is limited and often negligible, this has a notable effect on the computation time, since shorter classification rules are easier and faster to learn.
2) **Improvement in pattern synthesis:** The complexity of TSSL quantitative valuation for a given formula is proportional to the length of the formula. Therefore, a shorter TSSL formula results in a faster computation of (6). Consequently, the optimization step in Algorithm InteractiveDesign is performed faster since we need to compute the quantitative valuation at every iteration of a PSO.

Example 6: LS Pattern: We considered the experiment described in Example 3 and repeated the same procedure (same training and testing sets and simulation variables) using first- and second-order statistics for the LS pattern. The learning step took 23.3 s on the same computer described in Example 3. The classifier consists of eight rules. Note that the experiment of Example 3 consisted of 24 rules which were learned in 228 s. The classifier that is built using mean and variance of observed concentrations yields a high prediction accuracy (98.27%).

IX. Conclusion and Future Work

We defined a tree-spatial superposition logic (TSSL) whose semantics is naturally interpreted over quad trees of partitioned images. We showed that formulas in this logic can be efficiently learned from positive and negative examples. We defined a quantitative semantics for TSSL and, combined with an optimization algorithm, to develop a supervised, iterative procedure for the synthesis of pattern-producing parameters in a network of locally interacting dynamical systems.

While the experiments show that the current version of the logic works quite well and can accommodate translational and rotational symmetries commonly found in biology patterns, there are several directions of future work. First, we do not

exploit the full semantics of the logic in this paper. In future work, we plan to investigate reasoning about multiple branches and using the "until" operator. Second, we plan to apply this method to more realistic networks, such as populations of locally interacting engineered cells. We expect that experimental techniques from synthetic biology can be used to "tune" existing synthetic gene circuits to produce global desired patterns.

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] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[4] A. Jain, R. Duin, and J. Mao, "Statistical pattern recognition: A review," *IEEE Trans. Pattern Anal. Mach. Learn.*, vol. 22, pp. 4–37, 2000.

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

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

[7] E. A. Emerson, "Temporal and modal logic," in *Handbook of Theoretical Computer Science: Formal Models and Semantics*, J. van Leeuwen, Ed. North-Holland Pub. Co./MIT Press, 1990, vol. B, pp. 995–1072.

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

[9] A. M. Turing, "The chemical basis of morphogenesis," *Philos. Trans. Roy. Soc. London*, vol. 327, pp. 37–72, 1952.

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

[11] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. CVPR: IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recogn.*, vol. 1, Jun. 2005, pp. 886–893.

[12] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, pp. 509–521, 2002.

[13] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proc. Int. Conf. Comput. Vis.*, vol. 2, 1999, pp. 1150–1157.

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

[15] A. Rizk and F. Fages, "From model-checking to temporal logic constraint solving," in *Proc. CP: 15th Int. Conf. Principles Practice Constraint Program., Lisbon, Portugal, 20–24 September, ser. Lecture Notes in Computer Science*, vol. 5732. Springer, 2009, pp. 319–334.

[16] A. Donzé and O. Maler, "Robust satisfaction of temporal logic over real-valued signals," in *Proc. FORMATS, 8th Int. Conf. Formal Modeling and Analysis of Timed Systems*, Klosterneuburg, Austria, Sep. 8–10, vol. 6246, 2010, pp. 92–106.

[17] 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.

[18] G. Fainekos and G. Pappas, "Robust sampling for MITL specifications," in *Proc. FORMATS 2007, 5th Int. Conf. Formal Model. Anal. Timed Syst., ser. Lect. Notes Comput. Sci.*, 2007, vol. 8044, pp. 264–279.

[19] 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.

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

[21] L. de Alfaro, M. Faella, and M. Stoeling, "Linear and branching system metrics," *IEEE Trans. Softw. Eng.*, vol. 35, no. 2, pp. 258–273, 2009.

[22] M. I. Schlesinger and V. Hlavac, *Ten Lectures on Statistical and Structural Pattern Recognition*. Springer, 2002, vol. 24.

[23] W. Ren and R. Beard, *Distributed Consensus in Multi-Vehicle Cooperative Control: Theory and Applications*. Springer-Verlag, London, U.K., 2008.

[24] M. Mesbahi and M. Egerstedt, *Graph Theoretic Methods for Multiagent Networks*. Princeton University Press, Princeton, NJ, USA, 2010.

[25] F. Bullo, J. Cortés, and S. Martínez, *Distributed Control of Robotic Networks. A Mathematical Approach to Motion Coordination Algorithms*. Princeton University Press, 2009.

[26] E. A. Gol, E. Bartocci, and C. Belta, "A formal methods approach to pattern synthesis in reaction diffusion systems," in *Proc. 53rd IEEE Conf. Dec. Control*, Los Angeles, CA, USA, 2014, pp. 108–113.

[27] R. Collantes, "Algorithm alley. Dr. Dobb's journal," Dec. 1996.

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

[29] Y. Kwon and G. Agha, "Scalable modeling and performance evaluation of wireless sensor networks," in *Proc. 12th IEEE Real-Time Embedded Technol. Appl. Symp.* , 2006, pp. 49–58.

[30] E. M. Clarke and E. A. Emerson, "Design and synthesis of synchronization skeletons using branching-time temporal logic," in *Proc. Logic Programs Workshop, Ser. Lect. Notes Comput. Sci.*, vol. 131, 1982, pp. 52–71.

[31] A. Biere, A. Cimatti, E. M. Clarke, O. Strichman, and Y. Zhu, "Bounded model checking," *Adv. Comput.*, vol. 58, pp. 117–148, 2003.

[32] W. W. Cohen, "Fast effective rule induction," in *Proc. 12th Int. Conf. Mach. Learn.*. Morgan Kaufmann, 1995, pp. 115–123.

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

[34] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, vol. 4, 1995, pp. 1942–1948.

Ezio Bartocci received the B.S. degree in computer science, the M.S. degree in bioinformatics, and the Ph.D. degree in information sciences and complex systems from the University of Camerino, Italy, in 2002, 2005, and 2009, respectively.

From 2010 to 2012, he was a Research Associate with the Department of Applied Math and Statistics and Research Scientist with the Department of Computer Science at the State University of New York at Stony Brook, Stony Brook, NY, USA. He joined the Faculty of Informatics at Vienna University of Technology, Vienna, Austria, in 2012 as a University Assistant. Since 2015, he has been a tenure-track Assistant Professor with the Department of Computer Engineering at Vienna University of Technology. The primary focus of his research is to develop formal methods, and computational tools and techniques which support the modeling and the automated analysis of complex computational systems, including software systems, cyberphysical systems, and biological systems.

Ebru Aydin Gol (M'12) received the B.S. degree in computer engineering from ODTU, Ankara, Turkey, in 2008, the M.S. degree in computer science from Ecole Polytechnique Federale de Lausanne, Lausanne, Switzerland, in 2010, and the Ph.D. degree in systems engineering from Boston University, Boston, MA, USA, in 2014.

Currently, he is an Assistant Professor at Orta Dogu Teknik Universitesi (ODTU) in the Department of Computer Engineering. Prior to joining ODTU in 2016, she was a Site Reliability Engineer at Google. Her research interests include formal verification and control, probabilistic verification, hybrid systems, software/release evaluation, as well as verification and design of cyberphysical systems.

Iman Haghighi (S'16) received the B.S. degree in mechanical engineering from Sharif University of Technology, Tehran, Iran, in 2013, the M.S. degree in systems engineering from Boston University, Boston, MA, USA, in 2016, and is currently pursuing the Ph.D. degree in systems engineering at the division of systems engineering, Boston University.

His current research interests include formal verification and synthesis in networked systems, pattern recognition and formation, and spatiotemporal logic synthesis.

Calin Belta (SM'11) is a Professor in the Department of Mechanical Engineering at Boston University, Boston, MA, USA, where he holds the Tegan Family Distinguished Faculty Fellowship. He is the Director of the BU Robotics Lab, and is also affiliated with the Department of Electrical and Computer Engineering, the Division of Systems Engineering at Boston University, the Center for Information and Systems Engineering (CISE), and the Bioinformatics Program. His research focuses on dynamics and control theory, with a particular emphasis on hybrid and cyberphysical systems, formal synthesis and verification, and applications in robotics and systems biology.

Prof. Belta received the Air Force Office of Scientific Research Young Investigator Award and the National Science Foundation CAREER Award.