SpaTeL: A Novel Spatial-Temporal Logic and Its Applications to Networked Systems

Iman Haghighi
Boston University
15 Saint Mary's Street
Brookline, MA
haghighi@bu.edu

Austin Jones
Boston University
15 Saint Mary's Street
Brookline, MA
austinmj@bu.edu

Zhaodan Kong
University of California, Davis
One Shields Avenue
Davis, CA
zdkong@ucdavis.edu

Ezio Bartocci
TU Wien
Treitlstrasse 3
Vienna, Austria
ezio@vmars.tuwien.ac.at

Radu Grosu
TU Wien
Treitlstrasse 3
Vienna, Austria
radu.grosu@tuwien.ac.at

Calin Belta
Boston University
110 Cummington Street
Boston, MA
cbelta@bu.edu

ABSTRACT

Networked dynamical systems are increasingly used as models for a variety of processes ranging from robotic teams to collections of genetically engineered living cells. As the complexity of these systems increases, so does the range of emergent properties that they exhibit. In this work, we define a new logic called Spatial-Temporal Logic (SpaTeL) that is a unification of signal temporal logic (STL) and tree spatial superposition logic (TSSL). SpaTeL is capable of describing high-level spatial patterns that change over time, e.g., "Power consumption in the northwest quadrant of the city drops below 100 megawatts if the power consumption in the southwest quadrant remains above 200 megawatts for two hours." We present a statistical model checking procedure that evaluates the probability with which a networked system satisfies a SpaTeL formula. We also develop a synthesis procedure that determines system parameters maximizing the average degree of satisfaction, a continuous measure that quantifies how strongly a system execution satisfies a given formula. We demonstrate our algorithms on two systems: a biochemical reaction-diffusion system and a demand-side management system for a smart neighborhood.

Categories and Subject Descriptors

D.2.1 [**Software Engineering**]: Requirements/Specifications; F.4.3 [**Mathematical Logic and Formal Languages**]: Formal Languages

General Terms

Algorithms, Theory

HSCC'15, April 14–16, 2015, Seattle, WA, USA.

http://dx.doi.org/10.1145/2728606.2728633.

Keywords

Spatial Temporal Logic; Statistical Model Checking; Parameter Synthesis; Networked Dynamical Systems

1. INTRODUCTION

A networked system can be roughly defined as a system whose components are distributed across space and connected via a communication network. Examples include electrical power grids, teams of robots, gene networks, and groups of animals. The complicated dynamics of individual system components, e.g. the stochastic power consumption demand of individual buildings in a neighborhood, and the interactions between these components via the network, e.g. load balancing between neighborhoods, allow networked systems to produce rich, complicated behaviors, e.g. fulfilling the overall demand for power consumption while ensuring that no load on the network is high enough to trip a blackout. Such dynamic patterns have been studied by diverse communities including physics, biology and computer science [23, 15]. However, these approaches have been domain-specific and thus how to formally characterize and use dynamic patterns is still an open question.

In this paper, we propose a novel logic called Spatial Temporal Logic (SpaTeL) to describe such high-level behaviors of networked systems. Our formulation combines spatial logic, which can be used to describe properties of the system across space such as "The power consumption in neighborhood A and in the adjacent neighborhood B remains below 150 MW and the combined power consumption of the two neighborhoods does not exceed 200 MW", with temporal logic, which can be used to describe properties of the system across time, such as "The power consumption never exceeds 150 MW for longer than 20 minutes and always exceeds 20 MW for 24 hours." With SpaTeL, we can express how a system evolves across space and time with properties such as "The power consumption in A and B each never exceeds 150 MW for longer than 20 minutes and the combined consumption is always between 20 MW and 200MW for 24 hours".

The spatial component of SpaTeL is based on Tree Spatial Superposition Logic (TSSL), which was introduced in [2] for supervised classification of patterns. TSSL is defined with respect to the quad tree data structure that is widely used

in pattern recognition. Quad trees are constructed by recursively partitioning images into quadrants. TSSL is based on Linear Spatial-Superposition-Logic (LSSL), which was developed in [16]. LSSL was successfully employed to specify and to detect the onset of electrical spirals in networks of cardiac myocytes, and the authors devised a method for learning an LSSL formula from examples of spatial patterns. TSSL was introduced in order to deal with the entire quad tree representation of the space instead of just a path. An example of a TSSL property is "The average intensity of the image is less than 0.5 and the average intensity of the northwest quadrant is less than 0.25". Given a quad tree and a TSSL formula, the quantitative semantics of TSSL can be used to describe how well the given tree satisfies the formula. If this value is high and positive, then the tree is a very good example of the pattern; if this value is negative, then the tree does not satisfy the pattern; and if the value is close to 0 a small change to the tree could affect whether or not it satisfies the given pattern. The authors of [2] performed parameter synthesis for a reaction diffusion network by optimizing the quantitative semantics of the steady-state system behavior with respect to a given TSSL formula that described desirable system behavior.

The temporal component of SpaTeL is based on Signal Temporal Logic (STL), a predicate temporal logic defined with respect to continuous-valued signals [21, 11]. STL formulae describe how inequalities defined over a signal evolve over time, e.g. "The squared distance from the origin ($x^2 + y^2$) is greater than or equal to 2 within 5 s of system initialization." Quantitative semantics can also be defined for an STL formula that quantify how well a given signal satisfies a given STL formula. STL has proven a useful tool for supervised learning from continuous signals. Parametric signal temporal logic (PSTL) is a version of STL in which physical constants and time bounds are replaced with free parameters, e.g. "$(x^2 + y^2) < \pi$ within τ seconds." In [17], the quantitative semantics of STL was used to find a parameterization of a given PSTL formula that described system output data. In [19] and [18], these methods were extended to supervised and unsupervised learning applications, respectively. A similar supervised learning problem with respect to Metric Interval Temporal Logic (MITL) was solved in [4].

In this paper, we combine TSSL and STL to form SpaTeL. SpaTeL describes how a networked system (whose state can be encapsulated by an image) evolves over time. We define quantitative semantics for this logic that describes how well a given execution of a system satisfies the given formula. We show how SpaTeL can be used in the analysis of networked systems via statistical model checking and in the control of systems via parameter synthesis. Our proposed algorithms are evaluated in simulations of a networked reaction-diffusion system used to describe skin pigmentation in animals and a smart electrical power grid.

2. RELATED WORK

Several logics have been proposed for specifying the behavior and the spatial structure of concurrent systems [9] and for reasoning about the *topological* [5] or *directional* [7] aspects of the interacting entities. In topological reasoning [5], the spatial objects are sets of points and the relation between them is preserved under translation, scaling and rotation. In directional reasoning, the relation between objects depends on their relative position. These logics are usually highly computationally complex [7] or even undecidable [22]. Even though there has been a lot work done in spatial logics and temporal logics with applications to several domains [8, 13], spatiotemporal reasoning is scarcely explored. To the best of our knowledge, the available results are mainly theoretical [5, 6, 20] and lack real practical applications such as those provided in this paper.

3. NETWORKED DYNAMICAL SYSTEMS AND QUAD TRANSITION SYSTEMS

In this section, we formalize the notion of a networked dynamical system and describe the process of abstracting such a system to a quad transition system (QTS) [2], the model with respect to which SpaTeL is defined. A networked system S can be modeled as a $K \times K$ square grid of K^2 subsystems $S_{i,j}$. We use $x(t) \in \mathbb{R}_+^{K\times K\times N}$ to denote the state of the system at time $t, t = 0, \ldots, T$ where $x_{i,j}(t) \in \mathbb{R}_+^N$ denotes the state of sub-system $S_{i,j}$. Each sub-system evolves according to a (possibly non-deterministic) difference relation. Without loss of generality, we will assume that K is a power of two, i.e. $K = 2^k$. A square sub-system of S is a collection of adjacent subsystems of S denoted $S_{i_1:i_2,j_1:j_2}$ where $1 \le i_1 \le i_2 \le K$, $1 \le j_1 \le j_2 \le K$ and $j_2 - j_1 = i_2 - i_1$ is a power of 2. The state of $S_{i_1:i_2,j_1:j_2}$ at time t is denoted as $x_{i_1:i_2,j_1:j_2}(t) = [x_{i,j}(t)]_{i=i_1,\ldots,i_2,j=j_1,\ldots,j_2}$.

A *quad tree* is a representation of $x(t)$ given as a quarternary tree structure $T(t) = (V(t), R(t))$ where each vertex $v \in V(t)$ represents the state of a square sub-system of S. The set $V(t)$ is constructed by recursively partitioning $x(t)$ into quarters, e.g. if $V(t)$ is initially $v_1 = x(t)$, then $v_2 \ldots v_5$, which represent four $K/2 \times K/2$ square subsystems of S would be added to V. The recursive partitioning happens $k = \log_2(K)$ times. The relation $R \subset V(t) \times V(t)$ is defined such that $(v, v') \in R(t) \Leftrightarrow v'$ was constructed from partitioning v.

The concept of a Quad Transition System (QTS) was introduced in [2] as a compact representation of quad-trees. A *quad transition system* is a tuple $QTS := (A, a_0, \tau, \Sigma, [.], L)$ where A is a set of states, $a_0 \in A$ is an initial state, $\tau \subseteq A \times A$ is a transition relation such that for every state $a \in A$, $1 \le |\{a|(a, a') \in \tau\}| \le 4$. Σ is a set of variables, and $[.] : A \to (\Sigma \to [0, b])$ is a function that assigns to each state $a \in A$ and variable $m \in \Sigma$ a value $[a](m) \in [0, b]$, and $L : \tau \to 2^{\mathcal{D}}$ where $\mathcal{D} = \{NW, NE, SW, SE\}$ is a labeling function for the relation τ with the extra constraint that only one successor exists for every direction.

From the state of the system S at time t, we can construct the QTS $Q(t) := (A(t), a_0(t), \tau(t), \Sigma, [.](t), L(t))$ via the algorithm given in [2]. In this paper, we suppress the dependence of the relation $[.](t)$ on time. We note that in contrast to transition systems typically used in formal methods applications, the QTS constructed via this algorithm represents the spatial relationships (patterns) of the system at a particular time. A *trace* corresponding to a trajectory $x : \{0, \ldots, T\} \to \mathbb{R}_+^{K\times K\times N}$ is a function $Q : \{0, \ldots, T\} \to \mathcal{Q}$ where $\mathcal{Q}$ is the space of quad transition systems. The set of all traces that a networked system S can produce is called the language of S and is denoted as $\mathcal{L}(S)$. The technical details of the QTS construction are omitted, but the relationship of the constructed QTS to the original networked system S is illustrated in the following example.

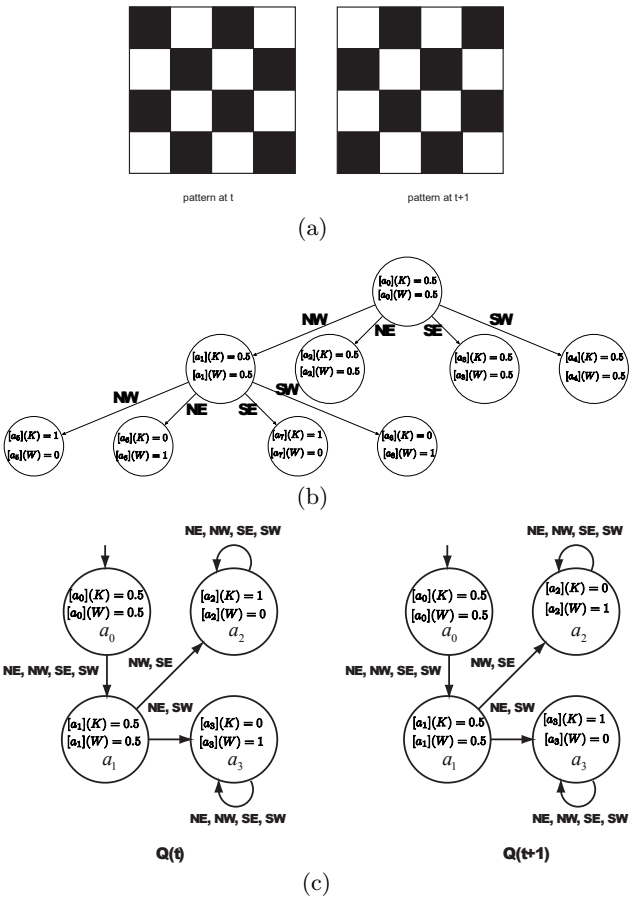


Figure 1: (a) Flipping checkerboard pattern. (b) A portion a the quad tree corresponding to the pattern at time t. (c) The derived QTSs at time t (before the flip) and $t+1$ (after the flip).

Example 1. A 4 by 4 checkerboard can be characterized by the QTSs shown in Figure 1 [2]. Each subsystem $S_{i,j}$ is the (i,j)th cell of the checkerboard with state $x_{i,j}(t) = [K_{i,j}(t), W_{i,j}(t)] \in \{0,1\}^2$, where $x_{i,j}(t) = [1,0]$ if the square is black and $x_{i,j}(t) = [0,1]$ if the square is white. The set of variables is defined as $\Sigma = \{K, W\}$ which represent the proportion of cells of a particular subsystem that are black and white, respectively. The QTS is constructed from the quad tree by first aggregating all of the states on the bottom level of the tree with equivalent values $x_{i,j}(t)$ into QTS states. The valuations of these states are defined as $[[a](K), [a](W)] = x_{i,j}(t)$. $a_2(t)$ and $a_3(t)$ correspond to the black and white cells of the checkerboard, respectively. Each of these states is assigned a self-transition. Next, the states in the next highest level of the quad tree are aggregated into QTS states. States at this level of the tree with identical children are aggregated into the same QTS state. $a_1(t)$ is the only QTS state constructed from this level because every state has identical children. Transitions from a new QTS state to an existing QTS state are constructed if the existing state represented a child of the new state in the quad tree. The transitions are annotated with the direction of the child's corresponding cell, e.g. the transition from $a_1(t)$ to $a_2(t)$ is annotated with the directions NW, SE. The values of $[a](C)$ for the new states a and variables $C \in \Sigma$ are calculated according to

$$[a](C) = \frac{[a_{NW}](C) + [a_{NE}](C) + [a_{SE}](C) + [a_{SW}](C)}{4}$$

where $a_d, d \in \mathcal{D}$ is such that $(a, a_d) \in \tau$ and $d \in L((a, a_d))$. This process continues until a state $a_0(t)$ that denotes the root of the quad tree is constructed.

At some point $t \in \{0, \ldots, T\}$, the color of all of the cells inverts. This flipped colors are represented by the QTS $Q(t+1)$. Note that in this case, the states $a_0(t+1), a_1(t+1)$ have the same values of K and W and the same transition between them as $a_0(t), a_1(t)$. This is because both $x(t)$ and $x(t+1)$ have the property that the neighbors of any given cell $S_{i,j}$ are the opposite color from $S_{i,j}$. However, the values of K and W associated with $a_2(t+1)$ and $a_3(t+1)$ are the opposite of $a_2(t)$ and $a_3(t)$, which demonstrates the color inversion at time t.

Definition 1. (Labeled paths) Given a set B of labels representing the spatial directions, a *labeled path* (lpath) of a QTS is an infinite sequence $\pi^B = a_0 a_1 a_2 \ldots$ of states such that $(a_i, a_{i+1}) \in \tau$ and $L(a_i, a_{i+1}) \cap B \neq 0$, $\forall i \in \mathbb{N}$. We denote the set of all labeled paths starting in state $a(t)$ as $LPath^B(a(t))$ and the i-th element of a path π^B as π^B_i. For example, in Fig. 1(c), $LPath^B(a_0) = \{a_0 a_1 a_2 a_2 \ldots\}$ if $B = \{NW, SE\}$.

4. SPATEL: SPATIAL TEMPORAL LOGIC

In this section, we define the syntax and qualitative and quantitative semantics of Spatial Temporal Logic (SpaTeL).

4.1 Syntax

SpaTeL has a nested syntax where inner spatial formulae are modified by temporal and logical operators. Spatial formulae are assertions about the spatial properties (patterns) of a networked system at a particular time instance, i.e. are defined with respect to a QTS at a single time instance. When the spatial formulae are modified by temporal and logical operators, the resulting SpaTeL formula express behaviors of sequences of patterns, i.e. are defined with respect to traces of a networked system.

Definition 2. (SpaTeL syntax) The syntax of a spatial formula is defined as

$$\begin{aligned} \varphi ::= \quad & \top | m \sim d | \neg\varphi | \varphi_1 \wedge \varphi_2 | \exists_B \bigcirc \varphi | \forall_B \bigcirc \varphi | \\ & \exists_B \varphi_1 U_k \varphi_2 | \forall_B \varphi_1 U_k \varphi_2, \end{aligned} \tag{1}$$

where $\sim \in \{\geq, \leq\}$, $d \in [0, b]$, $b \in \mathbb{R}_+$, $k \in \mathbb{N}_{>0}$, $B \subseteq \mathcal{D} := \{NW, NE, SE, SW\}$ with $B \neq \emptyset$, and $m \in \Sigma$ where Σ is as defined for a QTS. U_k and $\bigcirc$ are read as "until" and "next". The syntax of a SpaTeL formula is defined as

$$\Phi ::= \varphi | \neg\Phi | \Phi_1 \wedge \Phi_2 | \Phi_1 U_I \Phi_2, \tag{2}$$

where I is a time interval such that $I := [I_1, I_2)$, I_1, I_2 are non-negative and finite, and φ is a spatial formula.

Other spatial and temporal operators can be derived from the until operator as follows:

$$\begin{aligned} \exists_B F_k \varphi &:= \exists_B \top U_k \varphi & & \\ \exists_B G_k \varphi &:= \neg\forall_B F_k \neg\varphi & F_I \Phi &:= \top U_I \Phi \\ \forall_B F_k \varphi &:= \forall_B \top U_k \varphi & G_I \Phi &:= \neg F_I \neg\Phi. \\ \forall_B G_k \varphi &:= \neg\exists_B F_k \neg\varphi & & \end{aligned} \tag{3}$$

At this point, it becomes clear that SpaTeL is an integration of TSSL [2] and Signal Temporal Logic (STL) [21, 19]. Its spatial formulae share the same syntax with TSSL, while the temporal operators are defined similarly to their STL counterparts with predicates replaced by spatial formulae. At first glance, it may seem that requiring a spatial formula to be nested inside a temporal formula unnecessarily diminishes the expressivity of the logic, i.e. eliminates the interaction between temporal and spatial aspects. However, as pointed out in [20], allowing spatial and temporal operators to be nested arbitrarily can lead to undecidable cases. To further combat undecidability, we bounded time and space.

4.2 Semantics

We define the qualitative semantics of SpaTeL as follows.

Definition 3. (Qualitative Semantics of SpaTeL) Let Q be a trace of a networked system. The *qualitative semantics* of SpaTeL are defined recursively as [1]

$$
\begin{aligned}
(Q,t) &\models \neg\Phi \Leftrightarrow && (Q,t) \not\models \Phi \\
(Q,t) &\models \Phi_1 \wedge \Phi_2 \Leftrightarrow && (Q,t) \models \Phi_1 \wedge (Q,t) \models \Phi_2 \\
(Q,t) &\models \Phi_1 U_{[I_1,I_2)} \Phi_2 \Leftrightarrow && \exists t' \in [t+I_1, t+I_2) : (Q,t') \models \Phi_2 \\
& && \wedge \forall t'' \in [t,t'), (Q,t'') \models \Phi_1 \\
(Q,t) &\models \varphi \Leftrightarrow && (Q,a_0,t) \models \varphi \\
(Q,a,t) &\models \top && \\
(Q,a,t) &\models m \sim d \Leftrightarrow && [a(t)](m) \sim d \\
(Q,a,t) &\models \varphi_1 \wedge \varphi_2 \Leftrightarrow && (Q,a,t) \models \varphi_1 \wedge (Q,a,t) \models \varphi_2 \\
(Q,a,t) &\models \neg\varphi \Leftrightarrow && (Q,a,t) \not\models \varphi \\
(Q,a,t) &\models \exists_B \bigcirc \varphi \Leftrightarrow && \exists a'(t) : ((a(t),a'(t)) \in \tau(t) \wedge \\
& && L(t)(a(t),a'(t)) \cap B \neq \emptyset), \\
& && (Q,a',t) \models \varphi \\
(Q,a,t) &\models \forall_B \bigcirc \varphi \Leftrightarrow && \forall a'(t) : ((a(t),a'(t)) \in \tau(t) \wedge \\
& && L(t)(a(t),a'(t)) \cap B \neq \emptyset), \\
& && (Q,a',t) \models \varphi \\
(Q,a,t) &\models \exists_B \varphi_1 U_k \varphi_2 \Leftrightarrow && \exists \pi^B \in LPaths^B(a(t)) : \\
& && \exists i \in (0,k] : (Q,\pi_i^B,t) \models \varphi_2 \wedge \\
& && \forall j \in [0,i), (Q,\pi_j,t) \models \varphi_1 \\
(Q,a,t) &\models \forall_B \varphi_1 U_k \varphi_2 \Leftrightarrow && \forall \pi^B \in LPaths^B(a(t)) : \\
& && \exists i \in (0,k] : (Q,\pi_i^B,t) \models \varphi_2 \wedge \\
& && \forall j \in [0,i), (Q,\pi_j,t) \models \varphi_1.
\end{aligned}
$$

The trace Q satisfies Φ if $(Q,0) \models \Phi$. The qualitative semantics can be used to check whether a model satisfies or violates a dynamic pattern expressed in SpaTeL. However, it does not provide any information about how strongly the property is satisfied or violated. Quantitative semantics were proposed in [14] and [12] to provide a measure of satisfiability of a trace with respect to a STL formula. Similarly, [2] proposes a quantitative semantics that measures satisfiability of a pattern with respect to a TSSL formula. In the following, we integrate these two sets of semantics.

Definition 4. (Quantitative Semantics of SpaTeL) The quantitative valuation ρ_t of a SpaTeL spatial formula can be calculated according to the recursive quantitative semantics

$$
\begin{aligned}
\rho_t(\neg\Phi, Q, t) &= -\rho_t(\Phi, Q, t) \\
\rho_t(\Phi_1 \wedge \Phi_2, Q, t) &= \min(\rho_t(\Phi_1, Q, t), \rho_t(\Phi_2, Q, t)) \\
\rho_t(\Phi_1 U_{[I_1,I_2)} \Phi_2, Q, t) &= \sup_{t' \in [t+I_1, t+I_2)} (\min(\rho_t(\Phi_2, Q, t'), \\
& \quad \inf_{t'' \in [t+I_1, t')} \rho_t(\Phi_1, Q, t''))) \\
\rho_t(\varphi, Q, t) &= \rho_s(\varphi, a_0(t))
\end{aligned}
$$

and

$$
\begin{aligned}
\rho_s(\top, a) &= b \\
\rho_s(m \sim d, a) &= (\sim \text{ is } \geq)?([m](a) - d) : (d - [m](a)) \\
\rho_s(\neg\varphi, a) &= -\rho_s(\varphi, a) \\
\rho_s(\varphi_1 \wedge \varphi_2, a) &= \min(\rho_s(\varphi_1, a), \rho_s(\varphi_2, a)) \\
\rho_s(\exists_B \bigcirc \varphi, a) &= 0.25 \max_{\pi^B \in LPath^B(a)} \rho_s(\pi_1^B) \\
\rho_s(\forall_B \bigcirc \varphi, a) &= 0.25 \min_{\pi^B \in LPath^B(a)} \rho_s(\pi_1^B) \\
\rho_s(\exists_B \varphi_1 U_k \varphi_2) &= \sup_{\pi^B \in LPath^B(a), i \in (0,k]} (\min(0.25 \\
& \quad \rho_s(\varphi_2, \pi_i^B), \inf_{j \in [0,i)} 0.25^j \rho_s(\varphi_1, \pi_j^B))) \\
\rho_s(\forall_B \varphi_1 U_k \varphi_2) &= \inf_{\pi^B \in LPath^B(a), i \in (0,k]} (\min(0.25 \\
& \quad \rho_s(\varphi_2, \pi_i^B), \inf_{j \in [0,i)} 0.25^j \rho_s(\varphi_1, \pi_j^B))).
\end{aligned}
$$

With a slight abuse of notation, the quantitative semantics of a formula Φ with respect to a trace Q is denoted $\rho_t(\Phi, Q) = \rho_t(\Phi, Q, 0)$.

Remark 1. We restrict the spatial configuration to QTSs constructed from a system S modeled as a $K \times K$ grid. The syntax and semantics of SpaTeL can easily be modified to describe a networked system with a spatial configuration that can be represented as a transition system constructed from a generic tree structure. Extension to systems with general configurations will be studied in the future.

Remark 2. The absolute value of this quantitative valuation can be viewed as a measure of "distance to satisfaction". In other words, larger values correspond to traces that satisfy the formula better than traces with smaller quantitative valuation. Therefore, traces with a larger quantitative valuation are expected to conform quite strongly to the spatial and temporal patterns described by the given formula. For this reason, we refer to the value of the quantitative semantics of a formula with respect to a trace as its degree of satisfaction.

Remark 3. Discounting reduces the effect of deeper nodes in a quad tree, which correspond to more local portions of the network. This leads to a better description of global patterns.

We now show that given a trace Q and a SpaTeL formula Φ the sign of the quantitative evaluation ρ_t is consistent with the violation or satisfaction of the formula. That is, if $\rho_t(\Phi, Q, 0)$ is positive, then Q satisfies Φ and if it is negative, Q does not satisfy Φ.

THEOREM 1 (**Soundness**). *Let Φ be a SpaTeL formula and Q be a trace of a networked system. Then, the following properties hold for the two semantics:*

$$
\begin{aligned}
\rho_t(\Phi, Q, 0) > 0 &\Rightarrow Q \models \Phi \\
\rho_t(\Phi, Q, 0) < 0 &\Rightarrow Q \not\models \Phi
\end{aligned}
\tag{4}
$$

PROOF. (Sketch) Our previous results in [2] showed that the following properties hold for the spatial fragment of SpaTel:

$$
\begin{aligned}
\rho_s(\varphi, a_0) > 0 &\Rightarrow (Q,a,t) \models \varphi \\
\rho_s(\varphi, a_0) < 0 &\Rightarrow (Q,a,t) \not\models \varphi
\end{aligned}
\tag{5}
$$

SpaTeL is a special case of STL [11] where the predicates defined over signals are substituted with a TSSL spatial formula. The proof of soundness for STL can be then derived by structural induction on the operational semantics following the ideas from [11]. □

[1] (Q,t) is used to define the semantics of trace Q at time t (not to be confused with the value of Q at time t, $Q(t)$). Similarly, (Q,a,t) is used to define the semantics of trace Q at time t and state a.

Now, we define the QTS max distance, a measure of similarity of two given QTSs.

Definition 5. (QTS Max Distance) The max distance of two QTSs $Q^{(1)} = (A^{(1)}, a_0^{(1)}, \tau^{(1)}, \Sigma, [.]^{(1)}, L^{(1)})$ and $Q^{(2)} = (A^{(2)}, a_0^{(2)}, \tau^{(2)}, \Sigma, [.]^{(2)}, L^{(2)})$ is defined as:

$$d_\infty(Q^{(1)}, Q^{(2)}) = n_\infty(a_0^{(1)}, a_0^{(2)}, 0)$$

where $n_\infty : \mathcal{A} \times \mathcal{A} \times \mathbb{N} \to [0, b]$ such that:

$$n_\infty(a^{(1)}, a^{(2)}, k) = \begin{cases} \frac{1}{4^k} \underset{m \in \Sigma}{max} |[a^{(1)}]^{(1)}(m) - [a^{(2)}]^{(2)}(m)| \\ \quad \text{if } (a^{(1)}, a^{(1)}) \in \tau^{(1)} \wedge (a^{(2)}, a^{(2)}) \in \tau^{(2)} \\ max_{d \in \mathcal{D}} n_\infty(a_d^{(1)}, a_d^{(2)}, k+1) \text{ otherwise} \end{cases}$$

We now introduce a second theorem, showing that given a property Φ, if two bounded traces of QTS are similar enough, i.e. their max distance is less than the robustness value for the given formula, then if one trace satisfies the formula Φ implies that the other trace satisfies the same formula.

THEOREM 2 (**Correctness**). *Given two traces $Q^{(1)}(t)$ and $Q^{(2)}(t)$ of bounded length T then:*

$$(Q^{(1)}, 0) \models \Phi \wedge \| \mathbf{D}_\infty \|_\infty < \rho_t(\Phi, Q^{(1)}, 0) \Rightarrow (Q^{(2)}, 0) \models \Phi$$

with $\mathbf{D}_\infty$ defined as:

$$\mathbf{D}_\infty = [d_\infty(Q^{(1)}(0), Q^{(2)}(0)), \cdots, d_\infty(Q^{(1)}(T-1), Q^{(2)}(T-1))]$$

PROOF. (Sketch) The proof for the temporal fragment of SpaTeL is analogous to the one for STL in [14]. We provide here a sketch for the proof for the nested spatial fragment:

$$(Q^{(1)}, t) \models \varphi \wedge d_\infty(Q^{(1)}(t), Q^{(2)}(t)) < \rho_s(\varphi, a_0^{(1)}(t)) \Rightarrow (Q^{(2)}, t) \models \varphi$$

We can distinguish the following cases:

case $\varphi := \top$: in this case the theorem is true following the definition of the qualitative semantics.

case $\varphi := m \sim d$: This proof assumes $\sim :=$ " $\geq$ ". A similar proof can be derived from $\sim :=$ " $\leq$ ". We have that:

$$d_\infty(Q^{(1)}(t), Q^{(2)}(t)) < \rho_s(m \geq d, Q^{(1)}(t)) = [a_0^{(1)}(t)]^{(1)}(m) - d$$

Moreover, by the definition of d_∞ we have that:

$$|[a_0^{(1)}(t)]^{(1)}(m) - [a_0^{(2)}(t)]^{(2)}(m)| \leq d_\infty(Q^{(1)}(t), Q^{(2)}(t))$$
$$[a_0^{(1)}(t)]^{(1)}(m) - |[a_0^{(2)}(t)]^{(2)}(m) - [a_0^{(2)}(t)]^{(2)}(m)| - d > 0$$
$$[a_0^{(1)}(t)]^{(1)}(m) - |[a_0^{(2)}(t)]^{(2)}(m) - [a_0^{(2)}(t)]^{(2)}(m)| \leq [a_0^{(2)}(t)]^{(2)}(m)$$

From Theorem 1 we have that:

$$[a_0^{(2)}(t)]^{(2)}(m) - d > 0 \Rightarrow \rho_t(m \geq d, Q^{(2)}(t)) > 0 \Rightarrow (Q^{(2)}, t) \models \varphi$$

all other cases:

if $(Q^{(1)}, t) \models \varphi$ then we have:

$$\rho_s(\varphi, a_0^{(1)}(t)) = \begin{cases} (1)\text{: } \frac{1}{4^j} b\text{: } j \in \mathbb{N} \\ \text{or (2):} \\ \frac{1}{4^j}([a^{(1)}(t)]^{(1)}(m) - d) : j \in \mathbb{N}, a^{(1)} \in A^{(1)}, m \in \Sigma \end{cases}$$

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 $\varphi := m \sim d$. □

Example 1. cont'd.

The original checkerboard pattern can be described by a TSSL formula

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

where $B^* = \{SW, NE, NW, SE\}$. With SpaTeL, we can formulate the spatial-temporal pattern "All the tiles in the checkerboard flip their colors simultaneously at some time $t \in (0, T]$" as

$$\begin{aligned} \Phi := F_{(0,T]} \quad & ((\forall_{B^*} F_1((\forall_{\{SW,NE\}} \bigcirc (W \geq 1)) \\ & \wedge (\forall_{\{NW,SE\}} \bigcirc (W \leq 0))) \wedge \\ & G_{(0,1]}(\forall_{B^*} F_1((\forall_{\{NW,SE\}} \bigcirc (W \geq 1)) \wedge \\ & (\forall_{\{SW,NE\}} \bigcirc (W \leq 0))))). \end{aligned} \quad (6)$$

5. VERIFICATION AND SYNTHESIS

In this section we show that standard model checking and parameter synthesis algorithms presented in [29] and [2] can easily be applied to SpaTeL.

5.1 Statistical Model Checking

We are interested in determining whether the traces produced by a networked system S satisfy a given SpaTeL formula Φ. In general, a networked system produces traces non-deterministically due to variations in initial conditions, system parameters, or un-modeled dynamics. Let the non-determinism of the system be described by a random variable U with range space R_U such that a given realization u generates a unique trace Q_u and $\mathcal{L}(S) = \bigcup_{u \in R_U} Q_u$. Enumerating each trace Q_u and checking $Q_u \models \Phi$ is infeasible, as this set is (possibly) countably infinite. A traditional model checking algorithm [3], in which a formal proof of whether or not all traces produced by S satisfy Φ is generated, is also likely infeasible to implement due to the potential size and complexity of networked systems. Further, the question of whether or not all traces of S satisfy Φ may be too narrow of an inquiry, as we may only be interested in how frequently Φ is satisfied. Therefore, we propose to characterize the behavior of S by using the model to randomly generate a finite number of traces and solving the broader (and more feasible) statistical model checking problem.

Problem 1. A model of a networked dynamical system S and a SpaTeL formula Φ are given. Let $p = Pr[Q \models \Phi]$, $c \in (\frac{1}{2}, 1)$ be a confidence level, and $\delta > 0$ be a half-interval size. Find an interval (p_0, p_1) where $0 \leq p_0 < p_1 \leq 1$ and $p_1 - p_0 = 2\delta$ such that $Pr[p \in (p_0, p_1)] \geq c$.

Problem 1 asks for a confidence interval (p_0, p_1) because this interval can be calculated from a finite number of traces of S while explicitly calculating p in general would require an infinite number of traces.

Example 1. cont'd.

Each of the tiles $S_{i,j}$ flips its color at a random time $T_{i,j}$ distributed non-uniformly over the range $l_{i,j}, \ldots, h_{i,j}$. Let U be the random matrix $[T_{i,j}]$. We want to estimate (via a confidence interval) the probability that all the tiles in the checkerboard flip their color simultaneously at some time t, i.e. the probability of satisfying (6).

Our statistical model checking procedure is summarized in Algorithm 1. Algorithm 1 uses Bayesian Interval Estimation, an algorithm presented in [29] to recursively construct

a confidence interval for the mean of a Bernoulli random variable. We define an i.i.d. sequence of Bernoulli random variables $\{\chi_i\}_{i=1}^N$ such that

$$\chi_i = I(Q_{u_i} \models \Phi), \tag{7}$$

where u_i is a sample drawn from U and I is the indicator function. The sample mean of the variables $\{\chi_i\}_{i=1}^N$ thus approaches p as defined in Problem 1 as $N \to \infty$. For a finite value of N, we can estimate an interval $[\hat{p} - \delta, \hat{p} + \delta]$ such that

$$Pr[p \in (\hat{p} - \delta, \hat{p} + \delta)] \geq c, \tag{8}$$

where δ and c are as defined in Problem 1. This interval is called the "100c percent Bayesian interval estimate of p". The Bayesian interval estimation algorithm proceeds by collecting samples of χ_i and recursively applying Bayes' rule

$$f(p|\chi_i, \ldots, \chi_n) = \frac{f(\chi_1, \ldots, \chi_n|p)g(p)}{\int_0^1 f(\chi_1, \ldots, \chi_n|v)g(v)dv}, \tag{9}$$

where f and g are probability density functions (pdfs), until the estimated interval achieves the desired confidence level c. Recursively applying Bayes' rule requires us to have some prior pdf of p over the interval $[0, 1]$ before collecting the first sample χ_1. We use β priors as suggested in [29], where a β random variable has a pdf

$$\begin{aligned} \forall p \in [0,1] \qquad & g(p, \alpha, \beta) = \tfrac{1}{B(\alpha,\beta)} p^{\alpha-1}(1-p)^{\beta-1} \\ & B(\alpha, \beta) = \textstyle\int_0^1 t^{\alpha-1}(1-t)^{\beta-1}dt. \end{aligned} \tag{10}$$

and a cumulative density function (cdf)

$$F_{\alpha,\beta}(p) = \int_0^p g(t, \alpha, \beta)dt. \tag{11}$$

In the absence of a principled guess of the prior density of p, we make the assumption of uniform density, e.g. $F_{\alpha,\beta}(p) = p$, by setting $\alpha = \beta = 1$. If we use the β prior, the posterior mean $\hat{p}$ can be calculated as

$$\hat{p} = \frac{X + \alpha}{n + \alpha + \beta} \tag{12}$$

where $X = \sum_{i=1}^n \chi_i$. The posterior probability is:

$$Pr[p \in (p_0, p_1)] = F_{(X+\alpha, n-X+\beta)}(p_1) - F_{(X+\alpha, n-X+\beta)}(p_0). \tag{13}$$

5.2 Parameter Synthesis

Consider the case in which in addition to stochastic parameters U, the traces of S are influenced by some design parameters $\Pi \in \mathcal{P}$. Let $Q_{u,\Pi}$ be the unique trace associated with realization u and parameters Π such that $\mathcal{L}(S) = \bigcup_{u \in R_U} \bigcup_{\Pi \in \mathcal{P}} Q_{u,\Pi}$. We wish to find a parameterization Π^* such that the traces $\{Q_{u,\Pi^*}\}_{u \in R_U}$ strongly satisfy some desirable property Φ on average. Formally, we have:

Problem 2. A model of a system S, a SpaTeL formula Φ, and ranges of system parameters $\mathcal{P} = \mathcal{P}_1 \times \ldots \mathcal{P}_{|\Pi|}, \mathcal{P}_i \subset \mathbb{R}, i = 1, \ldots, |\Pi|$ are given. Determine the parameterization $\Pi^* \in \mathcal{P}$ such that the average degree of satisfaction of the resulting traces with respect to Φ is maximized.

Example 1. cont'd.

Assume that the maximum possible flipping time $h_{i,j}$ for each subsystem $S_{i,j}$ is randomly selected according to an exponential distribution with rate Π. A parameter synthesis problem is to determine the value of Π such that on average each tile flips its color as close to simultaneously as possible.

Algorithm 1: Statistical Model Checking

Input: Φ, S, $\delta \in (0, 1/2)$, $c \in (1/2, 1)$, α, β
Output: (p_0, p_1)

```
n ← 0 (number of traces drawn so far) ;
X ← 0 (number of traces satisfying Φ) ;
γ ← 0 (coverage probability of the interval (p0, p1)) ;
while γ < c do
    Qu ← draw a sample trace of the system ;
    n ← n + 1 ;
    if (Qu ⊨ Φ) then
        X ← X + 1;
    end
    p̂ ← (X + α)/(n + α + β) (compute posterior mean) ;
    (p0, p1) ← (p̂ − δ, p̂ + δ) (interval estimate) ;
    if p1 > 1 then
        (p0, p1) ← (1 − 2δ, 1)
    else
        if p0 < 0 then
            (p0, p1) ← (0, 2δ)
        end
    end
    γ ← Posterior probability of p ∈ (p0, p1) computed
    by (13)
end
```

The temporal version of problem 2 has been solved in [1, 26].We study the spatio-temporal case in this paper. Problem 2 can be formulated as an optimization problem:

$$\Pi^* = \arg\max_{\Pi \in \mathcal{P}} E_U[\rho_t(\Phi, Q_{u,\Pi})]. \tag{14}$$

We do not evaluate the expectation directly but rather approximate it by generating N traces $\{Q_{u_i,\Pi}\}_{i=1}^N$ and calculating the sample mean of the degree of satisfaction. In practice, we have found that relatively small sample sizes N sufficed for parameter synthesis.

In [2], the minimum value of the degree of satisfaction over all possible executions was maximized. The authors argue that if we make sure that the lowest possible quantitative valuation (which corresponds to the worst execution of the system with respect to the given specification) is still positive, then every possible execution will satisfy the specification. We have replaced the minimum degree of satisfaction with the sample mean because we deal with stochastic systems such that even for the optimal parameters, there may still be some traces among the samples where the specification is not satisfied. Further, the sample mean is a better indicator of the behavior of a typical execution of the system than the robustness of the "worst case" trace.

The process of parameter synthesis using particle swarm optimization is summarized in Algorithm 2. Many continuous optimization methods can be used to solve (14). In this paper, we use particle swarm optimization (PSO) [24], a randomized search algorithm in which a collection of points in the search space are updated at each iteration to move closer (on average) to a global optimal solution. The choice of PSO was motivated by [2] where it was chosen due to its inherent distributed nature and its ability to operate on ir-

regular search spaces. In particular, PSO does not require a differentiable objective function. The PSO procedure begins by randomly initializing a set of m particles with positions $z_i \in \mathcal{P}$ and velocities $v_i \in \mathcal{V} \subseteq R^{|\Pi|}$. The position of a particle represents a candidate solution to (14) and the velocity represents a search direction from the current solution. After the particles are generated, m sets of N traces $\{Q_{u_j,z_i}\}_{j=1}^{N}$ are produced and used to evaluate (14) for each point represented by the particles. The position of the ith particle that has performed the best so far is stored in the variable z_i^{best} and the optimal value of z_i^{best} is stored in the variable z^{best}. After all particles have been evaluated, their positions and velocities are updated according to the relations

$$v_i \leftarrow W v_i + \eta(0, r_p)(z_i^{best} - z_i) + \eta(0, r_g)(z^{best} - z_i)$$
$$z_i \leftarrow z_i + v_i. \tag{15}$$

$\eta(0, r_j)$ represents a random number uniformly distributed over the interval $[0, r_j]$. The parameters $W \in \mathbb{R}$, r_p, and r_g are specified by the user. The iterative process of updating particles and evaluating their performance proceeds until a termination criterion is met, e.g., z^{best} does not change after k consecutive iterations.

Algorithm 2: Parameter Synthesis

Input: Φ, S, $\mathcal{P}$, N, (W, r_p, r_g, m), k
Output: Π^*

```
for 1 ≤ j ≤ m do
    z_i ← initialize particle positions;
    v_i ← initialize particle velocities
end
while Π* has changed during the last k iterations do
    for 1 ≤ j ≤ N do
        Q_{u_j,z_i} ← draw a sample trace of the system ;
        ρ_t(Φ, Q_{u_j,z_i}) ← calculate quantitative valuation
        of Q_{u_j,z_i} with respect to Φ ;
    end
    [z_i, v_i] ← update particles according to (15) ;
    Π* ← the best position so far (z^best)
end
```

5.3 Complexity

The time complexity of executing either Algorithm 1 or 2 is difficult to compute explicitly. Each algorithm proceeds in an iterative fashion until a convergence criterion is met. Thus, the complexity depends on system-dependent convergence rates that can vary widely among application areas. Some insight into how different stopping criteria affect convergence of particle swarm optimization for a given objective function is given in [28]. Here, we establish the complexity of computing the degree of satisfaction for an execution trace of a networked dynamical system where the size of the network is $K \times K$ and the traces have a length of T. This is the core computational procedure that is executed during every iteration of each algorithm. The temporal and spatial portions of SpaTeL are inspired by STL and TSSL, respectively. The worst-case complexity of computing the degree of satisfaction of an STL formula was established as $\mathcal{O}(T^{2l_t})$ in [12] where l_t is the maximum number of nested temporal until operators in the formula. The quantitative semantics of SpaTeL is defined in such a way that the quantitative valuation for spatial and temporal until operators are computed using the same expressions. Therefore, computing the spatial portion of the SpaTeL quantitative valuation for a given quad tree has a complexity of $\mathcal{O}(4^{n_s}.n_s^{2l_s})$ where $n_s = \log K$ is the depth of the tree and l_s is the maximum number of nested spatial until operators. Finally, constructing of a quad tree from a $K \times K$ grid needs $\mathcal{O}(K^2 \log K)$ operations. Consequently, the total complexity is $\mathcal{O}(T^{2l_t} K^4 (\log K)^{2l_s+1})$.

6. CASE STUDIES

The matlab package SPATEL, which includes our implementations of Algorithms 1 and Algorithms 2 and the simulation software we used in our case studies, is available at `http://sites.bu.edu/hyness/software/`.

6.1 Reaction-Diffusion System

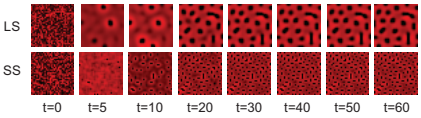


Figure 2: Patterns generated by system (16) with $R = [1, -12, -1, 16]$ and different diffusion parameters $D_{LS} = [5.6, 24.5]$ and $D_{SS} = [1.4, 5.3]$. The steady state observations produce large spots (LS) and small spots (SS).

Inspired by [27], we consider a reaction-diffusion system that describe the generation of skin pigments that produce spots in animals. Following [2], we consider a 32×32 reaction-diffusion system with two species. The concentrations of the species in $S_{i,j}$ evolve according to the ODE

$$\frac{dx_{i,j}^{(1)}}{dt} = D_1(\mu_{i,j}^{(1)} - x_{i,j}^{(1)}) + R_1 x_{i,j}^{(1)} x_{i,j}^{(2)} - x_{i,j}^{(1)} + R_2$$
$$\frac{dx_{i,j}^{(2)}}{dt} = D_2(\mu_{i,j}^{(2)} - x_{i,j}^{(2)}) + R_3 x_{i,j}^{(1)} x_{i,j}^{(2)} + R_4 \tag{16}$$

where $x_{i,j}^{(1)}$ and $x_{i,j}^{(2)}$ represent the concentration of each species at cell (i,j), $\mu_{i,j}^{(1)}$ and $\mu_{i,j}^{(2)}$ are the inputs of the (i,j)th system $S_{i,j}$ from neighboring systems,

$$\mu_{i,j}^{(n)} = \frac{1}{|\nu_{i,j}|} \sum_{\nu \in \nu_{i,j}} x_\nu^{(n)}, \tag{17}$$

$\nu_{i,j}$ is the set of indices of systems adjacent to $S_{i,j}$, $D_i, i = 1, 2$ are diffusion coefficients, and $R_i, i = 1, \ldots, 4$ are the parameters that define the local dynamics for the two species.

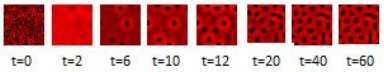
Fig. 2 shows the observed concentrations of species 1 at different time points for two different parameter choices. The more black cell (i,j) is, the larger the value of $x_{i,j}^{(1)}$. The two corresponding steady-state spatial patterns present at time $t = 60$ are called large spots (LS) and small spots (SS), respectively. In [2], the RIPPER supervised learning algorithm was used to learn the TSSL formulae φ_{LS} and φ_{SS} from examples of large and small spot patterns that were labeled manually. These formulae are too long to be displayed here. SpaTeL, a richer logic, can not only characterize these spatial patterns but also capture how they develop over time. Consider the following formulae:

$$\Phi_1 : F_{[0,30)} G_{[0,60)} \varphi_{SS} \tag{18}$$

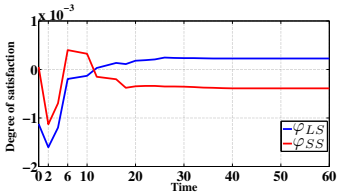
(D_1, D_2)	δ	c	$\hat{p}$	n
(1.44,5.27)	0.05	0.95	0.95	156
	0.05	0.99	0.95	538
	0.01	0.95	0.95	3766
	0.01	0.99	0.96	284
(5.6,24.5)	0.05	0.95	0.11	98
	0.05	0.99	0.13	71
	0.01	0.95	0.11	279
	0.01	0.99	0.11	318
(0.2,20)	0.05	0.95	0.03	28
	0.05	0.99	0.02	43
	0.01	0.95	0.007	148
	0.01	0.99	0.004	228

Table 1: Satisfaction probabilities for Φ_1. Each iteration took on average approximately 0.74 seconds on a machine with a 2.40 GHz processor and 8 GB RAM.

$$\Phi_2 : F_{[0,30)} G_{[0,60)} \varphi_{LS} \wedge G_{[0,60)} \neg \varphi_{SS}. \tag{19}$$



(a)



(b)

Figure 3: Time evolution of TSSL quantitative semantics for an execution of the reaction-diffusion system with parameter $D = [D_1, D_2] = [5.6, 24.5]$ where LS emerges in steady state and SS emerges in transient state. (a) Generated patterns (b)$\rho_s(\varphi_{LS}, Q(t))$ and $\rho_s(\varphi_{SS}, Q(t))$ with $t = 0, \ldots, 60$.

Formula Φ_1 specifies that the SS pattern appears within the first 30 seconds and persists for 60 seconds after it emerges. Formula Φ_2 is the conjunction of an expression which states that LS pattern emerges within the first 30 seconds and remains for the next 60 seconds and an expression which specifies that the SS pattern never occurs during the first 60 seconds, i.e. the large spots pattern is established unambiguously. The valuations of the quantitative semantics of a system execution with respect to φ_{LS} and φ_{SS} are plotted over time in Fig. 3(b). The figure shows that although this particular set of parameter lead to large spots pattern in steady state, the small spots pattern occur transiently during the system evolution.

We applied the statistical model checking procedure (Algorithm 1) to estimate the probability that formula Φ_1 holds for the Turing system. The results from using three different sets of diffusion rates, two different confidence levels c, and two different half-interval sizes δ are summarized in Table 1.

The system with the first set of parameters $(1.44, 5.27)$ (selected by an expert) satisfies Φ_1 with high probability. The other two parameterizations hold with very low probability. We also see that as c increases and δ decreases, the number of traces, and consequently the computation time, required to find the desired interval increases significantly.

N	(D_1^*, D_2^*)	$\hat{p}$
1	(2.65, 10.00)	0.74
10	(1.90, 7.03)	0.88
100	(1.45, 5.12)	0.96

Table 2: Synthesized diffusion rates for the reaction-diffusion system when the objective is satisfaction of Φ_1 with $\delta = 0.01$ and $c = 0.99$.

Next, we applied the parameter synthesis procedure (Algorithm 2) to find a pair of diffusion coefficients that maximized the expected degree of satisfaction with respect to Φ_1. We increased the number of traces at each iteration N and observed that the results improve as N grows (i.e., the synthesized parameter values correspond to higher probability of satisfaction). The results are illustrated in Table 2. Notice that computing the average quantitative valuation for only ten executions at every step results in a satisfaction probability as high as 88 %.

6.2 Smart Neighborhood Power Management

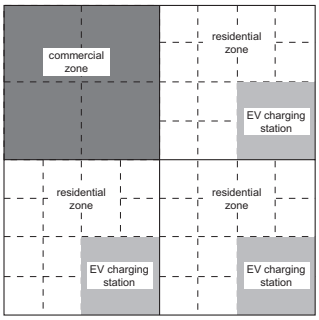


Figure 4: Smart neighborhood example: Configuration of the neighborhood.

In this section, we apply our procedures to a simulation of a smart neighborhood electricity grid whose power consumption is controlled by demand-side management (DSM). Due to the growing share of fluctuating renewable sources such as wind and solar in power generation, it becomes increasingly difficult to maintain the balance between power production and consumption by only managing power generation. Recent advent of the smart grid, a more flexible and reliable grid, enables DSM systems to play a more active role in mitigating the effects of such intermittent resources [25]. A DSM system controls power distribution in a network by varying the prices that consumers pay per unit of consumed power in response to consumer demand. Thus, when the demand for electricity is high, only those members of the market (network) that highly prioritize power consumption at that time will consume electricity.

Consider a Smart Neighborhood Operator (SNO) that manages loads in commercial and residential buildings in a neighborhood as shown in Fig. 4. The neighborhood has commercial buildings located at the northwestern quarter and residential buildings in the other quarters. There is an electrical vehicle (EV) charging station at the southeastern corner of each residential quarter.

(RP_D, RP_N)	δ	c	$\hat{p}$	n
(3.0,5.0)	0.05	0.95	0.95	117
	0.05	0.99	0.95	238
	0.01	0.95	0.94	3207
	0.01	0.99	0.94	3291
(3.3,4.7)	0.05	0.95	0.69	103
	0.05	0.99	0.72	104
	0.01	0.95	0.72	1201
	0.01	0.99	0.72	1200
(4,4)	0.05	0.95	0.04	45
	0.05	0.99	0.02	43
	0.01	0.95	0.02	113
	0.01	0.99	0.02	110

Table 3: Satisfaction probabilities for Φ_3. The price for the commercial district is fixed at 19. Each iteration required on average 0.85 seconds.

Following [10], at time t, inside each building $n_i(t)$ appliances are consuming actively with a rate r_i kW with subscripts c, r and e denoting commercial building, residential building and EV station, respectively. The arrival distribution of appliances for building class i over the period $[t, t+1]$ is Poisson distributed with a rate $\lambda_i(U_i - p_j(t))/U_i$, where U_i is the utility of an appliance of class i and $p_j(t)$ is the broadcast price for neighborhood class $j, j \in \{c, r\}$ with residential building and EV station charged by the same price. Once connected, an appliance continues to consume for a period of time τ_i which is exponentially distributed with rate μ_i. The goal of the SNO is to set the broadcast prices p_j such that the loads of different areas and the whole neighborhood satisfy certain specified load constraints.

The statistical model checking procedure was used to ensure that the power system conformed to the specification "Always ensure that for each of the four 'neighborhoods', the power consumption level m is below 300 and the power consumption is below 200 in each of the neighborhoods' quadrants at least once per hour. Ensure that after 6 hours, the power consumption in all residential areas is above level 3." This is written in SpaTeL as

$$\begin{aligned}\Phi_3 := \quad & G_{[0,18)} F_{[0,1)} (\forall_{(NW,NE,SW,SE)} \bigcirc (m \leq 300 \wedge \\ & \forall_{(NW,NE,SW,SE)} \bigcirc m \leq 200)) \wedge \\ & G_{[6,18)} (\forall_{(NE,SE,SW)} \bigcirc \forall_{(NW,NE,SW)} \bigcirc m \geq 3).\end{aligned}$$

Table 3 shows the probability of satisfaction for the above specification for different choices of daytime and nighttime residential power prices (RP_D, RP_N). The parameters (3,5) lead to a network that rarely violates the given specification. Altering these prices by even a small amount will cause the specification to be violated often.

Now consider the case where we want to synthesize power prices so that: the total power consumption of the commercial buildings is always less than 150; the power consumption is below 150 in each EV station and below 25 in each of the residential neighborhoods in the first 12 hours; after 12 hours, the power consumption of each EV station is between 30 and 200; after 15 hours, the power consumption in all residential areas is above 5. In SpaTeL, these requirements are

$$\begin{aligned}\Phi_4 := \quad & G_{[0,18)} (\forall_{NW} \bigcirc (m \leq 150)) \wedge G_{[0,12)} (\forall_{(NE,SE,SW)} \bigcirc \\ & (\forall_{(NW,NE,SW)} \bigcirc m < 25) \wedge (\forall_{SE} \bigcirc m \leq 150)) \wedge \\ & G_{[12,18)} (\forall_{(NE,SE,SW)} \bigcirc \forall_{SE} \bigcirc (m \leq 200 \wedge m \geq 30)) \\ & \wedge G_{[15,18)} (\forall_{(NE,SE,SW)} \bigcirc \forall_{(NW,NE,SW)} \bigcirc m \geq 5).\end{aligned}$$

The design parameters are daytime and nighttime prices of the residential areas RP_D, RP_N and nighttime power price of commercial areas CP_N. We fixed the daytime power price of factories at 19. Algorithm 2 results in prices specified in Table 4. The demand coefficients shift from daytime values $\lambda_{i,d}$ to nighttime values $\lambda_{i,n}$. The values of the simulation parameters were $(r_c, \lambda_{c,d}, \lambda_{c,n}, U_c) = (20, 0.33, 0.33, 20)$, $(r_r, \lambda_{r,d}, \lambda_{r,n}, U_r) = (4, 0.28, 1.02, 6)$, and $(r_e, \lambda_{e,d}, \lambda_{e,n}, U_e) = (9, 0.28, 1.02, 8)$. Fig. 5 illustrates a few traces that are executed with the final synthesized prices. The vast majority of the simulated traces satisfy the required specifications.

N	(RP_D^*, RP_N^*, CP_N^*)	$\hat{p}$
1	(4.69, 4.41, 19.70)	0.43
10	(4.42, 4.74, 19.70)	0.75
20	(4.40, 4.75, 19.70)	0.73
50	(4.15, 5.05, 19.70)	0.90

Table 4: Synthesized power prices for the smart neighborhood example with $\delta = 0.01$ and $c = 0.99$.

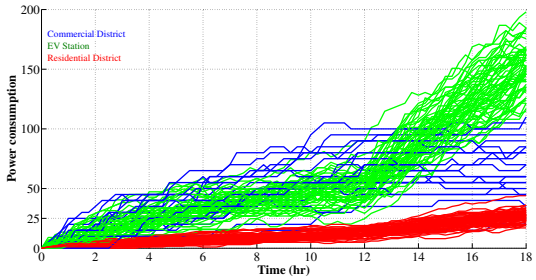


Figure 5: Traces generated the smart neighborhood model when $(RP_D, RP_N) = (4.15, 5.05)$ and $(CP_D, CP_N) = (19, 19.70)$.

7. CONCLUSION

The advent of spatially distributed and networked cyber-physical systems has increased the demand for novel efficient spatio-temporal reasoning techniques. SpaTeL provides an intuitive formal framework to specify the emergent behaviors to be detected or to be enforced. We have demonstrated that SpaTeL can be used to specify, verify, and enforce desired system behaviors. Future research includes using SpaTeL to learn dynamic patterns directly from data and synthesizing control policies to ensure a given dynamic pattern.

8. ACKNOWLEDGMENT

This work was partially supported by ONR under grant ONR N00014-14-1-0554 and NSF under grant CBET-0939511. E.B. and R.G. acknowledge the support of the Austrian FFG project HARMONIA (nr. 845631) and the Doctoral Program Logical Methods in Computer Science funded by the Austrian FWF. The authors would like to acknowledge Ebru Aydin Gol from Google Inc. for providing software and data for the reaction-diffusion system presented in [2] and Bowen Zhang and Michael Caramanis from Boston University for their helpful insight on the smart neighborhood model.

9. REFERENCES

[1] H. Abbas, B. Hoxha, G. Fainekos, and K. Ueda. Robustness-guided temporal logic testing and

verification for stochastic cyber-physical systems. In *Cyber Technology in Automation, Control, and Intelligent Systems (CYBER), 2014 IEEE 4th Annual International Conference on*, pages 1–6. IEEE, 2014.

[2] E. Aydin Gol, E. Bartocci, and C. Belta. A formal methods approach to pattern synthesis in reaction diffusion system. In *IEEE 53rd Annual Conference on Decision and Control*, 2014.

[3] C. Baier, J.-P. Katoen, et al. *Principles of model checking*, volume 26202649. MIT press Cambridge, 2008.

[4] E. Bartocci, L. Bortolussi, and G. Sanguinetti. Data-driven statistical learning of temporal logic properties. In *Formal Modeling and Analysis of Timed Systems*, pages 23–37. Springer, 2014.

[5] B. Bennett, A. Cohn, F. Wolter, and M. Zakharyaschev. Multi-dimensional modal logic as a framework for spatiotemporal reasoning. *Applied Intelligence*, 17(3):239–251, 2002.

[6] L. Bortolussi and L. Nenzi. Specifying and monitoring properties of stochastic spatio-temporal systems in signal temporal logic.

[7] D. Bresolin, P. Sala, D. Della Monica, A. Montanari, and G. Sciavicco. A decidable spatial generalization of metric interval temporal logic. In *Proc. of TIME 2010: the 17th International Symposium on Temporal Representation and Reasoning*, pages 95–102. IEEE, Sept 2010.

[8] S. Bufo, E. Bartocci, G. Sanguinetti, M. Borelli, U. Lucangelo, and L. Bortolussi. Temporal logic based monitoring of assisted ventilation in intensive care. In *Proc. of ISoLA 2014: 6th International Symposium on Leveraging Applications of Formal Methods, Verification and Validation, Part II*, volume 8803 of *LNCS*, pages 391–403. Springer, 2014.

[9] L. Caires and L. Cardelli. A spatial logic for concurrency (part ii). *Theoretical Computer Science*, 322(3):517–565, 2004.

[10] M. C. Caramanis, I. C. Paschalidis, C. G. Cassandras, E. Bilgin, and E. Ntakou. Provision of regulation service reserves by flexible distributed loads. In *IEEE 51st Annual Conference on Decision and Control*, pages 3694–3700, 2012.

[11] A. Donzé, T. Ferrère, and O. Maler. Efficient robust monitoring for STL. In *Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings*, volume 8044 of *Lecture Notes in Computer Science*, pages 264–279. Springer, 2013.

[12] A. Donzé and O. Maler. *Robust satisfaction of temporal logic over real-valued signals*. Springer, 2010.

[13] A. Donzé, O. Maler, E. Bartocci, D. Nickovic, R. Grosu, and S. A. Smolka. On temporal logic and signal processing. In *Proc. of ATVA 2012: the 10th International Symposium on Automated Technology for Verification and Analysis*, volume 7561 of *LNCS*, pages 92–106, 2012.

[14] G. E. Fainekos and G. J. Pappas. Robustness of temporal logic specifications for continuous-time signals. *Theoretical Computer Science*, 410(42):4262–4291, 2009.

[15] J. P. Gollub and J. Langer. Pattern formation in nonequilibrium physics. *Reviews of Modern Physics*, 71(2):S396, 1999.

[16] R. Grosu, S. A. Smolka, F. Corradini, A. Wasilewska, E. Entcheva, and E. Bartocci. Learning and detecting emergent betavior in networks of cardiac myocytes. *Communication of ACM*, 52(3):97–105, 2009.

[17] X. Jin, A. Donze, J. Deshmukh, and S. Seshia. Mining requirements from closed-loop control models. In *Hybrid Systems: Computation and Control (HSCC)*, 2013.

[18] A. Jones, Z. Kong, and C. Belta. Anamoly detection in cyber-physical systems: A formal methods approach. In *the 53rd IEEE Conference on Decision and Control*, 2014.

[19] Z. Kong, A. Jones, A. Medina Ayala, E. Aydin Gol, and C. Belta. Temporal logic inference for classification and prediction from data. In *Proceedings of the 17th international conference on Hybrid systems: computation and control*, pages 273–282. ACM, 2014.

[20] R. Kontchakov, A. Kurucz, F. Wolter, and M. Zakharyaschev. Spatial logic + temporal logic = ? *Handbook of Spatial Logics*, pages 497–564, 2007.

[21] O. Maler and D. Nickovic. Monitoring temporal properties of continuous signals. In *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, pages 152–166. Springer, 2004.

[22] M. Marx and M. Reynolds. Undecidability of compass logic. *Journal of Logic and Computation*, 9(6):897–914, 1999.

[23] J. K. Parrish and L. Edelstein-Keshet. Complexity, pattern, and evolutionary trade-offs in animal aggregation. *Science*, 284(5411):99–101, 1999.

[24] R. Poli, J. Kennedy, and T. Blackwell. Particle swarm optimization. *Swarm intelligence*, 1(1):33–57, 2007.

[25] F. Saffre and R. Gedge. Demand-side management for the smart grid. In *Network Operations and Management Symposium Workshops (NOMS Wksps), 2010 IEEE/IFIP*, pages 300–303. IEEE, 2010.

[26] S. Sankaranarayanan and G. Fainekos. Falsification of temporal properties of hybrid systems using the cross-entropy method. In *Proceedings of the 15th ACM international conference on Hybrid Systems: Computation and Control*, pages 125–134. ACM, 2012.

[27] A. M. Turing. The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, 237(641):37–72, 1952.

[28] K. Zielinski and R. Laur. Stopping criteria for a constrained single-objective particle swarm optimization algorithm. *Informatica (Slovenia)*, 31(1):51–59, 2007.

[29] P. Zuliani, A. Platzer, and E. M. Clarke. Bayesian statistical model checking with application to simulink/stateflow verification. In *Proceedings of the 13th ACM international conference on Hybrid systems: computation and control*, pages 243–252. ACM, 2010.